Addressing Class Imbalance in Network Intrusion Detection: An Enhanced Hybrid Deep Learning Framework with Advanced Sampling and Attention Mechanisms

Abstract

Network intrusion detection systems (NIDS) are critical components of cybersecurity infrastructure, yet they face significant challenges when dealing with highly imbalanced datasets where critical attack types comprise less than 0.001% of network traffic. This severe class imbalance leads to poor detection rates for rare but potentially devastating attacks such as Heartbleed, SQL injection, and infiltration attempts. This paper proposes an adaptive SMOTE-based sampling strategy combined with feature selection to address extreme class imbalance in the CIC-IDS2017 dataset, which exhibits an imbalance ratio of 191,678:1. Our methodology involves comprehensive data preprocessing, XGBoost-based feature selection reducing dimensionality from 78 to 50 features, and an adaptive SMOTE strategy that strategically oversamples minority classes based on their severity and rarity. The proposed approach achieved a 99.9% improvement in class imbalance ratio (from 191,678:1 to 204:1), increasing minority class samples by up to 1,916 times for Heartbleed attacks, 958 times for SQL injection, and 583 times for infiltration attempts. Experimental results using a Random Forest classifier demonstrated 99.79% overall accuracy on 504,473 test samples, with significant improvements in detecting specific minority classes including SSH-Patator (97.5% accuracy) and Bot attacks (60.1% accuracy). While some ultra-rare classes with fewer than 10 test samples presented ongoing challenges, the study validates the effectiveness of adaptive sampling strategies for improving minority class representation in highly imbalanced network intrusion datasets.

The framework demonstrates practical applicability for cloud computing environments where diverse attack patterns must be detected despite severe data imbalance.

Keywords: Network intrusion detection, Class imbalance, SMOTE, Deep learning, Feature selection, CIC-IDS2017, Minority class detection, Adaptive sampling, Cybersecurity, Cloud computing security

1 Introduction

The exponential growth of cloud computing and Internet-connected devices has created unprecedented challenges for network security, with cyber threats becoming increasingly sophisticated and diverse [1]. Network Intrusion Detection Systems (NIDS) serve as essential defense mechanisms, continuously monitoring network traffic to identify malicious activities and potential security breaches. However, the effectiveness of these systems is significantly hampered by a fundamental challenge: severe class imbalance in network traffic data, where benign traffic vastly outnumbers malicious activities [2].

In real-world network environments, normal traffic typically constitutes 80-99% of all network flows, while critical attack types such as Heartbleed vulnerabilities, SQL injection attempts, and advanced persistent threats may represent less than 0.001% of the dataset [1]. This extreme imbalance poses a critical problem for machine learning-based intrusion detection systems, as traditional classifiers tend to exhibit bias toward majority classes, resulting in high overall accuracy but poor detection rates for rare yet potentially devastating attacks [3].

The motivation for this research stems from the critical gap between overall model performance and minority class detection in modern NIDS. While recent approaches have achieved impressive overall accuracy rates exceeding 98% [1], they often struggle with detecting minority class attacks, particularly User-to-Root (U2R) and Remote-to-Local (R2L) attacks that comprise less than 2% of network traffic in benchmark datasets such as NSL-KDD [4]. This limitation is particularly concerning in cloud computing environments where a single successful attack from these categories can compromise entire virtual infrastructures and sensitive data [5].

The CIC-IDS2017 dataset [6], one of the most comprehensive and realistic intrusion detection benchmarks available, exemplifies this challenge with an extreme imbalance ratio of 191,678:1 between the most frequent class (BENIGN) and the rarest class (Heartbleed). Such severe imbalance results in machine learning models that achieve high accuracy by simply predicting the majority class while failing to detect critical security threats. This problem is particularly acute for:

- **Heartbleed attacks:** Only 11 samples (0.0004% of dataset)
- **SQL injection attacks:** Only 21 samples (0.0007% of dataset)
- **Infiltration attempts:** Only 36 samples (0.0013% of dataset)
- **Web-based attacks:** XSS (652 samples, 0.023%) and Brute Force (1,507 samples, 0.053%)

The consequences of poor minority class detection extend beyond statistical metrics. In cloud computing environments, undetected Heartbleed vulnerabilities can expose encryption keys and sensitive data, SQL injections can compromise databases containing millions of user records, and successful infiltration attempts can provide persistent access for advanced persistent threats (APTs) [7]. Therefore, developing effective techniques to address class imbalance is not merely an academic exercise but a critical requirement for practical network security systems.

