IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 August 2025, accepted 21 August 2025, date of publication 25 August 2025, date of current version 29 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3602562

RESEARCH ARTICLE

Enhanced Intrusion Detection via Hybrid Data Resampling and Feature Optimization

NAHIDA NIGAR[1,2] **AND RASHED MUSTAFA**[2]
[1]Department of Computer Science and Engineering, East Delta University, Chattogram 4209, Bangladesh
[2]Department of Computer Science and Engineering, University of Chittagong, Chattogram 4331, Bangladesh
Corresponding author: Nahida Nigar (nahida.n@eastdelta.edu.bd)

ABSTRACT Network security is increasingly challenged by sophisticated and evolving cyber threats, underscoring the need for efficient and robust intrusion detection systems (IDS). Although researchers have extensively studied intrusion detection, they have made limited progress in effectively detecting zero-day attacks, particularly those caused by subtle variations. Existing approaches often tackle class imbalance and feature selection independently, with minimal exploration of their combined impact as a unified preprocessing strategy. Moreover, researchers have shown that inconsistencies and artifacts in the widely used CICIDS2017 dataset impair model generalization and stability. This study proposes a systematic pipeline that employs a unified framework, integrating hybrid data balancing with advanced feature selection to enhance anomaly detection performance. By leveraging Extra Trees-based feature selection alongside a strategic combination of SMOTE and targeted undersampling, the approach effectively addresses critical challenges such as class imbalance and feature redundancy. Evaluated on the CICIDS2017 dataset, the framework demonstrates strong generalization across both machine learning and deep learning models. Notably, XGBoost and Decision Tree achieved balanced accuracy and F1-scores of up to 99.99% and 99.96%, respectively, in multiclass classification. The evaluation framework reports comprehensive metrics beyond accuracy, including Precision, Recall, F1-score, Matthews Correlation Coefficient (MCC), and ROC-AUC, ensuring robust assessment under class imbalance conditions. These results highlight the effectiveness of the approach for real-time anomaly detection. The study contributes a robust and efficient pipeline for improving the accuracy and reliability of anomaly-based intrusion detection systems.

INDEX TERMS Anomaly detection, class imbalance, decision tree, feature selection, intrusion detection systems (IDS), K-nearest neighbor (KNN), LSTM, multiclass classification, machine learning, SMOTE, transformer, XGBoost.

I. INTRODUCTION

Network attacks involve malicious activities aimed at disrupting, denying, degrading, or destroying information and services within computer networks. These attacks target network systems' Integrity, Confidentiality, and Availability [4]. Standard attack methods include e-mail viruses, system probing, internet worms, unauthorized access, and denial-of-service (DoS) attacks [1]. Attackers typically use four main approaches: Social Engineering, Masquerading, Implementation Vulnerability, and Abuse of Functionality [4]. These attack strategies compromise system performance and security, highlighting the need for robust intrusion detection and prevention measures.

Intrusion detection works on the principle that abnormal behavior differs from normal behavior and can therefore be detected. Detection methods are generally categorized into three types: (1) Anomaly-based detection, which identifies deviations from established normal patterns; (2) Signature-based detection, which compares network traffic to known attack patterns; and (3) Specification-based detection, which flags deviations from pre-defined rules created by experts. These methods help monitor network activity and identify suspicious behavior.

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Masini.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

The foundations of intrusion detection date back to the early 1970s, when James Anderson first discussed detecting unauthorized access and policy violations in computer systems [2], [3]. Since then, many security solutions, including Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)—have been developed. These systems can be host-based, network-based, or hybrid, and may operate in centralized or distributed environments [4].

IDS methods can also be classified based on how they detect threats. Signature-based systems rely on known attack signatures and are effective for detecting known threats but struggle with new or zero-day attacks. In contrast, anomaly-based systems model normal behavior and detect deviations, which makes them capable of identifying unknown threats—but at the cost of higher false alarm rates.

In recent years, Artificial Intelligence (AI), Machine Learning (ML), Neural Networks (NN), and Deep Learning (DL) have been increasingly applied to intrusion detection. These fields are closely related, with NN and DL being subsets of ML, and ML itself a core part of AI. Among them, neural networks are particularly known for learning patterns from data and adapting to complex threat environments [5].

The CICIDS2017 dataset [40], released by the Canadian Institute for Cybersecurity, is widely used for evaluating network intrusion detection models. This dataset was designed to address shortcomings in earlier benchmarks (like KDD Cup 1999 and NSL-KDD) by providing more realistic traffic and modern attack scenarios. It contains five days of labeled network flow records, comprising 2.83 million flows with more than 80 features each, spanning benign traffic and 14 distinct attack types (e.g., DDoS, web attacks, brute force, infiltration). Because of its comprehensiveness and diversity, CICIDS2017 has become a benchmark in anomaly-based IDS research.

However, CICIDS2017 presents several challenges. The dataset suffers from a severe class imbalance, where some attack types are overrepresented (e.g., DoS), while others appear only a few times (e.g., Heartbleed). This imbalance can severely bias classifiers - models tend to favor the majority classes, achieving high overall accuracy by detecting frequent attacks and abundant regular traffic while failing to detect rare attacks. Therefore, many studies explicitly address data imbalance to improve the detection of minority attack classes.

Despite significant advancements in Network Intrusion Detection Systems (NIDS), several critical issues remain unresolved. Many studies using the CICIDS2017 dataset fail to jointly address class imbalance and feature redundancy—two major factors affecting the accuracy and generalizability of IDS. Existing methods often treat these issues independently, which limits their effectiveness. Moreover, inconsistencies in the dataset and over-reliance on accuracy as a sole metric further hinder robust evaluation. These gaps call for a more integrated and efficient preprocessing strategy to enhance model performance and reliability in anomaly-based detection systems.

This study addresses the above limitations with the following contributions:

- We propose a unified anomaly-based intrusion detection (AIDS) framework that integrates hybrid data balancing with advanced feature selection to improve detection accuracy, reduce bias toward majority classes, and ensure consistent performance across various attack types.
- We evaluate the effectiveness of the proposed framework using a diverse set of machine learning and deep learning models on the CICIDS2017 dataset, employing comprehensive metrics to ensure robust assessment beyond accuracy.

The proposed methods significantly enhance detection accuracy and reduce false alarms, providing practical and actionable insights for network security analysts.

II. LITERATURE REVIEW

Building upon the need for robust IDS solutions, this section explores how existing research utilizes datasets and machine learning techniques to improve detection performance. Datasets play a pivotal role in Machine Learning (ML) for Intrusion Detection Systems (IDS), significantly influencing their performance. Prior research has emphasized the importance of purpose-built datasets designed specifically for IDS applications. While many studies have utilized Network Intrusion Detection Systems (NIDS) coupled with ML algorithms, Deep Learning (DL) techniques are increasingly prominent, demonstrating promising outcomes. This paper concentrates on multiclass anomaly-based NIDS using ML models applied to flow data, reviewing recent advancements in this area.

To train reliable models on benchmark datasets, many researchers emphasize preprocessing as a foundational step. Effective preprocessing is critical when working with CICIDS2017's raw flow data. In most studies, the original packet capture files are processed by CICFlowMeter to extract flow-level features. This yields a rich set of attributes (e.g., packet counts, byte volumes, durations, protocol flags) characterizing each network flow. However, researchers have noted inconsistencies and errors in the provided features. For instance, some features appear duplicated or mislabeled due to evolving versions of the CICFlowMeter tool. Engelen et al. [6] performed an in-depth ''troubleshooting'' of CICIDS2017, uncovering issues like duplicate records, flow timestamp inconsistencies, and labeling errors. Likewise, Panigrahi et al. [7] reported that certain artifacts in the dataset could bias an IDS's detection engine. Recent works have either cleaned the data or re-extracted features from raw traffic to address these. For example, Rosay et al. [8] introduced a corrected feature extraction tool (LycoSTand) to build a refined version of CICIDS2017 (called LYCOS-IDS2017), which improved data quality and downstream IDS performance.

Common preprocessing steps include handling missing values, normalizing numeric features, encoding categorical fields, and removing non-informative attributes. Many studies drop identifiers like flow IDs or timestamps that don't aid Classification. Categorical fields such as protocol or service labels are typically one hot encoded or label-encoded. Feature scaling is often applied to ensure that no single feature dominates the learning process. A popular choice is Z-score normalization, which standardizes features to have zero mean and unit variance. For example, Korium et al. [9] normalized all input features via Z-score to preserve data distribution while handling outliers. This preprocessing enabled their models to train effectively on the mixed-scale network features. They reported over 99.8% accuracy for their IDS when detecting anomalies in the combined dataset (CIC-IDS-2017/2018). Such high numbers are easier in the binary case, as evidenced by works that focus on detecting a single attack category. Some deep learning approaches even bypass manual feature normalization by learning directly from raw bytes or packet sequences; however, in CICIDS2017-based studies, the prevalent practice is to work with the provided flow features after cleaning.

Notably, data splitting is carefully done to evaluate generalization. A typical split is to use a portion of flows for training and hold out another portion for testing (commonly 70-80% train, 20-30% test). Cross-validation is also employed - e.g., 10-fold cross-validation in some experiments to ensure robust performance estimates. In multi-day datasets like CICIDS2017, some researchers ensure that training and testing sets are separated by days to simulate detecting new attacks on future days.

With 80 features per flow in CICIDS2017, several works explore feature selection to reduce dimensionality and focus on the most informative attributes. High dimensionality can lead to longer training times and potential overfitting, so identifying a smaller subset of relevant features is desirable. Panigrahi et. al. [7] conducted one of the earliest detailed feature analyses on CICIDS 2017. They examined the importance of each feature and identified a subset of 36 features that yielded the best classifier performance (in terms of accuracy and F1-score) for their models. This 36-feature optimized subset reportedly improved detection accuracy while discarding redundant attributes. Their study underscored that not all CICIDS2017 features are equally useful - some features (like certain flow byte counts or packet timings) were more predictive of attacks than others.

Techniques like Mutual Information (MI) and Chi-square tests were used to rank features by relevance. Selvakumar et al. [10] used MI to select an optimal feature set for their CNN model, augmenting the input with the most significant features to improve Classification. By choosing features with high mutual information with the class label, their Feature-Augmented CNN achieved better accuracy (95%) than using all features. In general, feature selection is often performed to handle ''the curse of dimensionality'' and improve model interpretability. The consensus is to find a balance: select key features to mitigate noise and complexity while retaining enough information to differentiate all attack types accurately.

Given the skewed distribution of attack categories, data balancing techniques are often applied to address this challenge. Oversampling minority classes is a common solution to handle high class imbalance between different attack categories. The Synthetic Minority Over-Sampling Technique (SMOTE) has been widely used to generate synthetic samples for under-represented attack types. Alfrhan et al. [11] demonstrated that applying SMOTE to CICIDS2017 significantly improves the detection of rare intrusions, compared to leaving the data skewed. In fact, in experiments on CICIDS2017 (as well as SCADA traffic), SMOTE outperformed other strategies like random oversampling and ADASYN in balancing the classes. By interpolating new minority samples, SMOTE helps classifiers receive a more balanced training signal, reducing bias towards majority classes.

Under-sampling the majority (i.e., normal traffic or very frequent attacks) is another approach. Ho et al. [12] evaluated Random Under-Sampling (RUS), SMOTE, and a hybrid of both on the unbalanced CICIDS2017 dataset. They found that RUS yielded the best overall detection performance, managing to boost true positive rates for 12 out of 15 classes (including some hard-to-detect attacks). The intuition is that by trimming down the overwhelming normal traffic, the classifier can focus on learning the patterns of each attack.

However, under-sampling risks discarding useful data; hence combinations of SMOTE + RUS (or SMOTE + Tomek links to remove borderline majority samples) have also been tried. Mbow et al. [13] proposed a hybrid sampling approach: they first applied SMOTE to amplify minority classes, then Tomek link under-sampling to remove ambiguous majority samples.

In addition to modifying the data distribution, researchers have explored algorithm-level solutions. For instance, besides sampling, some studies handle imbalance through the model's learning process itself. Class weight adjustment in the loss function is an effective alternative to resampling. Instead of altering the data distribution, class weighting assigns a higher penalty to mistakes on minority class samples during training. Selvakumar et al. [10] adopted this strategy for their deep ensemble model, computing weights inversely proportional to class frequencies. In their training of the CNN-based classifier, rarer attacks like Heartbleed were given larger weights, forcing the model to pay more attention to correctly classifying those. This method preserved the original dataset but mitigated bias by cost-sensitive learning. The result markedly improved minority-class detection rates - for instance, they reported substantially better recall for the Heartbleed attack when using weighted training. The consensus across the literature is that properly balanced training data (or algorithms that compensate for imbalance) is key to building effective IDS models across all attack types, not just the dominant ones.

The CICIDS2017 dataset [40] natively provides multiple attack categories, allowing multiclass learning. Most recent works leverage this and build multiclass classifiers to identify the specific attack type once an anomaly is detected. Multiclass Classification provides more granular insight (e.g., distinguishing a DDoS from a brute-force login attack). Still, it is more challenging due to the aforementioned class imbalance and the need to differentiate many classes.

Karatas et al. [13] emphasized deep learning's advantages for IDS, including reduced training time, improved accuracy, and Big Data compatibility, while contrasting it with traditional ML approaches.

Zhou et al. [15] evaluated six classifiers for Zero-Day attack detection using flow-based data (CICFlowMeter) and the CIC-AWS-2018 dataset, with decision trees achieving 100% accuracy on novel attacks. Concurrently, Kim et al. [16] proposed a CNN-based IDS, converting CSE-CIC-IDS 2018 dataset features into images, outperforming RNNs in multiclass attack classification but acknowledging the need for model optimization.

Kanimozhi et al. [17] developed an ANN-based IDS with hyper-parameter optimization on the CSE-CIC-IDS2018 dataset, achieving exceptional accuracy and precision for botnet attacks. However, broader applicability to other attack types remains unexplored. Collectively, these works demonstrate evolving methodologies—from hybrid ML architectures to deep learning—and underscore persistent challenges in generalizability, computational efficiency, and comprehensive threat coverage. However, it acknowledges limitations, such as a single attack focus and a CPU-based framework.

Karatas et al. [18] address the challenge of imbalanced and outdated datasets in IDSs, proposing the use of six machine learning algorithms and an up-to-date dataset, CSE-CIC- IDS2018, to enhance detection rates for rare intrusions. The authors leverage the Synthetic Minority Oversampling Technique (SMOTE) to address class imbalance and evaluate six machine learning classifiers—K Nearest Neighbor, Random Forest, Gradient Boosting, Adaboost, Decision Tree, and Linear Discriminant Analysis—demonstrating accuracy improvements ranging from 4.01% to 30.59% compared to recent literature.

Arafat et al. [19] further propose a machine learning-based framework to detect and isolate malicious nodes in wireless networks, specifically targeting denial-of-service (DoS) attacks induced by virtual jamming. Their approach focuses on mitigating adversarial interference by identifying compromised nodes through algorithmic analysis, contributing to enhanced network resilience against coordinated jamming threats. Outperforming non-machine learning methods, the proposed solution enhances network throughput and reduces delay by accurately classifying nodes as malicious or non-malicious, thereby ignoring channel requests from identified jammers.

Gopalan et al. [20] emphasized the critical role of balanced datasets in optimizing machine learning (ML) for intrusion detection systems (IDS), specifically analyzing the CSE-CIC IDS 2017 and 2018 datasets. The study identifies dataset imbalance as a key challenge contributing to algorithmic bias in ML models. It underscores the necessity of addressing this imbalance to avoid skewed classification or prediction outcomes. Additionally, the authors provide a structured taxonomy of ML methodologies applied to these datasets, offering a framework for future research. In a parallel study, Mubarak et al. [21] addresses intrusion detection in SCADA systems, employing ML algorithms to analyze system behavior and detect anomalies. This work highlights the application of ML in securing industrial control systems, focusing on identifying deviations from normal operational patterns to mitigate threats in critical infrastructure environments.

Singh et al. [22] introduce a novel framework using One Class SVM and active learning, demonstrating better performance than other datasets. Lalduhsaka et al. [23] propose an ensemble approach using machine learning algorithms, achieving high classification accuracy of 98.3% and 95.1% on the CICIDS 2017 and CICIDS 2018 datasets, respectively.

Arevalo-Herrera et al. [24] evaluate traditional machine learning techniques such as Decision Trees and Random Forests, finding a high accuracy of 97% for all features and 87% for SDN features. Overall, these papers highlight the effectiveness of machine learning techniques for anomaly detection in the CICIDS 2017 and CICIDS 2018 datasets.

Numerous studies have contributed significantly to the advancement of intrusion detection systems (IDS), particularly those leveraging machine learning and deep learning with the CICIDS2017 dataset [40]. However, a closer examination reveals that many existing approaches address individual challenges—such as feature selection or class imbalance—without integrating them into a unified framework.

For instance, Selvakumar et al. [10] applied mutual information-based feature selection to improve CNN performance. Conversely, Alfrhan et al. [11] focused on using SMOTE to balance minority classes but did not consider the impact of redundant or irrelevant features on model performance. Similarly, Mbow et al. [13] proposed a hybrid sampling technique (SMOTE + Tomek links) but did not explore dimensionality reduction or feature optimization. Our proposed framework differs from these by integrating both a strategic hybrid data balancing method (SMOTE + targeted undersampling) and advanced feature selection using the Extra Trees Classifier as a systematic pipeline, allowing the model to better generalize and detect both frequent and rare attacks effectively.

Furthermore, many previous works, including those by Korium et al. [9] and Lalduhsaka et al. [23], reported high accuracy rates but focused primarily on binary or limited multiclass classification, often using overall accuracy as the main metric. In contrast, our study emphasizes multiclass classification and evaluates model performance using a comprehensive set of metrics, including Precision, Recall, F1-score, ROC-AUC, and MCC. This provides a more nuanced and

robust understanding of model performance, particularly under the natural class imbalance present in real-world intrusion detection scenarios.

Additionally, although Zhou et al. [15] and Kim et al. [16] demonstrated promising results using deep learning models, their approaches did not specifically tackle the combined impact of dataset quality, high-dimensionality, and class imbalance. Our method addresses these limitations by first refining the dataset through systematic feature reduction and balancing, followed by extensive evaluation across both traditional ML and deep learning models.

In summary, while previous studies have made notable progress in IDS development, the novelty of our work lies in the integration of critical preprocessing steps—strategic hybrid resampling and feature selection—into a single framework. This not only enhances detection capability but also ensures fair and reliable performance benchmarking using realistic, imbalanced datasets.

III. BACKGROUND AND METHODOLOGY

In this study, we have used six machine learning strategies, especially for multiclass attack classification. The main objective is to design an anomaly detection-specialized framework for high-speed networks and analyze the performance of heterogeneous machine learning models for network threat detection and prediction. We employ KNN, XGBoost, Decision Tree, LSTM, Transformer models, and SVM.

Given the challenges posed by class-imbalanced datasets—particularly their adverse effect on accurately detecting network attacks—we employed a strategic hybrid resampling strategy that combines both oversampling and undersampling techniques. This approach aims to enhance the model's sensitivity to rare attack types. The technique balances the minority and majority attack classes to eliminate false alarms and unbiasedly ensure the distribution of different attack classes into the model. The model, following oversampling, employed an innovative method for selecting features and chose the 42 most significant features from the 84 features of the dataset. The method greatly enhanced the detection power of the intrusion detection system above the achieved outputs in other literatures [25], [26], [27], [28], [29], [30], and [31].

A. SOFTWARE STACK

Our research used Python (version 3.6) [39] because it's a user-friendly, free, and open-source language. Python's syntax is concise, making it a convenient language to write and read, and it's well-documented and supported by a strong community. We used Scikit-learn (Sklearn) [37], a widely used Python library with a vast collection of robust machine-learning algorithms. Sklearn was documented correctly; therefore, it was appropriate for all operations in our research. For the handling and processing of data, we used Pandas, which effortlessly deals with huge datasets and offers simple filtering, editing, and data manipulation. We used the Matplotlib library [39] to visualize data clearly, which is known for easily creating detailed graphs and charts. We used NumPy [38] for calculations, a Python library for mathematical and logical operations.

B. PLATFORM HARDWARE

All the experiments throughout this work were carried out using Google Colab Pro, a cloud-based computing environment offered by Google. The specifications used throughout the experimental procedure are listed as follows:

- CPU: Intel® Xeon® CPU @ 2.20GHz (or equivalent, as offered by Colab Pro)
- GPU: NVIDIA Tesla T4 or NVIDIA Tesla P100 (depending on availability)
- RAM: Approximately 25 GB (standard for Colab Pro)
- Operating System: Ubuntu 18.04 LTS (Linux-based environment)

This configuration provided stable computer resources to ensure reliable execution and replicability of experiments.

C. METHODOLOGY

In this section, we present the general framework of our proposed anomaly-based intrusion detection system. The proposed framework demonstrated in Fig. 1 gives a clear roadmap to develop a robust anomaly-based IDS using the CICIDS2017 dataset. The workflow is systematically organized into five key stages: data collection, preprocessing, feature selection, data partitioning, and model evaluation within an anomaly-based intrusion detection (AIDS) framework.

The process starts with collecting the CICIDS2017 dataset and then combining it into one complete dataset. This initial step provides consistency in data structure across multiple files and prepares the dataset for the subsequent analysis steps.

Following dataset consolidation, the data preprocessing phase is initiated. This entails the elimination of duplicate or missing records, thus making the data more reliable and of higher integrity. Encoding and scaling processes are followed to transform the categorical variables into numerical ones and normalize the data distribution. These steps ensure machine learning algorithm compatibility and enhance convergence efficiency in the model training procedure.

One of the major improvements in this framework is class imbalance issues are effectively managed through a combination of the Synthetic Minority Oversampling Technique (SMOTE) and strategic under-sampling. SMOTE is particularly applied to minority attack classes to synthetically generate new instances and thus eliminate bias due to rare intrusion activities. Additionally, under-sampling is applied to the majority benign class to enhance the overall dataset balance. These techniques combined enhance the fairness and the classification models' performance.

Following preprocessing and resampling, the data proceeds to the feature selection step, in which the relative importance of each feature is measured using the ExtraTreesClassifier,

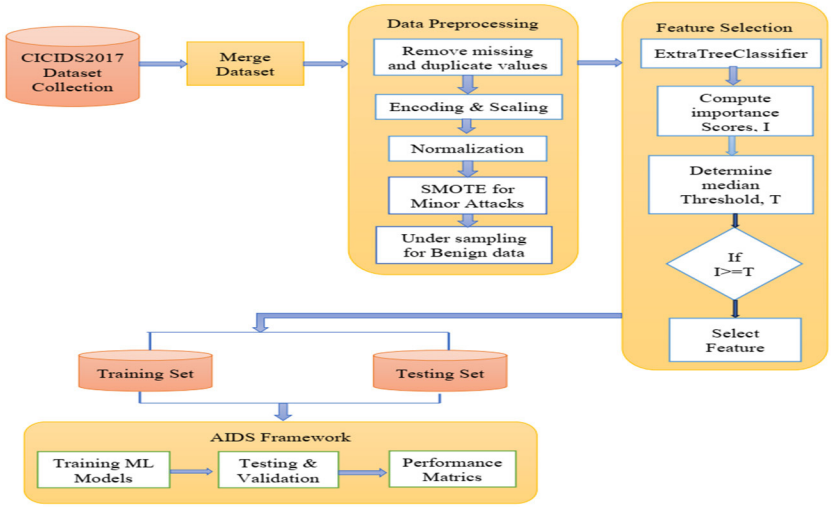

FIGURE 1. Proposed anomaly-based intrusion detection model.

TABLE 1. Class distribution before sampling.

Label	COUNT
BENIGN	2359289
DoS Hulk	231073
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slow loris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack - Brute Force	1507
Web Attack - XSS	652
Infiltration	36
Web Attack - SQL Injection	21
Heartbleed	11

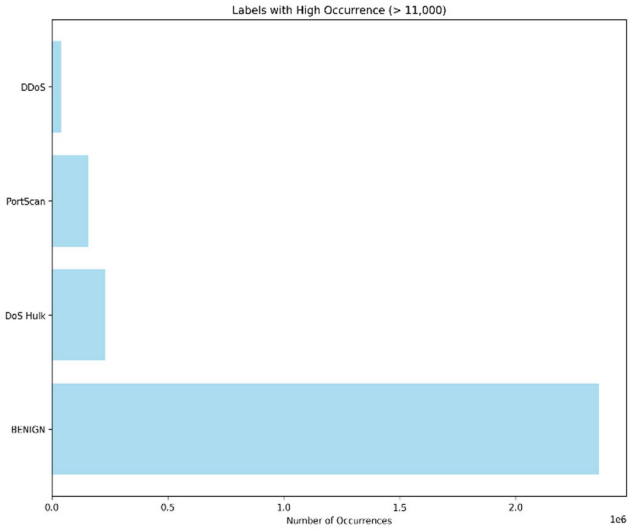

FIGURE 2. High occurrence of traffic categories within the CICIDS2017 dataset.

a tree-based ensemble learning algorithm. Importance scores, I for all features are calculated by the model, and a median threshold, T is determined. Features with importance scores greater than or equal to this threshold (I >= T) are retained for further model building. The target reduction of the feature space seeks to remove irrelevant or less useful features to decrease computational complexity and enhance model generalization.

Following feature selection, the dataset is divided into a training and testing subset. This stratified split guarantees that model evaluation will demonstrate realistic performance expectations with varying data distributions.