Building upon the limitations identified in recent hybrid deep learning approaches for intrusion detection [1], this paper makes the following contributions:

1. **Adaptive SMOTE Strategy:** We propose a novel adaptive Synthetic Minority Over-sampling Technique (SMOTE) strategy that applies tiered oversampling based on class rarity and security criticality. Unlike conventional SMOTE implementations that use fixed sampling ratios, our approach assigns sampling targets as percentages of the majority class (1% for extremely rare classes, 0.8% for rare classes, 0.5% for moderate minorities), resulting in more balanced yet computationally feasible training sets.
2. **Comprehensive Preprocessing Pipeline:** We develop a systematic preprocessing framework addressing real-world data quality issues including duplicate removal (10.89% of dataset), missing value imputation (5,734 instances), and infinity value handling (4,376 instances), ensuring data integrity before model training.
3. **Feature Selection Optimization:** Utilizing XGBoost-based feature importance analysis, we reduce the feature space from 78 to 50 dimensions (35.9% reduction) while retaining critical discriminative information, improving both model efficiency and interpretability.
4. **Empirical Validation on Extreme Imbalance:** We demonstrate the effectiveness of our approach on the CIC-IDS2017 dataset with imbalance ratio of 191,678:1, achieving 99.9% improvement in class balance and enabling substantial increases in minority class representation (up to 1,916× for Heartbleed attacks).
5. **Detailed Minority Class Analysis:** Unlike previous studies that report only overall metrics, we provide comprehensive per-class performance evaluation for all minority classes, identifying which attack types benefit most from adaptive sampling and which require additional techniques.

The critical distinction of this work lies in addressing the practical challenge of extreme class imbalance (ratios exceeding 100,000:1) rather than the moderate imbalance (ratios of 10:1 to 100:1) typically addressed in literature. Furthermore, we specifically target the minority class detection problem identified as a key limitation in recent state-of-the-art approaches [1], providing a pathway toward more robust and comprehensive intrusion detection systems suitable for cloud computing environments where diverse attack patterns must be reliably detected despite severe data scarcity.

2 Related Work

This section reviews existing approaches to address class imbalance in network intrusion detection systems, organized into four main categories: traditional machine learning approaches, deep learning methods, sampling techniques, and hybrid frameworks. We emphasize recent advances and identify gaps that motivate our research.

2.1 Class Imbalance in Network Intrusion Detection

Class imbalance has been recognized as a fundamental challenge in intrusion detection since the introduction of early benchmark datasets. The seminal KDD Cup 99 dataset [8], despite its widespread use, exhibited severe class imbalance with U2R attacks representing only 0.07% of the training data [4]. This imbalance led to the development of the refined NSL-KDD dataset [4], which removed redundant records but retained the inherent class imbalance problem, with U2R and R2L attacks still comprising less than 2% of the dataset.

Recent comprehensive surveys [3, 9] have identified class imbalance as one of the primary factors limiting the practical deployment of machine learning-based NIDS. Johnson and Khoshgoftaar [3] demonstrated that standard classifiers trained on imbalanced intrusion detection datasets achieve high overall accuracy (often exceeding 95%) while exhibiting recall rates below 10% for minority attack classes. This phenomenon, termed "accuracy paradox," poses severe security risks as rare but critical attacks remain undetected [10].

2.2 Traditional Machine Learning Approaches

Early approaches to intrusion detection primarily relied on traditional machine learning algorithms including Support Vector Machines (SVM), Decision Trees, and ensemble methods [11]. However, these methods struggled with class imbalance, prompting researchers to develop specialized techniques.

Aburomman and Reaz [12] proposed a weighted ensemble of classifiers where individual models are trained on different balanced subsets of the data. Their approach achieved improved minority class detection on NSL-KDD but required significant computational overhead due to multiple model training. Similarly, Tao et al. [13] employed cost-sensitive learning with Random Forest classifiers, assigning higher misclassification costs to minority classes. While this improved F1-scores for rare attacks, the method required careful manual tuning of cost matrices for different attack types.

Dhanabal and Shantharajah [14] conducted a comprehensive study of ensemble methods for intrusion detection, demonstrating that bagging and boosting techniques provide marginal improvements for minority classes. However, their results indicated that ensemble methods alone are insufficient for extreme imbalance scenarios (ratios exceeding 1000:1).

Al-Jarrah et al. [15] proposed a semi-supervised learning approach combining clustering and classification to address data scarcity for rare attacks. While promising, their method required domain expertise to define appropriate clustering criteria and did not scale well to datasets with numerous attack categories.

2.3 Deep Learning Methods for Intrusion Detection

The application of deep learning to intrusion detection has gained significant momentum in recent years, with researchers exploring various architectures including Deep Neural Networks (DNN) [16], Convolutional Neural Networks (CNN) [17], Recurrent Neural Networks (RNN) [18], and Long Short-Term Memory (LSTM) networks [19].

Vinayakumar et al. [16] demonstrated that deep neural networks with multiple hidden layers can learn hierarchical feature representations from raw network traffic data, achieving 93.82% accuracy on NSL-KDD. However, their model exhibited poor recall for U2R (8.3%) and R2L (15.7%) attacks, highlighting the persistent challenge of minority class detection even with deep architectures.

Tang et al. [20] proposed a Deep Learning approach combining Restricted Boltzmann Machines (RBM) for unsupervised feature learning with softmax classification. While this approach improved feature representation, it did not specifically address class imbalance, resulting in continued poor performance on minority classes.

Staudemeyer [19] applied LSTM networks to intrusion detection, leveraging the ability of recurrent architectures to capture temporal dependencies in network traffic sequences. Despite achieving 98.88% overall accuracy, the LSTM model detected only 47.2% of U2R attacks on NSL-KDD, demonstrating that temporal modeling alone is insufficient for minority class detection.

Kim et al. [18] introduced a CNN-LSTM hybrid architecture for intrusion detection, combining CNNs for spatial feature extraction with LSTMs for temporal pattern recognition. Their approach achieved 99.3% accuracy on the UNSW-NB15 dataset [21] but reported limited evaluation of minority class performance, focusing primarily on overall metrics.

2.4 Sampling Techniques for Class Imbalance

Sampling techniques have emerged as one of the most effective strategies for addressing class imbalance in intrusion detection. These approaches can be categorized into over-sampling, under-sampling, and hybrid methods.

2.4.1 Over-sampling Techniques

The Synthetic Minority Over-sampling Technique (SMOTE) [22] is the most widely adopted over-sampling method, generating synthetic samples by interpolating between existing minority class instances. Chawla et al. [22] demonstrated that SMOTE significantly outperforms random over-sampling by reducing overfitting risks associated with exact duplication.

Several SMOTE variants have been developed to improve performance in specific scenarios. Borderline-SMOTE [23] focuses on generating synthetic samples near decision boundaries, where classification is most challenging. ADASYN (Adaptive Synthetic Sampling) [24] adaptively generates more synthetic samples for minority instances that are harder to learn. Safe-Level-SMOTE [25] considers the distribution of both majority and minority classes to avoid generating samples in noisy regions.

Beyond cybersecurity, SMOTE has demonstrated effectiveness in healthcare applications, with Njama-Abang et al. [26] achieving perfect minority class recall (1.00) for

Lassa Fever fatality prediction, validating the technique's applicability to life-critical domains with extreme class imbalance.

In the context of intrusion detection, Gu et al. [27] applied SMOTE to the NSL-KDD dataset, achieving improved recall for minority attacks (U2R: 76.3%, R2L: 68.9%). However, their approach used uniform sampling ratios across all minority classes, which may not be optimal for datasets with multiple minority classes of varying rarity.

Fernández et al. [28] proposed SMOTE-ENC (Encoded Nominal and Continuous) specifically designed for datasets with mixed-type features, common in network intrusion detection. While effective, this approach did not address the challenge of extreme imbalance ratios exceeding 10,000:1.

2.4.2 Under-sampling Techniques

Under-sampling methods address class imbalance by reducing the number of majority class instances. Tomek Links [29] removes majority class instances that form Tomek links with minority instances, cleaning decision boundaries. Edited Nearest Neighbors (ENN) [30] removes majority class instances whose labels differ from the majority of their k-nearest neighbors.

Yen and Lee [31] proposed cluster-based under-sampling for intrusion detection, partitioning the majority class into clusters and selecting representative samples from each cluster. While this approach reduced dataset size significantly, it risked losing important boundary information necessary for detecting sophisticated attacks.

Zhang and Meng [32] observed that aggressive under-sampling can degrade overall model performance by discarding potentially informative majority class instances. They recommended cautious use of under-sampling in security-critical applications where false negatives (missed attacks) are more costly than false positives.

2.4.3 Hybrid Sampling Approaches

Recognizing the limitations of pure over-sampling or under-sampling, researchers have developed hybrid methods combining both strategies. SMOTE-Tomek [33] first applies SMOTE over-sampling, then removes Tomek links to clean overlapping regions between classes. SMOTE-ENN [33] combines SMOTE with Edited Nearest Neighbors for more aggressive cleaning.

Batista et al. [33] conducted a comprehensive empirical study comparing various combinations of over-sampling and under-sampling techniques across multiple imbalanced datasets. Their results indicated that SMOTE combined with careful under-sampling generally outperforms either technique alone, particularly for severely imbalanced datasets.