Finally, the AIDS framework includes the training of six different machine learning models, validation of performance, and evaluation using traditional performance metrics such as accuracy, precision, recall, and F1-score. All this comprehensive approach helps in determining the best-suited model architecture for effective and reliable network intrusion detection.

1) PREPROCESSING

A comprehensive preprocessing pipeline was implemented to correct for extreme class imbalance inherent in CICIDS2017. Initially, the original dataset contained 2,830,743 records and 85 features. Categorical attributes are removed at an early stage to prevent encoding-related overhead. Subsequently, all numerical features are normalized using the StandardScaler() to ensure uniform scale and reduce model sensitivity to differing feature magnitudes.

Fig.2 - 4 offers a clear depiction of the class distribution within the CICIDS2017 dataset, highlighting the significant

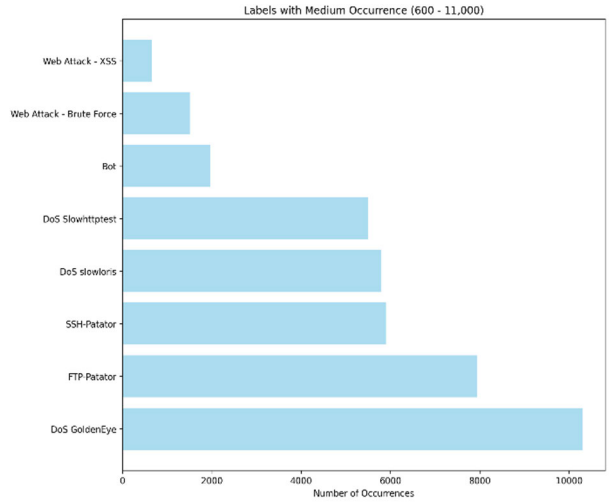

FIGURE 3. Medium occurrence of traffic categories within the CICIDS2017 dataset.

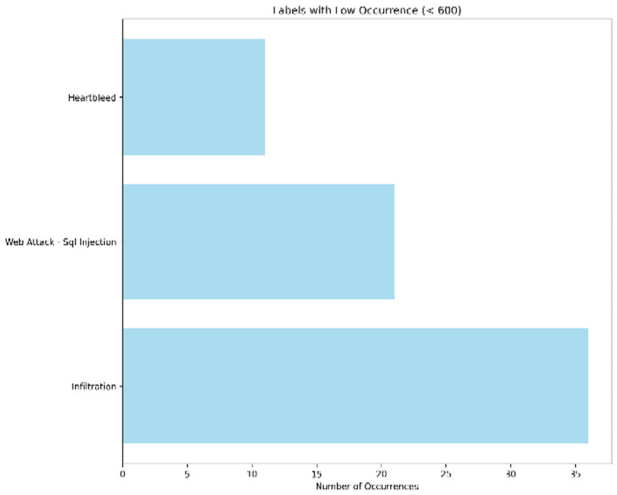

FIGURE 4. Low occurrence of traffic categories within the CICIDS2017 dataset.

imbalance across different network traffic categories. Fig. 2 illustrates labels with high frequency—most notably, the BENIGN class, which overwhelmingly dominates the dataset with over 2.3 million instances. Other frequently occurring attacks include DoS Hulk, PortScan, and DDoS, each with a considerable number of samples, yet still far fewer than the benign class. Fig. 3 focuses on attacks of moderate frequency, such as DoS GoldenEye, FTP-Patator, DoS Slowloris, and SSH-Patator, which range between 600 and 11,000 instances.

These classes, while not rare, are significantly underrepresented compared to the dominant labels and require targeted handling through resampling or boosting techniques to prevent classifier bias.

Fig. 4 visualizes rare attack types with fewer than 600 occurrences, including Heartbleed, Web Attack – SQL Injection, and Infiltration. These underrepresented classes are especially challenging to detect due to the scarcity of training samples and pose a risk of being overlooked by standard classification algorithms. This distribution clearly demonstrates the long-tail problem inherent in intrusion detection datasets, emphasizing the need for advanced strategies—such as Synthetic Minority Oversampling Technique (SMOTE), anomaly detection methods, or class-weighted loss functions—to ensure robust detection performance across both majority and minority classes. Addressing this imbalance is critical for developing an effective and equitable intrusion detection system.

TABLE 2. Attack class distributions before and after sampling.

Label	Before Sampling	After Sampling	Change (%)
BENIGN	2359289	400000	-83.04574
Bot	1966	1966	0
DDoS	41835	41835	0
DoS GoldenEye	10293	10293	0
DoS Hulk	231073	231073	0
DoS Slowhttptest	5499	10000	81.851246
DoS slowloris	5796	10000	72.532781
FTP-Patator	7938	7938	0
Heartbleed	11	2000	18081.81818
Infiltration	36	5000	13788.88889
PortScan	158930	158930	0
SSH-Patator	5897	5897	0
Web Attack - Brute Force	1507	10000	563.570007
Web Attack - SQL Injection	21	3000	14185.71429
Web Attack - XSS	652	8000	1126.993865

2) ATTACK CLASS DISTRIBUTION ADJUSTMENT THROUGH SAMPLING

To improve the intrusion detection models' reliability and fairness, the dataset underwent a careful resampling process aimed at mitigating severe class imbalance. The reliability and fairness of the BENIGN class, which overwhelmingly dominated the dataset with over 2.3 million instances, was significantly reduced to 400,000 samples through undersampling. This substantial 83% reduction was necessary to prevent the model from being biased toward the majority class and overlooking minority attack classes.

Conversely, several underrepresented attack categories were enriched through oversampling techniques. For instance, the Web Attack – SQL Injection class, initially comprising only 21 instances, was expanded to 3,000, marking a remarkable 14,185% increase. Similarly, the Web Attack – XSS and Web Attack – Brute Force classes were increased by over 1,100% and 560%, respectively. These enhancements ensure that rare attack types are sufficiently represented during model training, which is crucial for improving sensitivity to sophisticated or low-frequency threats.

Interestingly, certain classes such as Bot, DDoS, FTP-Patator, and PortScan retained their original sample counts. This indicates that their existing distributions were

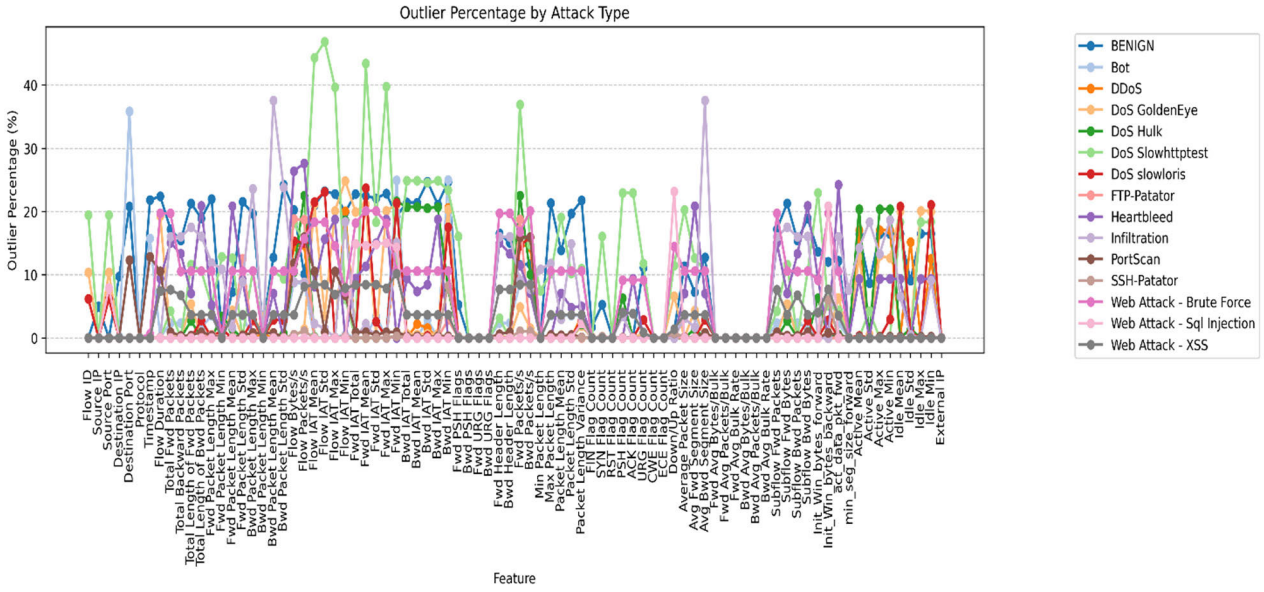

FIGURE 5. Outlier percentage of attack types.

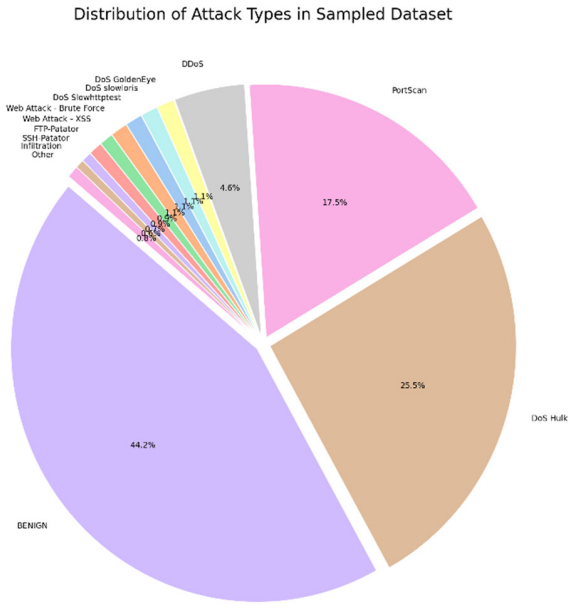

FIGURE 6. Distribution of attack types in sampled dataset Outlier percentage of attack type.

deemed adequate for maintaining a balanced learning process without introducing redundancy or overfitting.

Through a combination of the Synthetic Minority Over-sampling Technique (SMOTE) and controlled under-sampling, the final dataset reflects a more balanced structure ability of the model but also enha , allowing the classifier to learn more effectively across all attack types. This adjustment not only strengthens the generalization ability of the model but also enhances its capability to detect both frequent and rare intrusions with higher accuracy and lower false-negative rates.

To preserve data fidelity, the statistical properties of the original and resampled datasets were compared, focusing on the variation in mean values across numerical features. Features exhibiting more than 5% deviation were flagged to assess the impact of resampling on data quality.

Different attack types influence network traffic in unique ways, which can be captured through statistical outlier analysis. Identifying such patterns assists in feature selection, helps improve anomaly detection performance, and can be integrated into Explainable AI (XAI) frameworks to justify classification decisions. Fig. 5 visualizes the outlier percentage of attack types.

Fig. 6 visualization demonstrates how a class imbalance in the original dataset has been effectively managed through a mix of undersampling (of benign traffic) and oversampling (of underrepresented attack types). The outcome is a more balanced distribution, which improves the fairness, generalization, and detection performance of intrusion detection models trained on this data. Finally, the balanced dataset was saved for downstream machine-learning applications. This preprocessing approach enhances the learning capability of models by ensuring equitable representation across all attack types while maintaining the statistical integrity of the feature space.

3) FEATURE SELECTION

All 85 features in the dataset were identified as numerical, and no categorical variables were present, simplifying the preprocessing pipeline by removing the need for one-hot encoding or label encoding steps. Key numeric features include:

TABLE 3. Top selected features with importance scores.

	Feature	Importance		Feature	Importance
0	Source IP	0.063569	**21**	Fwd IAT Std	0.01724
1	External IP	0.061924	**22**	Down/Up Ratio	0.0168
2	PSH Flag Count	0.049141	**23**	Max Packet Length	0.01586
3	Destination IP	0.04126	**24**	Timestamp	0.01555
4	Flow IAT Max	0.037495	**25**	Average Packet Size	0.0151
5	min_seg_size_forward	0.032293	**26**	Min Packet Length	0.01494
6	Flow ID	0.030326	**27**	Packet Length Std	0.01398
7	ACK Flag Count	0.030316	**28**	Fwd Packets/s	0.01269
8	Source Port	0.029594	**29**	Packet Length Mean	0.01254
9	Protocol	0.029111	**30**	Flow IAT Std	0.01205
10	Bwd Packet Length Mean	0.028025	**31**	URG Flag Count	0.01143
11	Bwd Packet Length Max	0.025744	**32**	FIN Flag Count	0.01086
12	Idle Min	0.025115	**33**	Fwd Packet Length Min	0.00858
13	Init_Win_bytes_forward	0.023844	**34**	Subflow Fwd Packets	0.00854
14	Fwd IAT Max	0.022994	**35**	Bwd IAT Max	0.00852
15	Idle Mean	0.022656	**36**	Packet Length Variance	0.00849
16	Idle Max	0.022476	**37**	Fwd IAT Mean	0.00799
17	Avg Bwd Segment Size	0.021563	**38**	Flow Duration	0.00787
18	Destination Port	0.020985	**39**	Fwd IAT Total	0.00706
19	Bwd Packet Length Min	0.020882	**40**	Bwd IAT Std	0.00672
20	Bwd Packet Length Std	0.017658	**41**	Flow IAT Mean	0.00637

- Traffic flow characteristics (e.g., Flow Duration, Flow Bytes/s)
- Packet-level metrics (e.g., Fwd Packet Length Mean, Packet Length Variance)
- Protocol behavior (e.g., SYN Flag Count, ACK Flag Count)
- Session state indicators (e.g., Idle Mean, Active Std)
- Network header and protocol information (e.g., Protocol, Source Port, Destination IP)

After cleaning and standardizing the numeric data, the preprocessing pipeline transformed the dataset into a final shape of (2830743, 84), indicating that one non-informative or duplicate feature was removed. A feature selection process was then applied to retain only the most informative features.