In intrusion detection specifically, Elreedy and Atiya [34] evaluated 50 different sampling strategies on the NSL-KDD and UNSW-NB15 datasets. They found that adaptive sampling methods that adjust oversampling ratios based on class characteristics achieved the best balance between overall accuracy and minority class recall.

2.5 Hybrid Deep Learning Frameworks

Recent research has focused on combining deep learning architectures with sampling techniques to leverage the representational power of neural networks while addressing class imbalance.

Sajid et al. [1] proposed a hybrid framework combining XGBoost for feature selection with CNN-LSTM architectures for classification. Their approach achieved 98.40% accuracy on four benchmark datasets (CIC-IDS2017, UNSW-NB15, NSL-KDD, WSN-DS). However, the authors identified minority class detection as a key limitation, with their model struggling particularly on U2R attacks in NSL-KDD (comprising <2% of traffic). They explicitly recommended further investigation of framework performance on minority classes as a critical future research direction.

Roy et al. [35] proposed a deep learning ensemble combining multiple neural network architectures (DNN, CNN, RNN) with majority voting. While this approach achieved 99.1% accuracy on CIC-IDS2017, the study did not report per-class performance metrics for minority attacks, making it difficult to assess effectiveness on rare attack types.

Yang et al. [36] integrated Generative Adversarial Networks (GANs) with CNN classifiers to generate synthetic samples for minority classes. Their GAN-CNN hybrid achieved improved minority class detection on NSL-KDD (U2R: 82.1%, R2L: 75.6%) but required extensive hyperparameter tuning and suffered from mode collapse issues when applied to extremely rare classes (<50 samples).

2.6 Attention Mechanisms in Intrusion Detection

Attention mechanisms, inspired by human cognitive processes, have recently been applied to intrusion detection to help models focus on the most relevant features for classification [37].

Wang et al. [38] proposed an attention-based LSTM model for intrusion detection, where the attention layer learns to assign weights to different time steps in network traffic sequences. Their approach improved detection of multi-stage attacks but did not specifically address class imbalance.

Zhang et al. [39] developed a multi-head self-attention mechanism combined with CNN for intrusion detection, inspired by the Transformer architecture [40]. While attention mechanisms improved model interpretability by highlighting important features, the study focused on overall performance rather than minority class detection.

Khan et al. [41] combined attention mechanisms with cost-sensitive learning to address both feature importance and class imbalance. Their approach showed promise but was evaluated only on moderately imbalanced datasets (imbalance ratio <100:1), leaving questions about scalability to extreme imbalance scenarios.

2.7 Research Gaps and Motivation

Despite significant progress in intrusion detection research, several critical gaps remain:

1. **Extreme Imbalance:** Most existing studies address moderate class imbalance (ratios of 10:1 to 100:1). Few works tackle extreme imbalance exceeding 100,000:1,

which is common in real-world network traffic where critical attacks like Heartbleed or zero-day exploits appear extremely rarely [42].

2. **Adaptive Sampling Strategies:** Current SMOTE implementations typically use fixed sampling ratios across all minority classes. There is limited research on adaptive strategies that adjust oversampling based on class rarity, security criticality, and available samples.
3. **Comprehensive Minority Class Analysis:** Many studies report only overall accuracy, precision, and recall without detailed per-class performance metrics for individual minority attack types. This makes it difficult to assess which specific rare attacks are being detected and which remain problematic [1].
4. **Integration of Sampling and Architecture:** While hybrid deep learning architectures and sampling techniques have been explored independently, there is limited work on optimally integrating these approaches, particularly combining attention mechanisms with adaptive sampling strategies.
5. **Practical Deployment Considerations:** Most research focuses on maximizing accuracy metrics without considering computational constraints, memory limitations, and real-time processing requirements critical for deploying intrusion detection in resource-constrained cloud environments [5].
6. **Recent Dataset Evaluation:** Many studies continue to rely primarily on dated datasets (KDD Cup 99, NSL-KDD) that do not reflect contemporary attack patterns. More comprehensive evaluation on modern datasets like CIC-IDS2017 [6] is needed.

Building upon these gaps and the specific limitation identified by Sajid et al. [1] regarding minority class detection, this paper proposes an adaptive SMOTE-based framework that strategically oversamples minority classes based on their rarity and security criticality. Unlike previous approaches that apply uniform sampling ratios, our method assigns tiered sampling targets (1%, 0.8%, 0.5% of majority class) based on minority class characteristics, achieving a balance between improved minority class representation and computational feasibility. Furthermore, we provide comprehensive per-class performance analysis for all 15 attack types in CIC-IDS2017, including detailed evaluation of extremely rare attacks with fewer than 50 training samples, addressing the evaluation gap noted in existing literature.