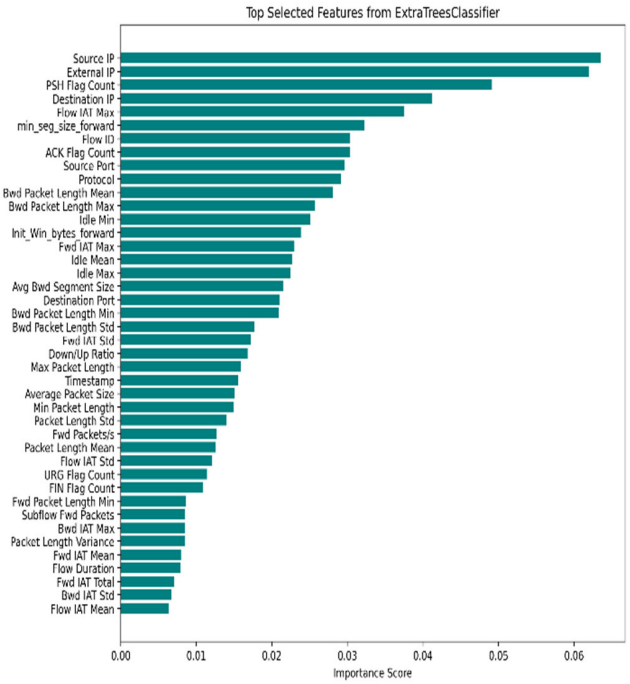

FIGURE 7. Top selected features from ExtraTreesClassifier.

To assess the relative importance of each feature, the Extra Trees Classifier (ETC) from the Scikit-learn library is employed. This ensemble method constructs a collection of randomized decision trees, each contributing to a cumulative estimation of feature significance. The model was initialized with 50 estimators and a fixed random state of 42 to ensure reproducibility. The model was then fitted on the processed feature set Preprocessed label x along with the corresponding target labels y. This configuration was chosen to balance computational efficiency with classification performance, leveraging the ensemble nature of Extra Trees to improve generalization and robustness in intrusion detection tasks.

After computing the feature importance using the trained Extra Trees Classifier, the SelectFromModel class was employed to perform dimensionality reduction. Specifically, features with importance values greater than or equal to the median were retained. This selection strategy offers a balanced trade-off between reducing the dimensionality of the dataset and preserving the most informative features. The transformation was applied to the preprocessed feature set label x, resulting in a refined dataset that serves as input for subsequent model training and evaluation.

By choosing a median cut-off, the process averts arbitrary removal and ensures inclusion of at least half of the most informative features. The most informative selected features, as well as their importance scores, are shown in table 3.

Algorithm 1 Feature Selection Using Extra Trees Classifier

Input: Preprocessed dataset D', Target variable Y
Output: Selected feature set F
1: Initialize selected feature set F $\leftarrow \emptyset$
2: Train ExtraTreesClassifier model M on D′ and Y
3: Compute importance scores I = $\{i_1, i_2, \ldots, i_n\}$ for all features in D′ using model M
4: Determine median importance threshold T $\leftarrow$ median(I)
5: for each feature f_i in D′ do
6: if importance score $i_i \geq$ T then
7: F $\leftarrow$ F $\cup$ $\{f_i\}$
8: end if
9: end for
10: return Selected features F

The most crucial step is to extract the final selected features by indexing their original feature names, which ensures transparency in the model pipeline. This step is essential for explainability and allows us to refer to chosen features meaningfully in further modeling and analysis phases.

These selected features include metrics that significantly contribute to detecting anomalies in network traffic, such as Fwd Packets/s, Packet Length Std, ACK Flag Count, Avg Bwd Segment Size, and several inter-arrival time metrics. Fig. 7 visualizes the most important 42 features used in our model evaluation.

This feature reduction helps mitigate overfitting, reduce computational overhead, and enhance the interpretability of the model.

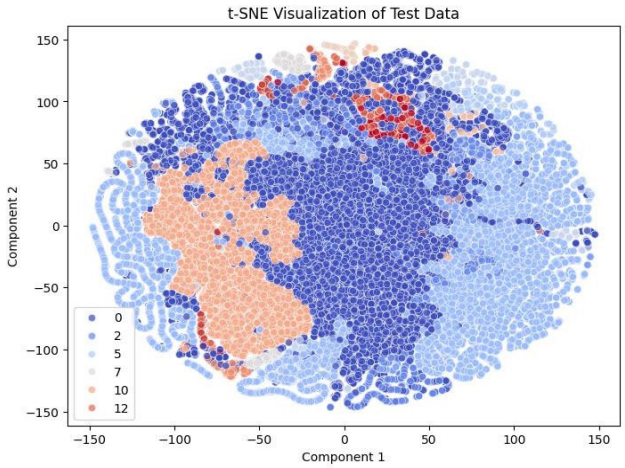

FIGURE 8. t-SNE visualization of test data.

4) SPLIT TRAIN AND TEST DATA

The CICIDS2017 dataset [40] used in this study does not provide predefined training and testing splits; instead, it consists of a single consolidated dataset. This dataset was split into 70% training and 30% test subsets to enable proper model evaluation.

A visual t-distributed Stochastic Neighbor Embedding (t-SNE) was constructed to visualize different attack classes and benign traffic separability and their distribution in the test dataset. t-SNE is a nonlinear dimensionality reduction method that projects high-dimensional data into a lower dimensional (usually 2D) in a manner that preserves the local geometry of the information. This makes it particularly suitable for visually assessing complex datasets' class overlap and clustering tendencies.

As shown in Fig. 8, each point represents an individual sample from the test dataset, and the colors denote different class labels—each corresponding to a specific type of network traffic, including benign and various attack categories.

The two axes, Component 1 and Component 2 are abstract representations derived from the original feature space and do not correspond to any specific feature. Instead, they reflect the relative positions of data points based on their similarity in the original multidimensional space.

A visual t-distributed Stochastic Neighbor Embedding (t-SNE) was constructed to visualize different attack classes and benign traffic separability and their distribution in the test dataset. t-SNE is a nonlinear dimensionality reduction method that projects high-dimensional data into a lower dimensional (usually 2D) in a manner that preserves the local geometry of the information. This makes it particularly suitable for visually assessing complex datasets' class overlap and clustering tendencies.

As shown in Fig. 8, each point represents an individual sample from the test dataset, and the colors denote different class labels—each corresponding to a specific type of network traffic, including benign and various attack categories. The two axes, Component 1 and Component 2, are abstract representations derived from the original feature space and do not correspond to any specific feature. Instead, they reflect the relative positions of data points based on their similarity in the original multidimensional space.

The t-SNE visualization reveals several essential insights: Distinct Clustering of Classes: Some classes, especially the benign traffic, and a few prominent attack types, exhibit well-defined clusters, suggesting that the features extracted by the preprocessing pipeline effectively capture patterns that differentiate them. This supports the suitability of the chosen features and resampling strategy.

Overlap Between Classes: There is a certain extent of overlap among a couple of classes, particularly among classes with nuanced behavioral patterns, i.e., low-volume or Web-based attacks. The overlap can lead to vagueness in labeling and false positives or false negatives in real-time detection.

Imbalanced density: denser regions in the plot represent more occurring classes, i.e., DoS-based or benign attacks, and sparse regions represent minority classes, i.e., infiltration or rare web-based attacks that have been synthetically balanced in preprocessing.

Cumulatively, we can see from this t-SNE plot that feature-engineered data still holds a helpful structure, with class-specific patterns remaining in place. It's a testament

to the feature engineering and resampling practices adopted. It presents a strong visual argument for why the classifier can distinguish selectively normal from anomalous traffic.

5) MODEL ARCHITECTURES AND TRAINING CONFIGURATION

This section provides a comprehensive overview of the architecture and training setup of this study's six machine learning models for network intrusion detection. These include both traditional classifiers and deep learning architectures. Each model is configured using best practices relevant to its type, with consistent input features derived from the CICIDS2017 dataset. Evaluation is done on a hold-out test set, ensuring the model's performance reflects generalization ability.

a: INPUT AND OUTPUT DESIGN

The input to each model consists of a structured feature set extracted from network traffic flows. Specifically, 42 features were selected from the raw dataset, including statistical indicators such as Flow Duration, Packet Length Mean, Fwd IAT Total, and several TCP flag-based fields. These features capture both temporal and behavioral characteristics of network traffic.

The target variable represents one of the 15 traffic categories for classification tasks, including regular and various attack classes (e.g., DoS Hulk, PortScan, Web Attack Brute Force). After preprocessing, categorical labels were encoded into integer form using LabelEncoder. The models output a single integer class label as the prediction, which is subsequently mapped back to its corresponding attack category for evaluation.

b: LONG SHORT-TERM MEMORY (LSTM)

The LSTM model is a recurrent neural network designed to capture temporal dependencies within sequential data. The architecture consists of:

- An input layer receives 42 continuous features.
- Two stacked LSTM layers with 128 hidden units.
- A fully connected (dense) output layer with implicit activation (uses tanh and sigmoid) for multiclass Classification.

Training Configuration:

- Loss Function: Cross-Entropy Loss
- Optimizer: Adam
- Learning Rate: 0.001
- Epochs: 50 full training iterations over the dataset
- Batch Size: 32 samples per batch

c: TRANSFORMER MODEL

The Transformer-based neural network has been adapted for tabular data classification. Unlike its traditional use in sequence modeling, we restructured the Transformer to process fixed-length, high-dimensional feature vectors extracted from network flow records. Key components of architecture:

- A fully connected linear layer maps the original input feature vector (input_dim = 43) to a higher-dimensional latent space (d_model = 64), enabling rich representation learning before attention is applied.
- Composed of 2 stacked encoder layers with:
  - Multi-head self-attention mechanism (nhead = 4) to capture inter-feature dependencies,
  - Feedforward neural subnetwork (dim_feedforward = 128) with ReLU activations,
  - Dropout (dropout = 0.1) to prevent overfitting.
- A fully connected linear layer maps the output of the transformer encoder to the final prediction over num_classes, supporting multiclass Classification.
- The model outputs logits which are converted to class probabilities via softmax during evaluation.
- Loss Function: Cross-Entropy Loss
- Optimizer: Adam
- Learning Rate: 0.001
- Epochs: 50 full training passes
- Batch Size: 32 samples per batch
- Attention Dropout: 0.1

d: K-NEAREST NEIGHBORS (KNN)

The KNN model is a non-parametric classifier that labels each test instance based on a majority vote of its k=5 nearest neighbors in the training space. No explicit training occurs beyond storing the training data. Key elements of architecture:

- Base Algorithm: Distance-based majority voting
- Number of Neighbors: 5
- Distance Metric: Euclidean
- Weight: Uniform

Due to the memory-intensive nature of KNN, dimensionality reduction via feature selection was applied beforehand.

e: EXTREME GRADIENT BOOSTING (XGBOOST)

XGBoost is a gradient boosting ensemble method that constructs decision trees sequentially to correct errors from previous iterations. Its architecture consists of hundreds of shallow trees aggregated to produce robust predictions. Key components of architecture:

- Base Learner: Decision Trees (Classification and Regression Trees – CART)
- Boosting Strategy: Additive gradient boosting
- Objective Function: Multiclass softmax with mlogloss as the evaluation metric
- Regularization: Built-in L1 and L2 regularization to prevent overfitting
- Label Encoding: Target labels encoded using Label Encoder to support categorical class values.

f: DECISION TREE (DT)

The Decision Tree classifier is a univariate tree structure where internal nodes split based on the most discriminative

feature. It constructs a tree using information gain as the splitting criterion. Key components of architecture:

- Criterion: Gini Index
- Maximum Depth: Unrestricted (default)
- Minimum Samples per Split: 2
- Random State: 42 (for reproducibility)

g: SUPPORT VECTOR MACHINE

The SVM model is implemented using SGDClassifier with a hinge loss, simulating a linear SVM optimized through stochastic gradient descent. The key component of the architecture:

- Loss Function: Hinge (for linear SVM)
- Regularization: L2 penalty
- Alpha (learning rate decay): 0.001
- Max Iterations: Determined automatically based on convergence
- Implementation: scikit-learn (SGDClassifier)

IV. EVALUATION

We utilized four key performance metrics to assess our intrusion detection system (IDS) performance: Accuracy, Precision, Recall, and F1-Score. These metrics provide a good measure of the model's predictive power, especially when working with imbalanced datasets, which are common in network security.

A. CONFUSION MATRIX

The four essential components of the confusion matrix:

•**True Positive (TP):** Instances where the model correctly predicts an attack.

•**True Negative (TN):** Instances where the model correctly predicts normal (benign) activity.

•**False Positive (FP):** Instances where the model incorrectly predicts an attack on normal activity (Type I error).

•**False Negative (FN):** Instances where the model fails to detect an actual attack, predicting it as normal (Type II error).

B. ACCURACY

Accuracy calculates the proportion of total correct predictions (true positives as well as true negatives) to the total cases examined.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

While accuracy provides a general sense of performance, accuracy can be misleading in imbalanced datasets where a class greatly outnumbers any other class.

C. PRECISION

Precision assesses the ability of the model to identify actual positive cases (attacks) among all cases that it labeled as positive.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

High accuracy means low false positives, which is very crucial to reducing unnecessary alarms in an IDS.

D. RECALL (SENSITIVITY)

Recall tests the ability of a model to find all actual positive instances (attacks).

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

High recall ensures most of the attack episodes are captured by IDS, with minimal threats evading detection.

E. F1 SCORE

The F1-Score represents a harmonic means of precision and recall, which provides a balanced measure of the two.

$$\text{F1}-\text{Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

This measure is most useful where a balance must be found between precision and recall, as in cases where false positives and false negatives have a high cost.

F. AUC-ROC

The micro-averaged receiver operating characteristic (ROC) curve is a graph of the true positive rate (TPR) over the false positive rate (FPR) over a range of thresholds. The area under the curve (AUC) provides a general measure of the discriminatory ability afforded by a classifier. An AUC value equal to 1.0 indicates perfect classification performance and a value of 0.5 indicates no more than random chance.

G. PRECISION-RECALL

A precision-recall (PR) graph is beneficial for imbalanced multiclass Classification where other performance measures, e.g., accuracy, would be misleading. In the graph, the X-axis is Recall (True Positive Rate), i.e., actual intrusions detected correctly and the Y-axis is Precision, i.e., detected intrusions that proved true. In the graph, every line corresponds to the precision-recall trade-off of a model at various thresholds. The area under the PR graph (average precision), the better the performance, the closer the value will be to 1.

Analyzing these metrics gives us a nuanced understanding of our IDS's performance, enabling us to fine-tune the model for optimal detection capabilities while minimizing false alarms.

V. EXPERIMENTAL RESULTS

We have implemented six models KNN, XGBoost, Decision Tree, LSTM, Transformer models, and SVM, using scikit learn [37] Python's library. Classification reports, confusion matrices, precision-recall curves, and ROC-AUC curves have been widely utilized to evaluate model performance comprehensively.

TABLE 4. XGBOOST classification report.

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
0	1.00	1.00	1.00	120000
1	0.99	1.00	1.00	590
2	1.00	1.00	1.00	12551
3	1.00	1.00	1.00	3088
4	1.00	1.00	1.00	69322
5	1.00	1.00	1.00	3000
6	1.00	1.00	1.00	3000
7	1.00	1.00	1.00	2381
8	1.00	1.00	1.00	600
9	1.00	1.00	1.00	1500
10	1.00	1.00	1.00	47679
11	1.00	1.00	1.00	1769
12	1.00	1.00	1.00	3000
13	1.00	1.00	1.00	900
14	1.00	1.00	1.00	2400
accuracy			1.00	271780
macro avg	1.00	1.00	1.00	271780
weighted avg	1.00	1.00	1.00	271780

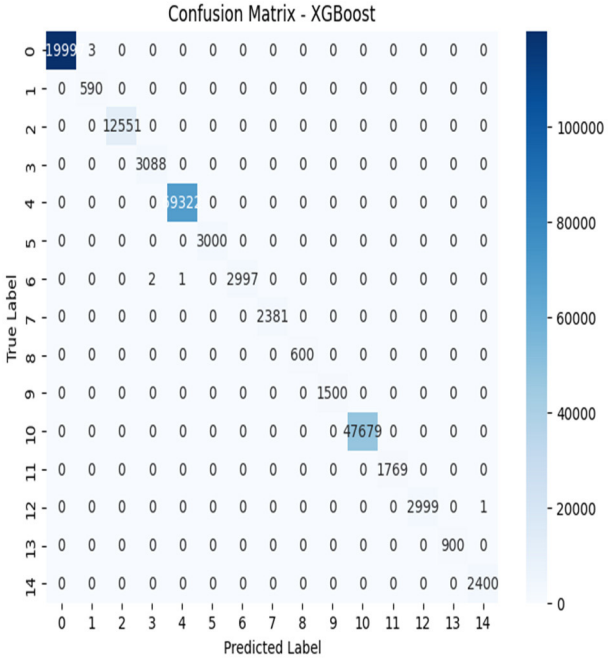

FIGURE 9. Confusion Matrix of XGBoost.

A. MULTICLASS CLASSIFICATION

The analysis was done by assessing classification reports created for all the six machine learning models given for evaluation on the network intrusion detection dataset.

Both Decision Tree and XGBoost models achieved an overall accuracy rate of 100%, along with macro and weighted average F1-scores of 1.00 for all 15 classes. These findings jointly highlight their outstanding ability to classify all network traffic types, including samples belonging to minority classes, without any misclassifications being made. The classification report and confusion matrix of XGBoost and Decision Tree are shown in Tables 4, 5, and Fig. 9 and 10, respectively.

TABLE 5. Decision tree classification report.

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
0	1.00	1.00	1.00	120000
1	0.99	1.00	0.99	590
2	1.00	1.00	1.00	12551
3	1.00	1.00	1.00	3088
4	1.00	1.00	1.00	69322
5	1.00	1.00	1.00	3000
6	1.00	1.00	1.00	3000
7	1.00	1.00	1.00	2381
8	1.00	1.00	1.00	600
9	1.00	1.00	1.00	1500
10	1.00	1.00	1.00	47679
11	1.00	1.00	1.00	1769
12	1.00	1.00	1.00	3000
13	1.00	1.00	1.00	900
14	1.00	1.00	1.00	2400
accuracy			1.00	271780
macro avg	1.00	1.00	1.00	271780
weighted avg	1.00	1.00	1.00	271780

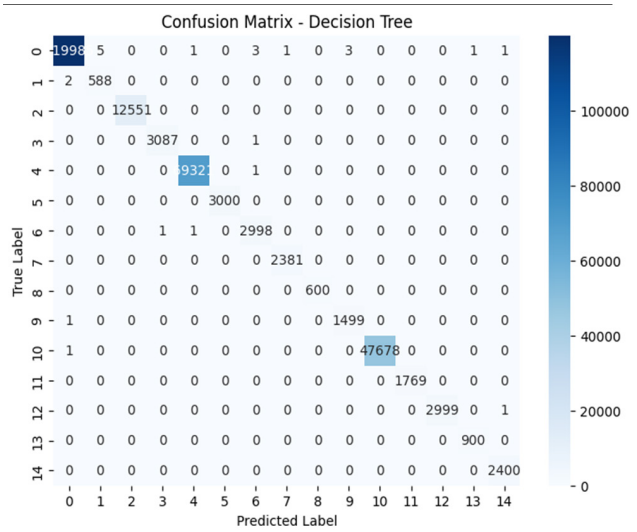

FIGURE 10. Confusion matrix of decision tree.

The XGBoost confusion matrix in Fig. 9 shows classification performance obtained from the Extreme Gradient Boosting (XGBoost) classifier for network intrusion detection processes. The confusion matrix gives a performance report on XGBoost classification by comparing the classes predicted by the classifier and actual labels. The actual class (True Label) is indicated by rows, and the predicted class (Predicted Label) by columns.

The matrix is inspected and a readily observable line pattern diagonally reflecting the high percentage of correct Classification for most intrusion classes is observed. The strong diagonal trend confirms the superior performance of the XGBoost approach in achieving good discrimination between a considerable number of network intrusion classes at highly correct percentages. Almost all diagonal entries reflect the sparse misclassifications and confirm the strength and validity of the approach.

TABLE 6. K nearest neighbor classification report.

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.99	0.97	0.98	120000
1	0.98	0.94	0.96	590
2	0.83	0.95	0.89	12551
3	0.99	0.98	0.99	3088
4	0.99	1.00	0.99	69322
5	0.99	0.99	0.99	3000
6	0.98	0.99	0.98	3000
7	0.98	0.99	0.99	2381
8	1.00	1.00	1.00	600
9	0.97	0.99	0.98	1500
10	0.99	1.00	0.99	47679
11	0.97	0.99	0.98	1769
12	0.94	0.94	0.94	3000
13	0.98	0.99	0.99	900
14	0.92	0.93	0.93	2400
accuracy			0.98	271780
macro avg	0.97	0.98	0.97	271780
weighted avg	0.98	0.98	0.98	271780

TABLE 7. Support vector machine classification report.

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.84	0.81	0.83	120000
1	0.00	0.00	0.00	590
2	0.50	0.80	0.62	12551
3	0.87	0.77	0.81	3088
4	0.81	0.78	0.79	69322
5	0.91	0.69	0.79	3000
6	0.41	0.50	0.45	3000
7	0.00	0.00	0.00	2381
8	0.88	0.80	0.83	600
9	0.00	0.00	0.00	1500
10	0.86	0.91	0.88	47679
11	0.18	0.00	0.01	1769
12	0.00	0.00	0.00	3000
13	0.00	0.00	0.00	900
14	0.36	0.93	0.52	2400
accuracy			0.78	271780
macro avg	0.44	0.47	0.43	271780
weighted avg	0.78	0.78	0.78	271780

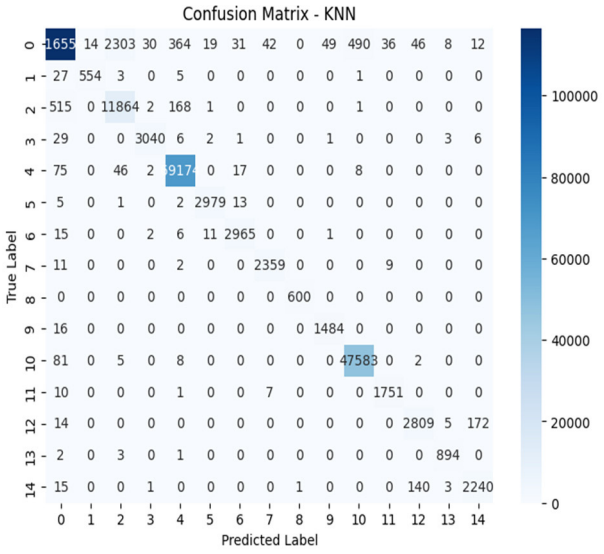

FIGURE 11. Confusion matrix of KNN.

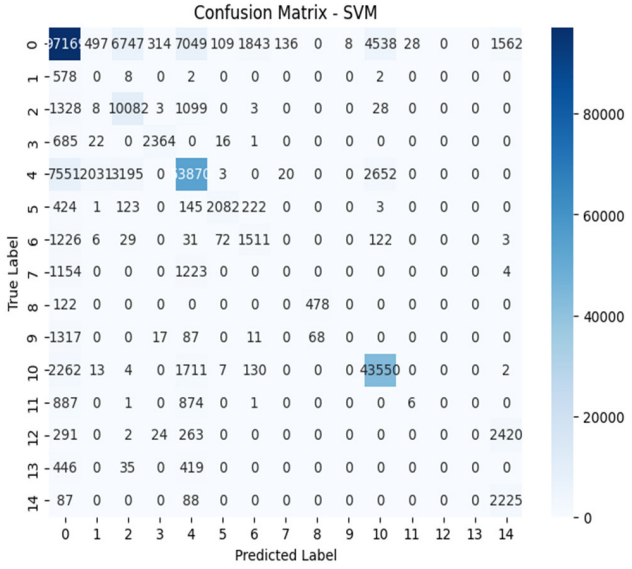

FIGURE 12. Confusion matrix of SVM.

Still, minor misclassifications exist; for example, class 0 is mistakenly classified as class 1, reflecting a bit of ambiguity or overlap between the two classes. Despite those small misclassifications, performance, as reflected in the confusion matrix, generally remains high and shows the strength of XGBoost for accurate and reliable intrusion detection in complex, multiclass network environments.