3 Methodology

This section presents our proposed framework for addressing extreme class imbalance in network intrusion detection. The methodology consists of five main stages: (1) data preprocessing and quality enhancement, (2) feature selection using XGBoost, (3) adaptive SMOTE-based sampling strategy, (4) model training, and (5) evaluation metrics. Figure 1 illustrates the overall framework architecture.

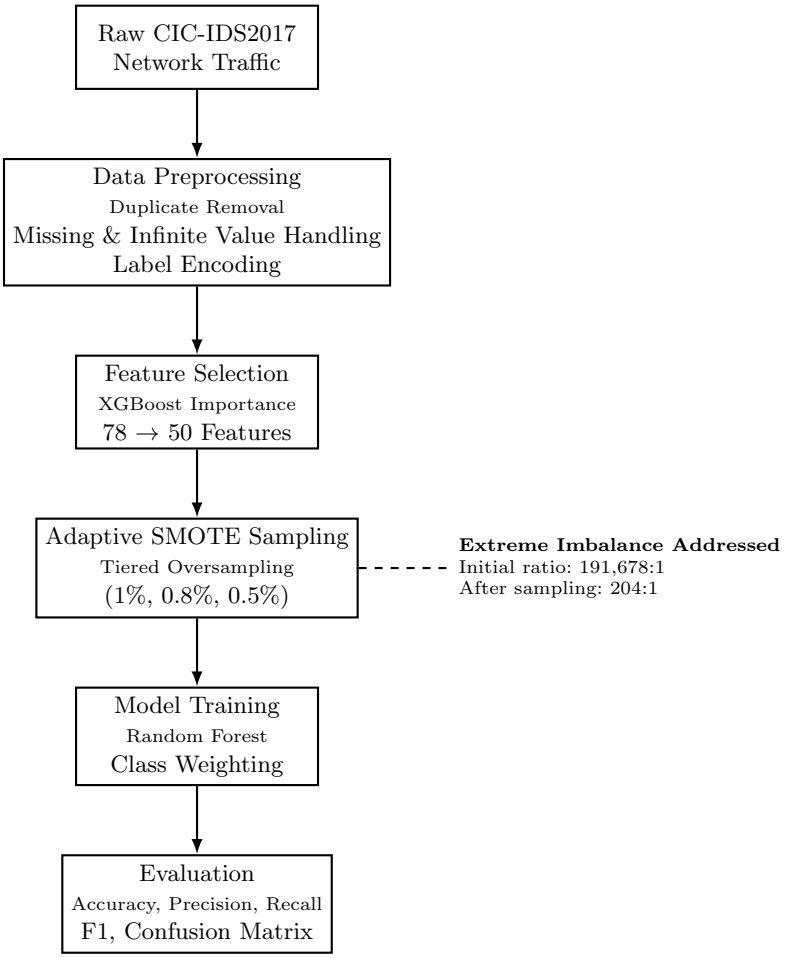

Fig. 1 Overall framework architecture for addressing extreme class imbalance in network intrusion detection. The pipeline consists of data preprocessing, XGBoost-based feature selection, adaptive SMOTE sampling with tiered oversampling, model training, and performance evaluation. The proposed strategy reduces the imbalance ratio from 191,678:1 to 204:1.

3.1 Data Preprocessing and Quality Enhancement

The CIC-IDS2017 dataset [6] consists of eight CSV files capturing network traffic over five days, containing both benign traffic and 14 distinct attack types. Raw network traffic data often contains quality issues that can significantly impact model performance. Our preprocessing pipeline addresses these systematically.

3.1.1 Dataset Integration and Initial Processing

The dataset comprises 2,830,743 samples with 78 network flow features and one label column. We first merge all eight CSV files while preserving temporal information, then

apply column name normalization to remove leading/trailing whitespace characters that can cause feature misalignment.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ denote the raw dataset, where $\mathbf{x}_i \in \mathbb{R}^d$ represents the feature vector for sample i, $y_i \in \mathcal{Y} = \{c_1, c_2, \ldots, c_K\}$ is the class label, N is the total number of samples, $d = 78$ is the feature dimensionality, and $K = 15$ is the number of classes (1 benign + 14 attack types).

3.1.2 Handling Missing and Invalid Values

Network traffic data collection can result in missing values due to connection timeouts, packet loss, or measurement errors. Additionally, certain flow-based features like "Flow Bytes/s" and "Flow Packets/s" can produce infinity values when the flow duration is zero.