The K-Nearest Neighbor algorithm achieved 98% overall accuracy, supplemented by a 0.97 macro F1-score. It performed well in most classes but showed a slight decrease in accuracy in some minority classes, namely Class 2 and Class 14. Despite these subtle differences, KNN is still considered a reliable and effective model for intrusion detection. Table 6 represents the classification report of KNN. Fig. 11 below displays the confusion matrix for the K-Nearest Neighbor (KNN) model for the network intrusion detection problem. Rows represent the actual classes (True Label), while columns represent the predicted classes (Predicted Label). Diagonal elements in the matrix represent correctly classified cases and true positive predictions, while off-diagonal elements represent misclassifications. The high density along the diagonal reflects the KNN model correctly predicting most classes, representing high classification accuracy. Still, some classes have high misclassifications. For example, class 0 reflects relatively higher misclassification rates into other classes, implying probable overlap or similarities among features defining the said intrusion types and others.

The Support Vector Machine (SVM) achieved an overall accuracy rate of 78% and a macro F1-score of 0.43. While it accurately classified the majority classes, it was completely ineffective at identifying the minority classes, namely Classes 1, 7, 9, and 13, which makes it less suitable for use in its current form. The classification report of SVM is shown in Table 7. The SVM classifier's confusion matrix

TABLE 8. Long short-term memory classification report.

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.90	0.88	0.89	120000
1	0.57	0.42	0.48	590
2	0.65	0.45	0.53	12551
3	0.93	0.84	0.88	3088
4	0.84	0.97	0.90	69322
5	0.79	0.63	0.70	3000
6	0.66	0.64	0.65	3000
7	0.79	0.79	0.79	2381
8	0.72	0.98	0.83	600
9	0.58	0.49	0.53	1500
10	0.94	0.99	0.96	47679
11	0.87	0.75	0.80	1769
12	0.78	0.08	0.15	3000
13	0.66	0.83	0.73	900
14	0.84	0.04	0.07	2400
accuracy			0.88	271780
macro avg	0.77	0.65	0.66	271780
weighted avg	0.87	0.88	0.87	271780

TABLE 9. Transformer model classification report.

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
0	0.99	0.97	0.98	120000
1	0.90	0.92	0.91	590
2	0.81	0.96	0.88	12551
3	0.97	0.94	0.95	3088
4	0.99	1.00	0.99	69322
5	0.90	0.87	0.88	3000
6	0.97	0.74	0.84	3000
7	1.00	0.99	0.99	2381
8	0.99	1.00	1.00	600
9	0.93	0.97	0.95	1500
10	1.00	1.00	1.00	47679
11	0.99	0.99	0.99	1769
12	0.56	1.00	0.72	3000
13	0.99	0.96	0.97	900
14	1.00	0.03	0.06	2400
accuracy			0.97	271780
macro avg	0.93	0.89	0.87	271780
weighted avg	0.97	0.97	0.97	271780

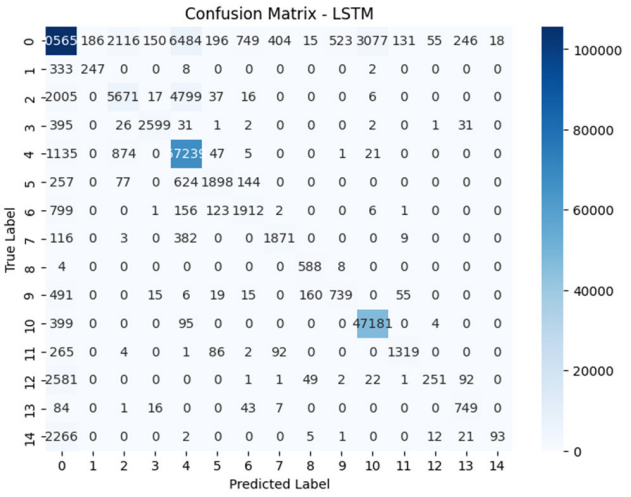

FIGURE 13. Confusion matrix of LSTM.

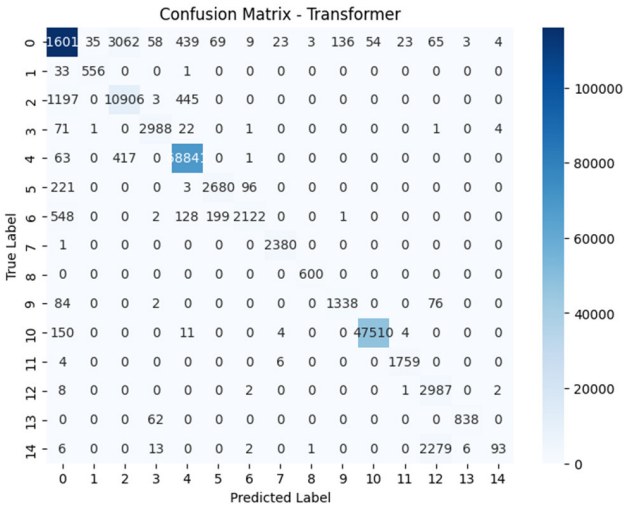

FIGURE 14. Confusion matrix of transformer.

in Fig. 12 reveals many misclassifications due to numerous off-diagonal elements. The trend indicates SVM's failure to differentiate between several intrusion classes effectively may result from overlapping distributions of features or a lack of discrimination capacity. The spread of predictions between classes indicates a lot of challenges in the clear classification of some attacks. It suggests the incapability of the SVM model to handle complex multiclass intrusion datasets effectively.

The performance of the LSTM model is good on prominent classes like class 0 (Benign), class 4 (DoS Hulk), and class 10 (PortScan) with F1-scores ranging from 0.89 to 0.96 indicative of the model picking up patterns across heavily represented attacks. Lower performance across sparse attacks like Class 1 (F1 = 0.48) and Class 2 (F1 = 0.53) is indicative of the inability to detect the less frequent attacks being a problem, Class 12 (F1 = 0.15) and Class 14 (F1 = 0.07), both with very low recall and indicative of picking up effectively none of the instances—presumably as a result of the issue of class imbalance and feature overlap. 3, 7, 11, and 13 are medium-performance cases with F1 scores ranging from 0.73 to 0.88. Table 8 shows the classification report of LSTM.

The confusion matrix of the LSTM model in Fig. 13 has numerous misclassifications spread in multiple classes, as indicated by extensive off-diagonal elements. This shows that the LSTM model has a lot of trouble classifying some types of intrusions properly because some actual classes tend to be poorly predicted by a large set of incorrect ones. Class 0 has a lot of misclassifications, pointing towards complexity or ambiguity of features and trouble for LSTM in recognizing unique temporal patterns. These results call for model adjustment or refinement through tuning or optimization of features or hyperparameters.

The classification report in Table 9 justifies using the Transformer model for multiclass intrusion detection in the case of imbalanced and high-speed networks. Benign traffic (Class 0) is adequately classified with a high precision value (0.98) and high recall value (0.97), justifying the capability of

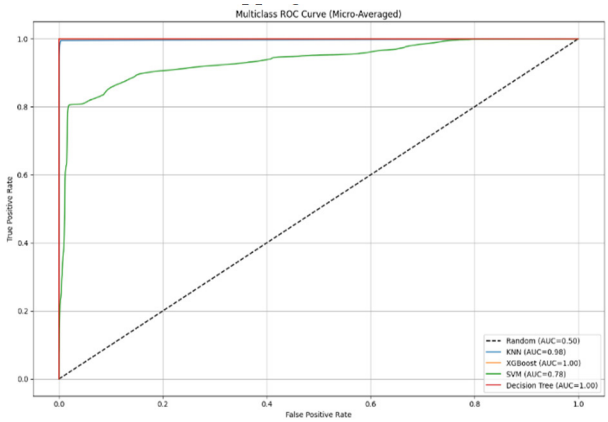

FIGURE 15. AUC-ROC curve for KNN, XGBoost, SVM, and Decision tree.

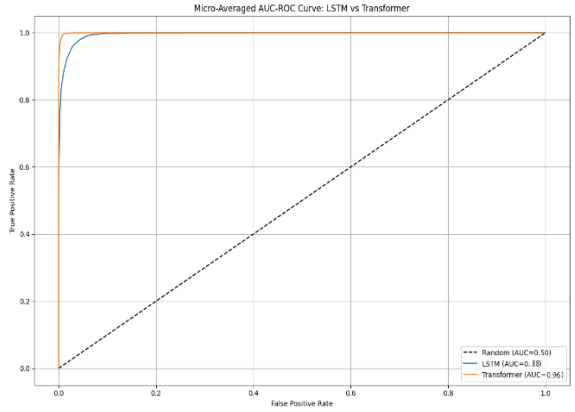

FIGURE 16. AUC-ROC curve for DL models LSTM and transformer.

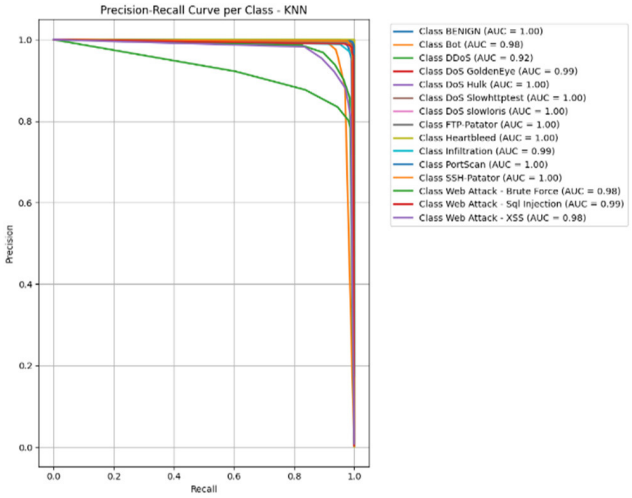

FIGURE 17. Precision-recall curve of KNN.

the model towards detection of normal behavior. High detection is discovered in Web Attacks (Class 4 with 0.99 F1) and Infiltration or Heartbleed-based ones (Class 10 and 11 with F1-scores 1.00 and 0.99, respectively). New and novel attack types, such as Class 14, possess a low value of recall (0.04) but a precision value of 0.90, proving the model does not classify the class correctly due to the potential imbalance in the respective class or feature similarity with benign traffic.

Counterintuitively enough, however, Class 12 shows a precision value of 0.56 but a value of 0.87 for the recall, reflecting that the model identifies all the true positives but misclassifies numerous negatives as positives too. The Transformer model confusion matrix in Fig. 14 shows noticeably better classification accuracy and a stronger trend down the main diagonal and smaller off-diagonal misclassifications compared to LSTM. While misclassifications do exist, their quantities are far smaller and indicate the Transformer's strength in modeling long-range dependencies and in separating well between adjacent classes of network intrusions. While a few of the classes, especially Class 0, still experience misclassification, the performance of the Transformer shows a superior capacity for extracting and making sense of meaningful sequential patterns from intrusion data.

Based on these findings, XGBoost, the decision tree, and transformer models are good candidates for network intrusion detection due to their high performance, accuracy, and stability levels. SVM and LSTM require optimization, improving data preprocessing, and better feature engineering to be as efficient as real-life intrusion detection systems.

B. RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE ANALYSIS

The performance of the proposed models was also verified using the micro-averaged receiver operating characteristic (ROC) curve, a graph of the true positive rate (TPR) over the false positive rate (FPR) over a range of thresholds. The area under the curve (AUC) provides a general measure of the discriminatory ability afforded by a classifier. An AUC value equal to 1.0 indicates perfect classification performance and a value of 0.5 indicates no more than random chance.

Fig. 15 shows the ROC graph comparing the K-Nearest Neighbor (KNN), XGBoost, Support Vector Machine (SVM), and the Decision Tree classifiers. The XGBoost and the Decision Tree models achieved 1.00 perfect AUC values, reflecting a high ability to differentiate between regular and attack traffic. The KNN model achieved 0.98, reflecting excellent ability towards differentiating benign and malicious traffic classes. The SVM model achieved a lower AUC value of 0.78, reflecting relatively inferior performance, particularly in dealing with the problem of class imbalance or overlapping border instances.

Fig. 16 displays performance as a Receiver Operator Characteristic (ROC) curve comparing the performance of two deep architectures—Transformer and LSTM—for multiclass network intrusion detection. The Transformer model can achieve a micro-averaged AUC value of 0.96 and can differentiate benign and malicious traffic classes. Its line rises sharply and hugs the upper-left corner of the graph closely, reflecting high sensitivity and low false favorable rates. The

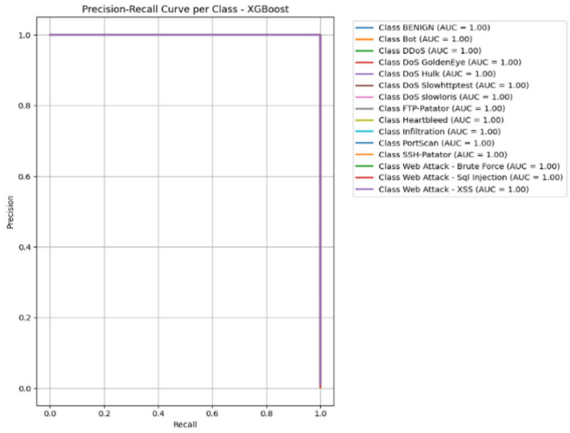

FIGURE 18. Precision-recall curve of XGBoost.