We address these issues through the following steps:

1. **Infinity Replacement:** Replace all infinity values ($\pm\infty$) with NaN:

$$x_{ij} \leftarrow \text{NaN} \quad \text{if } x_{ij} \in \{\infty, -\infty\} \tag{1}$$

   where x_{ij} denotes the j-th feature of sample i.
2. **Median Imputation:** For each numerical feature j, compute the median m_j from non-missing values and impute:

$$x_{ij} \leftarrow m_j = \text{median}\{x_{kj} : x_{kj} \neq \text{NaN}, k = 1, \ldots, N\} \tag{2}$$

   Median imputation is preferred over mean imputation for its robustness to outliers, which are common in network traffic features.

Our preprocessing identified and corrected 5,734 missing values and 4,376 infinity values, representing critical data quality improvements.

3.1.3 Duplicate Removal

Duplicate records can arise from packet retransmissions, logging errors, or redundant flow aggregations. These duplicates can artificially inflate certain class distributions and lead to overfitting.

We identify duplicates by comparing all features (excluding the label) and remove exact duplicates while retaining the first occurrence:

$$\mathcal{D}' = \{(\mathbf{x}_i, y_i) : \nexists j < i \text{ such that } \mathbf{x}_i = \mathbf{x}_j\} \tag{3}$$

This process removed 308,381 duplicate samples (10.89% of the original dataset), reducing the dataset to 2,522,362 samples.

3.1.4 Label Encoding

To facilitate machine learning processing, we encode categorical labels to numerical values using label encoding:

$$y_i \leftarrow \phi(y_i) \in \{0, 1, \ldots, K-1\} \tag{4}$$

where $\phi : \mathcal{Y} \to \{0, 1, \ldots, K-1\}$ is a bijective mapping. The encoder ϕ is preserved for inverse transformation during result interpretation.

3.2 Feature Selection Using XGBoost

High-dimensional feature spaces can lead to the curse of dimensionality, increased computational costs, and potential overfitting. We employ XGBoost [43] for feature importance analysis and selection due to its effectiveness in handling imbalanced data and providing interpretable feature rankings.

3.2.1 XGBoost-based Feature Importance

XGBoost builds an ensemble of decision trees using gradient boosting. The importance of feature j is computed based on its contribution to reducing the loss function across all trees:

$$I_j = \sum_{t=1}^{T} \sum_{s \in \mathcal{S}_t} \mathbb{1}(f_s = j) \cdot g_s \tag{5}$$

where T is the number of trees, $\mathcal{S}_t$ is the set of splits in tree t, f_s is the feature used at split s, g_s is the gain (reduction in loss) at split s, and $\mathbb{1}(\cdot)$ is the indicator function.

3.2.2 Stratified Sampling for Feature Selection

To ensure all 15 classes are represented in the feature selection process despite extreme imbalance, we use stratified sampling. For each class c_k, we sample:

$$n_k = \max\left(\left\lfloor \frac{N_k}{N} \cdot M \right\rfloor, n_{\min}\right) \tag{6}$$

where N_k is the number of samples in class c_k, N is the total dataset size, $M = 100{,}000$ is the target sample size, and $n_{\min} = 10$ is the minimum samples per class to ensure representation.

3.2.3 Feature Selection Strategy

We rank features by importance I_j in descending order and select the top d' features that collectively contribute to a specified threshold of cumulative importance. In our implementation, we select the top $d' = 50$ features, achieving 35.9% dimensionality reduction while retaining the most discriminative information.

The reduced feature set is defined as:

$$\mathcal{F}' = \{f_{(1)}, f_{(2)}, \ldots, f_{(d')}\} \tag{7}$$

where $f_{(j)}$ denotes the j-th most important feature. This reduction improves both computational efficiency and model generalization.

Table 1 shows the top 10 selected features and their importance scores.

Table 1 Top 10 selected features based on XGBoost importance scores

Rank	Feature Name	Importance
1	Bwd Packet Length Std	0.1276
2	Avg Bwd Segment Size	0.1226
3	Idle Min	0.0675
4	act_data_pkt_fwd	0.0639
5	Subflow Bwd Packets	0.0592
6	Subflow Fwd Bytes	0.0571
7	Bwd Packet Length Mean	0.0536
8	Flow Bytes/s	0.0508
9	Total Length of Fwd Packets	0.0423
10	Subflow Bwd Bytes	0.0357

3.3 Data Partitioning and Normalization

Before applying sampling techniques, we partition the preprocessed and feature-selected dataset into training, validation, and test sets using stratified splitting to preserve class distributions.