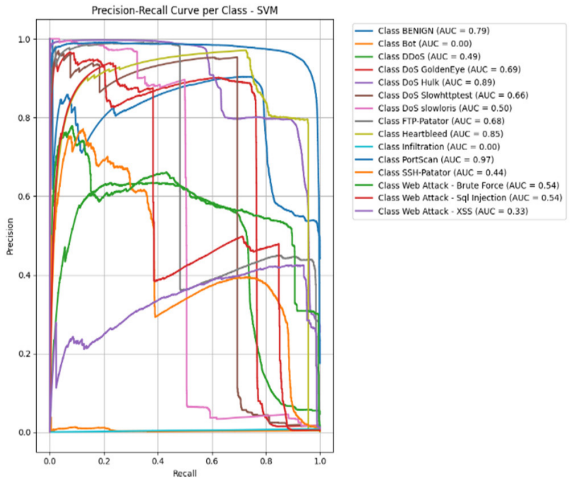

FIGURE 19. Precision-recall curve of SVM.

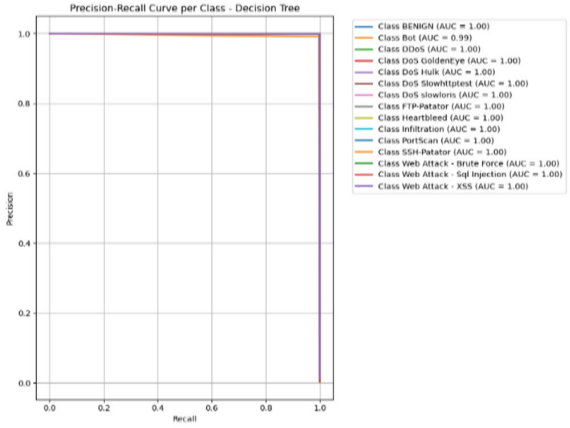

FIGURE 20. Precision-recall curve of decision tree.

LSTM model possesses a lower value of AUC of 0.88 and reflects a relatively weaker but still good level of performance classification. Its line is steep but crosses the top-left axis at a lower value than the line of the Transformer and reflects

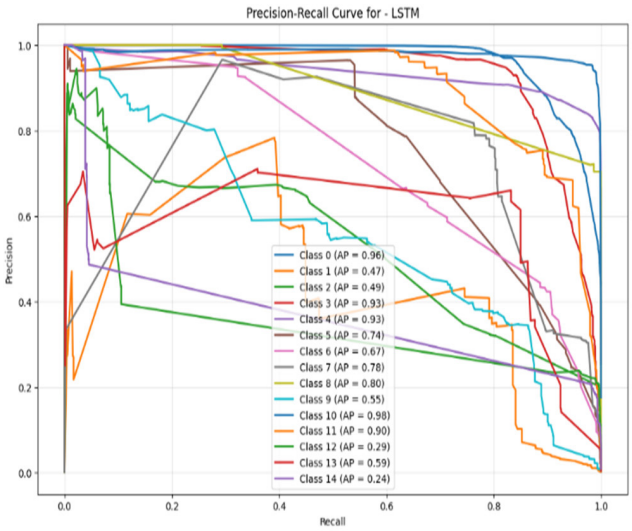

FIGURE 21. Precision-recall curve of LSTM.

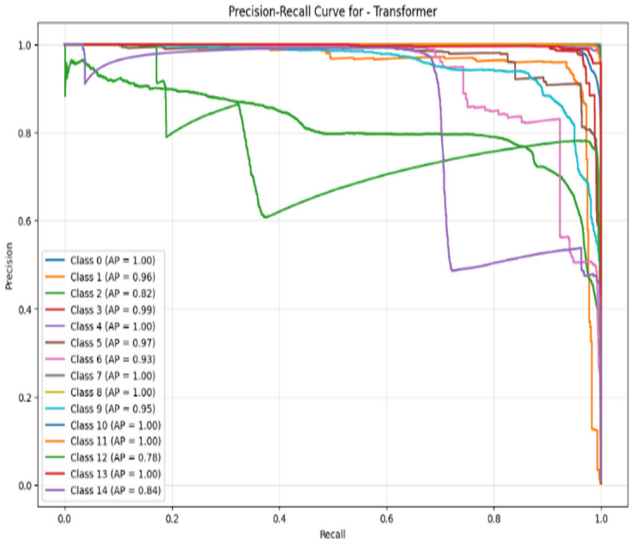

FIGURE 22. Precision-recall curve of transformer.

relatively higher rates of misclassifications at lower thresholds. The broken diagonal line in the plot shows the performance of a stochastic classifier with AUC = 0.50 and compares the baseline performance with the considered models.

The XGBoost and Decision Tree curves are similar and near the top-left axis, demonstrating their high sensitivity and specificity. The SVM curve spreads out more slowly, illustrating that it tends to have a greater false positive rate at different thresholds.

C. PRECISION-RECALL

Fig. 17- 21 visualize the Precision recall curve for all six models in our study. XGBoost, Decision Tree, and Transformer maintain extremely high precision at all levels of recall, i.e., extremely accurate and stable intrusion detection.

TABLE 10. Cross-validation summary of six models.

Model	Accuracy	Precision	Recall	F1-score	Balanced Accuracy	MCC	Train Time (s)
KNN	0.9819	0.9829	0.9819	0.9821	0.9767	0.9746	0.99
XGBoost	1.0000	1.0000	1.0000	1.0000	0.9999	1.0000	28.37
Decision Tree	0.9999	0.9999	0.9999	0.9999	0.9996	0.9999	11.22
SVM (SGD)	0.7850	0.7806	0.7850	0.7789	0.4663	0.6992	419.94
LSTM	0.8757	0.8720	0.8757	0.8651	0.6521	0.8240	4748.16
Transformer	0.9626	0.9671	0.9626	0.9603	0.8792	0.9474	5485.75

The performance of KNN remains outstanding except for some minor departures from perfection, and it is not very unstable. The performance of LSTM varies across classes, which may be due to false positives at trying to identify all intrusive attacks. SVM is worst of all - the sharp decline in precision at higher levels of recall indicates an issue of not finding a good trade-off balance, probably due to the complexity of the feature space.

D. CROSS-VALIDATION PERFORMANCE COMPARISON

A consistent experimental setup was applied to evaluate the generalizability and robustness of the six classification models, in which 70% of the dataset was used for training and 30% for testing. The of each model was assessed using a comprehensive set of metrics, including accuracy, precision, recall, F1-score, balanced accuracy, and Matthews Correlation Coefficient (MCC). The training time was also recorded to assess computational efficiency. The comparative results are summarized in Table 10.

The KNN model achieved 98.19% accuracy, with high precision (0.9829) and F1-score (0.9821), reflecting its consistent ability to classify regular and attack classes. A balanced accuracy of 0.9767 and an MCC of 0.9746 indicate strong performance across imbalanced classes. Notably, it required only 0.99 seconds of training time, making it the most computationally efficient among all models evaluated.

XGBoost outperformed all other models with 100% accuracy across almost all evaluation metrics except balanced accuracy, which is 99.99%. This suggests exceptional robustness and generalization capability in detecting all categories of network traffic, including minority attacks. Despite its complexity, it maintained a reasonable training time of 28.37 seconds, indicating an excellent trade-off between performance and efficiency.

The Decision Tree classifier also delivered near-perfect performance, with 99.99% accuracy and similarly high values across all other metrics. It achieved a balanced accuracy of 0.9996 and an MCC of 0.9999, demonstrating excellent predictive power. Its training time was minimal at 11.22 seconds, confirming its suitability for fast deployment in time-sensitive environments.

The SVM model, trained using the SGD optimizer, exhibited moderate performance, achieving 78.50% accuracy and an F1-score of 0.7789. However, its balanced accuracy of 0.4663 and MCC of 0.6992 reveal limitations in handling class imbalance. Moreover, the training time of 419.94 seconds was relatively high compared to its performance, suggesting inefficiencies in scalability and generalization.

The LSTM model produced reasonable performance with an accuracy of 87.57%, an F1-score of 0.8651, and a balanced accuracy of 0.6521. It maintained a good MCC score of 0.8240, indicating strong predictive reliability. However, it incurred a high training time of 4748.16 seconds, which could be a bottleneck in real-time or resource-constrained environments.

The Transformer model achieved strong performance, with 96.26% accuracy, precision of 0.9671, and F1-score of 0.9603. It maintained a balanced accuracy of 0.8792 and a high MCC of 0.9474, reflecting excellent capability in handling multiclass and imbalanced scenarios. However, it was the most computationally intensive model, with a training time of 5485.75 seconds, highlighting the need for GPU acceleration or architectural optimization for practical deployment.

The proposed intrusion detection system is designed to work efficiently with large amounts of data and in real-time network environments. By using a feature optimization technique, the system reduces the number of features, which helps lower memory use and speeds up processing. Among all the models tested, XGBoost showed the best performance for real-time detection because it is fast, uses system resources efficiently, and can handle large datasets easily. It can also be trained quickly and supports parallel processing, which makes it scalable and practical for real-world use.

Although deep learning models like LSTM and Transformer need more computing power, their performance can be improved by using GPUs, batch processing, and model optimization techniques. The Transformer model, in particular, is more efficient in handling complex and high-dimensional data due to its parallel attention mechanism. Additionally, the system's structure—where data preparation, feature selection, and classification are handled in separate steps—makes it easier to use across multiple systems or devices, including edge and cloud platforms. Overall, the method balances good detection accuracy with fast and efficient processing, making it a strong choice for real-time intrusion detection.

VI. DISCUSSION

The dataset used in our research is the benchmark dataset of real network traffic for evaluating anomaly-based intrusion

TABLE 11. Comparison of our study with recent works on the CICIDS2017 dataset.

Reference	Methodology	Approaches & Techniques	Key contributions	Limitations	Result
[32]	SMOTE-Tomek Links balancing	DT, RF, XGBoost, KNN, AdaBoost	Improved detection of balanced data	Long training time for binary Classification	Binary: 99.96%, Multiclass: 96.37%
[33]	Transformer architecture (self-attention)	Transformer encoder-decoder model	Robust feature extraction for high-dimensional data	High computational demand	99.17% F1-score (Transformer)
[34]	Enhanced Hybrid Deep Learning	Autoencoder–CNN, Transformer–DNN	Robust imbalance handling	Computational complexity	Binary: ~99.90-99.92%, Multiclass: ~99.95-99.96%
[35]	Stacked Ensemble (Random Forest, Extra Trees, XGBoost)	Ensemble with robust feature selection	Effective feature reduction	Offline dataset reliance	Binary: ~99.90%
[36]	LSTM + SMOTE	Deep learning with focal loss	Handles rare attacks effectively	Performance variance in the testing phase	~99.33%
[37]	Hybrid deep learning-based IDS	LSTM based IDS coupled with SMOTE and RFE	Binary classification with feature extraction and rare attacks handling	Model complexity and scalability	Binary: ~99.34%
[38]	Multiclass Classification	Decision Trees, ANN, CNN	Strong interpretability with XAI	High feature dependency	F1-score: ~0.96878
Proposed Work	Enhanced feature selection (Extra Trees Classifier) with automatic thresholding, SMOTE + undersampling to handle class imbalance	Decision Tree, XGBoost, KNN, SVM, LSTM, Transformer	Comprehensive preprocessing, feature importance identification, multi-model evaluation	Model Interpretability	Multiclass classification: XGBoost: 100%, Decision Tree: 99.96% Transformer: 96.26%

detection systems, which has also been reported in many other publications. For comparison, we have selected some similar publications [25], [26], [32], [33], [34], [35], [36], and [37] which use the CICIDS2017 dataset [40]. Table 11 shows the comparison of our proposed work. In [25], oversampling and undersampling were used to address the class imbalance, while RTIDS [26] used SMOTE to balance classes artificially. Our solution, by contrast, takes recourse to SMOTE and selective undersampling of the dominant benign class and hence gains a controlled, optimized balance. Feature selection in [25] employs statistical methods such as chi-square and information gain for dimension reduction, but RTIDS employs the self-attention process of the Transformer model to learn feature importance implicitly. The model we employed explicitly chooses the 42 most important features out of 84 original attributes based on feature importance scores, which improves model efficiency. The performance is evaluated based on standard parameters—Accuracy, Precision, Recall, and F1-score—to several baseline and deep learning models such as Decision Tree (DT), XGBoost, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Transformer, and Long Short-Term Memory (LSTM). The result in Table 11 shows the suggested Decision Tree and XGBoost models outperform their counterparts in other research work with accuracy rates (1.00) over all the evaluation parameters. Similarly, the KNN model in the suggested work also shows improved performance with 0.98 accuracy and 0.97 F1-score grading. The overall comparative analysis confirms the suggested XGBoost, Decision tree, KNN, and Transformer models achieve significant improvement in detection accuracy and stability in comparison to the latest techniques presented in the current literature. A comparative analysis with recent studies demonstrates that our evaluation framework is the most comprehensive, integrating cross-validation accuracy, micro-averaged AUC-ROC curves, and precision-recall curves over traditional measures. The wide-range validation between classical and deep learning frameworks reflects the dominance and resilience of our proposed anomaly-based intrusion detection methods for identifying different network attacks. Our proposed method of feature selection and balancing dataset significantly improved the detection accuracy of the ensemble, decision tree, and Transformer model.

VII. CONCLUSION AND FUTURE WORK

The study systematically evaluated six machine learning and deep learning models—Decision Tree, XGBoost, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), LSTM, and Transformer—under a multiclass classification setup. Our framework emphasized detailed and fair evaluation by incorporating overall accuracy, precision, recall, F1-score, and ROC-AUC metrics. Among all models, the Decision Tree and XGBoost achieved exceptional performance with near-perfect classification metrics across all classes.

Transformer and KNN also demonstrated strong capabilities, especially in identifying common and rare intrusions. The findings validate our hypothesis that the synergy between well-calibrated data balancing and informed feature reduction significantly enhances IDS performance. The results also demonstrate that ensemble methods like XGBoost and interpretable models like Decision Tree can be accurate and practical for real-time deployment. At the same time, Transformer architecture offers promising adaptability to high-dimensional, imbalanced network traffic data.

This study was conducted using the CICIDS2017 dataset, which, while realistic and widely used, may not capture the full variability of real-world traffic. In future work, the framework can be evaluated on other benchmarks and real-time datasets to enhance generalizability. The current evaluation was performed in an offline setting. For deployment in real-time environments, the proposed framework will be optimized for online processing. Additionally, future extensions may explore adaptive feature selection and incorporate interpretability techniques to further improve practical usability in network security operations. The deployment of this IDS on edge and cloud environments, along with performance benchmarking under adversarial conditions, also remains a critical direction for future exploration.

REFERENCES

[1] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network Intrusion Detection and Prevention*. Boston, MA, USA: Springer, 2010, doi: 10.1007/978-0-387-88771-5.

[2] J. P. Anderson, *Computer Security Technology Planning Study*, document ESD-TR-73-51, 1972, p. 4, vol. 1.

[3] R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–35, Nov. 2020.

[4] W. Stallings and L. Brown, *Computer Security: Principles and Practice*. London, U.K.: Pearson, 2015.

[5] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.

[6] G. Engelen, V. Rimmer and W. Joosen, "Troubleshooting an intrusion detection dataset: the CICIDS2017 case study," in *Proc. IEEE Security Privacy Workshops (SPW)*, San Francisco, CA, USA, 2021, pp. 7–12, doi: 10.1109/SPW53761.2021.00009.

[7] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3.24, pp. 479–482, 2018.

[8] A. Rosay, F. Carlier, E. Cheval, and L. Pascal, "From CIC-IDS2017 to LYCOS-IDS2017: A corrected dataset for better performance," in *IEEE/WIC/ACM Int. Conf. Web Intelligence*, Dec. 2021, doi: 10.1145/3486622.3493973.

[9] M. S. Korium, M. Saber, A. Beattie, A. Narayanan, S. Sahoo, and P. H. J. Nardelli, "Intrusion detection system for cyberattacks in the Internet of Vehicles environment," *Ad Hoc Netw.*, vol. 153, Feb. 2024, Art. no. 103330.

[10] B. Selvakumar, M. Sivaanandh, K. Muneeswaran, and B. Lakshmanan, "Ensemble of feature augmented convolutional neural network and deep autoencoder for efficient detection of network attacks," *Sci. Rep.*, vol. 15, no. 1, pp. 570–575, Feb. 2025.

[11] A. A. Alfrhan, R. H. Alhusain, and R. U. Khan, "SMOTE: Class imbalance problem in intrusion detection system," in *Proc. Int. Conf. Comput. Inf. Technol. (ICCIT)*, Tabuk, Saudi Arabia, Sep. 2020, pp. 1–5, doi: 10.1109/ICCIT-144147971.2020.9213728.

[12] S. Ho, S. A. Jufout, K. Dajani, and M. Mozumdar, "A novel intrusion detection model for detecting known and innovative cyberattacks using convolutional neural network," *IEEE Open J. Comput. Soc.*, vol. 2, pp. 14–25, 2021, doi: 10.1109/OJCS.2021.3050917.

[13] G. Karatas, O. Demir, and O. K. Sahingoz, "Deep learning in intrusion detection systems," in *Proc. Int. Congr. Big Data, Deep Learn. Fighting Cyber Terrorism (IBIGDELFT)*, Dec. 2018, pp. 113–116.

[14] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Softw. Netw.*, vol. 2017, no. 1, pp. 177–200, 2017.

[15] Q. Zhou and D. Pezaros, "Evaluation of machine learning classifiers for zero-day intrusion detection—An analysis on CIC-AWS-2018 dataset," 2019, *arXiv:1905.03685*.

[16] J. Kim, Y. Shin, and E. Choi, "An intrusion detection model based on a convolutional neural network," *J. Multimedia Inf. Syst.*, vol. 6, no. 4, pp. 165–172, 2019.

[17] V. Kanimozhi and T. P. Jacob, "Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2019, pp. 33–36.

[18] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32150–32162, 2020.

[19] Y. Arafat, K. S. Yeaser, A. Rahman, and A. Dasgupta, "A machine learning based approach for protecting wireless networks against DoS attacks," in *Proc. 7th Int. Conf. Netw., Syst. Secur.*, Dec. 2020, pp. 126–132.

[20] S. S. Gopalan, D. Ravikumar, D. Linekar, A. Raza, and M. Hasib, "Balancing approaches towards ML for IDS: A survey for the CSE-CIC IDS dataset," in *Proc. Int. Conf. Commun., Signal Process., their Appl. (ICCSPA)*, Mar. 2021, pp. 1–6.

[21] S. Mubarak, M. H. Habaebi, M. R. Islam, and S. Khan, "ICS cyber attack detection with ensemble machine learning and DPI using cyberkit datasets," in *Proc. 8th Int. Conf. Comput. Commun. Eng. (ICCCE)*, Jun. 2021, pp. 349–354.

[22] R. Singh and G. Srivastav, "Novel framework for anomaly detection using machine learning technique on CIC-IDS2017 dataset," in *Proc. Int. Conf. Technological Advancements Innov. (ICTAI)*, Nov. 2021, pp. 632–636.

[23] R. Lalduhsaka, N. Bora, and A. K. Khan, "Anomaly-based intrusion detection using machine learning: An ensemble approach," *Int. J. Inf. Secur. Privacy (IJISP)*, vol. 16, no. 1, pp. 1–15, 2022.

[24] J. Arevalo-Herrera, J. E. C. Mendoza, and J. I. M. Torre, "Network anomaly detection with machine learning techniques for SDN networks," in *Proc. 7th Int. Conf. Inf. Educ. Innov. (ICIEI)*, Apr. 2022, pp. 129–135.

[25] F. A. Khan, A. A. Shah, N. Alshammry, S. Saif, W. Khan, M. O. Malik, and Z. Ullah, "Balanced multi-class network intrusion detection using machine learning," *IEEE Access*, vol. 12, pp. 178222–178236, 2024.

[26] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375–64387, 2022.

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

[28] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving AdaBoost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," *J. Phys., Conf. Ser.*, vol. 1192, Mar. 2019, Art. no. 012018.

[29] C. Zhang, Y. Chen, Y. Meng, F. Ruan, R. Chen, Y. Li, and Y. Yang, "A novel framework design of network intrusion detection based on machine learning techniques," *Secur. Commun. Netw.*, vol. 2021, pp. 1–15, Jan. 2021.

[30] A. A. A. Alsameraee and M. K. Ibrahem, "Toward constructing a balanced intrusion detection dataset on CICIDS2017," *Samarra J. Pure Appl. Sci.*, vol. 2, no. 3, pp. 132–142, Sep. 2021.

[31] A. Raza, K. Munir, M. S. Almutairi, and R. Sehar, "Novel class probability features for optimizing network attack detection with machine learning," *IEEE Access*, vol. 11, pp. 98685–98694, 2023.

[32] H. Kamal and M. Mashaly, "Enhanced hybrid deep learning models-based anomaly detection method for two-stage binary and multi-class classification of attacks in intrusion detection systems," *Algorithms*, vol. 18, no. 2, p. 69, Jan. 2025.

[33] W. F. Urmi, M. N. Uddin, M. A. Uddin, M. A. Talukder, M. R. Hasan, S. Paul, M. Chanda, J. Ayoade, A. Khraisat, R. Hossen, and F. Imran, "A stacked ensemble approach to detect cyber attacks based on feature selection techniques," *Int. J. Cognit. Comput. Eng.*, vol. 5, pp. 316–331, Jan. 2024.

[34] M. W. Nawaz, R. Munawar, A. Mehmood, M. M. Ur Rahman, and Q. H. Abbasi, "Multi-class network intrusion detection with class imbalance via LSTM & SMOTE," 2023, *arXiv:2310.01850*.

[35] H. R. Sayegh, W. Dong, and A. M. Al-madani, "Enhanced intrusion detection with LSTM-based model, feature selection, and SMOTE for imbalanced data," *Appl. Sci.*, vol. 14, no. 2, p. 479, Jan. 2024, doi: 10.3390/app14020479.
[36] M. Bacevicius and A. Paulauskaite-Taraseviciene, "Machine learning algorithms for raw and unbalanced intrusion detection data in a multi-class classification problem," *Appl. Sci.*, vol. 13, no. 12, p. 7328, Jun. 2023.
[37] scikit-learn. (2019). *Scikit-learn: Machine Learning in Python-Scikit-learn 0.16.1 Documentation*. Scikit-learn.org. [Online]. Available: https://scikit-learn.org/
[38] Numpy. (2024). *NumPy*. Numpy.org. [Online]. Available: https://numpy.org/
[39] PYTHON. (2025). *Python*. Python.org. [Online]. Available: https://www.python.org/
[40] Canadian Institute for Cybersecurity. (2017). *IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. [Online]. Available: www.unb.ca, 2017. https://www.unb.ca/cic/datasets/ids-2017.html

NAHIDA NIGAR received the B.Sc. (Engg.) and M.Sc. (Engg.) degrees in computer science and engineering from the University of Chittagong, Bangladesh. She is currently an Assistant Professor with the Department of Computer Science and Engineering, East Delta University, Bangladesh. Previously, she held academic positions with Southern University Bangladesh and Asian University for Women. She has nearly a decade of teaching experience. Her research contributions include several publications in reputed international journals, primarily in the areas of networking, wireless communications, network security, and artificial intelligence. Her current research interests include cybersecurity, machine learning, deep learning, intrusion detection systems, and explainable AI.

RASHED MUSTAFA received the B.Sc. (Engineering) degree in computer science and engineering from the Shahjalal University of Science and Technology, Bangladesh, in 2003, the M.Phil. degree in computer science and engineering from the University of Chittagong, Bangladesh, in 2010, and the Ph.D. degree in computer application technology from Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China, in 2015. He is currently a Professor with the Department of Computer Science and Engineering, University of Chittagong, Bangladesh. With more than two decades of teaching and research experience, he has made significant contributions to high-performance computing, cloud video processing, image processing, and pattern recognition. He was also the Dean of the Faculty of Engineering, University of Chittagong, from 2020 to 2024. He was an Adjunct Faculty with Binary University, Malaysia, from 2024 to 2025. He is working on resolving geospatial semantic conflicts using ontology. His research focused on accelerating cloud video censoring methods for efficient obscene content identification. His research interests include cloud-based video processing, high-performance computing, deep learning, and knowledge management. Over the years, he has led multiple funded research projects. His work has been published in leading international journals, such as *Journal of Universal Computer Science*, *Soft Computing*, *Internet of Things*, and *Jurnal Kejuruteraan*. He has received several prestigious scholarships and awards. He has been invited as a keynote speaker at various international conferences (Australia, U.K., Malaysia, Lebanon, Japan, Italy, and India). Beyond research, he is actively engaged in academic mentoring, supervising Ph.D. and M.Phil. students. He remains committed to fostering innovation in cloud-based video processing, artificial intelligence, and computer vision applications. His dedication to research excellence and global collaboration continues to have a profound impact on the field.