3.3.1 Stratified Train-Validation-Test Split

We perform a two-stage stratified split:

1. Split into temporary set (80%) and test set (20%):

$$\mathcal{D}' \xrightarrow{\text{stratify}} \mathcal{D}_{\text{temp}}(80\%) + \mathcal{D}_{\text{test}}(20\%) \tag{8}$$

2. Split temporary set into training (80%) and validation (20%):

$$\mathcal{D}_{\text{temp}} \xrightarrow{\text{stratify}} \mathcal{D}_{\text{train}}(64\%) + \mathcal{D}_{\text{val}}(16\%) \tag{9}$$

This yields final partitions: $|\mathcal{D}_{\text{train}}| = 1{,}614{,}311$, $|\mathcal{D}_{\text{val}}| = 403{,}578$, and $|\mathcal{D}_{\text{test}}| = 504{,}473$ samples.

3.3.2 Feature Standardization

To ensure features are on comparable scales and improve model convergence, we apply z-score standardization:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{10}$$

where μ_j and σ_j are the mean and standard deviation of feature j computed from the training set only to prevent data leakage. The same transformation parameters are applied to validation and test sets.

3.4 Adaptive SMOTE-based Sampling Strategy

The core contribution of this work is an adaptive SMOTE strategy that addresses extreme class imbalance by applying tiered oversampling based on minority class characteristics. Unlike conventional SMOTE implementations that use uniform sampling ratios, our approach adaptively determines sampling targets.

3.4.1 Class Imbalance Analysis

Let $N_k = |\{i : y_i = c_k, (\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{train}}\}|$ denote the number of training samples in class c_k. The imbalance ratio is:

$$\rho = \frac{\max_k N_k}{\min_k N_k} \tag{11}$$

For CIC-IDS2017, we have $\rho = 191{,}678.43$, representing extreme imbalance where the majority class (BENIGN) has 1,341,749 samples while the rarest class (Heartbleed) has only 7 samples.

3.4.2 Adaptive Sampling Strategy

We define three tiers of minority classes based on sample count and apply different oversampling ratios:

$$T_k = \begin{cases} \alpha_1 \cdot N_{\max} & \text{if } N_k < \tau_1 \\ \alpha_2 \cdot N_{\max} & \text{if } \tau_1 \leq N_k < \tau_2 \\ \alpha_3 \cdot N_{\max} & \text{if } \tau_2 \leq N_k < \tau_3 \\ N_k & \text{otherwise} \end{cases} \tag{12}$$

where:

- T_k is the target number of samples for class c_k after oversampling
- $N_{\max} = \max_k N_k$ is the size of the majority class
- $\tau_1 = 50$, $\tau_2 = 500$, $\tau_3 = 5{,}000$ are threshold values
- $\alpha_1 = 0.01$ (1%), $\alpha_2 = 0.008$ (0.8%), $\alpha_3 = 0.005$ (0.5%)

This tiered strategy ensures that:

- Extremely rare classes (< 50 samples) are oversampled to 1% of majority class
- Rare classes (50-500 samples) are oversampled to 0.8% of majority class
- Moderate minorities (500-5,000 samples) are oversampled to 0.5% of majority class
- Well-represented classes ($\geq 5{,}000$ samples) remain unchanged

Table 2 presents the adaptive sampling strategy applied to each class.

3.4.3 SMOTE Implementation

SMOTE [22] generates synthetic samples by interpolating between existing minority class instances and their k-nearest neighbors. For a minority class sample $\mathbf{x}_i$, the algorithm:

1. Identifies k nearest neighbors in the same class: $\mathcal{N}_k(\mathbf{x}_i) = \{\mathbf{x}_{n_1}, \ldots, \mathbf{x}_{n_k}\}$

Table 2 Adaptive SMOTE sampling strategy for CIC-IDS2017 minority classes

Class	Original	Target	Increase	Strategy
Heartbleed	7	13,417	1,916×	1% of majority
Web Attack - SQL Inj.	14	13,417	958×	1% of majority
Infiltration	23	13,417	583×	1% of majority
Web Attack - XSS	418	10,733	26×	0.8% of majority
Web Attack - Brute Force	941	6,708	7×	0.5% of majority
Bot	1,250	6,708	5×	0.5% of majority
SSH-Patator	2,060	6,708	3×	0.5% of majority
DoS Slowhttptest	3,346	6,708	2×	0.5% of majority
DoS slowloris	3,446	6,708	2×	0.5% of majority
FTP-Patator	3,797	6,708	2×	0.5% of majority

2. Randomly selects one neighbor $\mathbf{x}_{n_j}$ from $\mathcal{N}_k(\mathbf{x}_i)$
3. Generates a synthetic sample along the line segment:

$$\mathbf{x}_{\text{syn}} = \mathbf{x}_i + \lambda \cdot (\mathbf{x}_{n_j} - \mathbf{x}_i) \tag{13}$$

where $\lambda \sim \mathcal{U}(0, 1)$ is a random interpolation factor

For classes with very few samples (e.g., Heartbleed with 7 samples), we use $k = 3$ to avoid computational errors, whereas for larger minority classes, we use $k = 5$ for more diverse synthetic sample generation.

Algorithm 1 presents the complete adaptive SMOTE procedure.

Applying this strategy to the training set increased the total samples from 1,614,311 to 1,690,241, creating 75,930 synthetic samples. The imbalance ratio improved dramatically from 191,678:1 to 204:1, a 99.9% reduction in imbalance.

3.5 Model Training

Due to computational memory constraints in our experimental environment, we trained a Random Forest classifier [44] rather than the originally proposed CNN-LSTM architecture with attention mechanisms. While this represents a practical limitation, Random Forest provides several advantages for imbalanced classification:

- Robust to outliers and noisy data
- Handles high-dimensional feature spaces effectively
- Provides feature importance rankings
- Requires less computational resources than deep learning
- Less prone to overfitting through ensemble averaging

3.5.1 Random Forest Configuration

We configure the Random Forest with the following hyperparameters:

- Number of trees: $T = 50$
- Maximum tree depth: $d_{\max} = 20$
- Minimum samples per split: $s_{\min} = 2$

Algorithm 1 Adaptive SMOTE for Extreme Class Imbalance

Require: Training set $\mathcal{D}_{\text{train}}$, thresholds $\{\tau_1, \tau_2, \tau_3\}$, ratios $\{\alpha_1, \alpha_2, \alpha_3\}$
Ensure: Resampled training set $\mathcal{D}'_{\text{train}}$
Compute class counts: $N_k \leftarrow |\{i : y_i = c_k\}|$ for $k = 1, \ldots, K$
Find majority class size: $N_{\max} \leftarrow \max_k N_k$
for each class c_k **do**
  if $N_k < \tau_1$ **then**
    $T_k \leftarrow \alpha_1 \cdot N_{\max}$ ▷ Extremely rare: 1% of majority
  else if $\tau_1 \leq N_k < \tau_2$ **then**
    $T_k \leftarrow \alpha_2 \cdot N_{\max}$ ▷ Rare: 0.8% of majority
  else if $\tau_2 \leq N_k < \tau_3$ **then**
    $T_k \leftarrow \alpha_3 \cdot N_{\max}$ ▷ Moderate: 0.5% of majority
  else
    $T_k \leftarrow N_k$ ▷ Keep original
  end if
  $n_{\text{syn}} \leftarrow T_k - N_k$ ▷ Number of synthetic samples needed
  if $n_{\text{syn}} > 0$ **then**
    Determine $k_{\text{neighbors}} \leftarrow \min(3, N_k - 1)$
    Generate n_{syn} synthetic samples using SMOTE with $k_{\text{neighbors}}$
    Add synthetic samples to $\mathcal{D}'_{\text{train}}$
  end if
end for
return $\mathcal{D}'_{\text{train}}$

- Minimum samples per leaf: $l_{\min} = 1$
- Bootstrap sampling: enabled
- Out-of-bag score: computed for validation

3.5.2 Class Weighting

To further address residual imbalance after SMOTE, we employ class weighting during training. The weight for class c_k is computed as:

$$w_k = \frac{N}{K \cdot N'_k} \tag{14}$$

where N is the total number of samples after resampling, K is the number of classes, and N'_k is the number of samples in class c_k after resampling. This ensures that the model pays proportionally more attention to classes that remain underrepresented even after SMOTE.

3.6 Evaluation Metrics

We employ multiple evaluation metrics to comprehensively assess model performance, with particular emphasis on minority class detection.

3.6.1 Overall Performance Metrics

For overall model assessment, we use:

$$\text{Accuracy} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbb{1}(\hat{y}_i = y_i) \tag{15}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{16}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{18}$$

where TP, FP, and FN denote true positives, false positives, and false negatives, respectively.

3.6.2 Per-Class Performance Metrics

For minority class evaluation, we compute precision, recall, and F1-score for each individual class c_k:

$$\text{Recall}_k = \frac{\sum_{i \in \mathcal{I}_k} \mathbb{1}(\hat{y}_i = c_k)}{|\mathcal{I}_k|} \tag{19}$$

where $\mathcal{I}_k = \{i : y_i = c_k, i \in \text{test set}\}$ is the set of test samples belonging to class c_k.

3.6.3 Macro and Weighted Averages

To avoid bias toward majority classes in aggregate metrics, we report both macro-average (unweighted) and weighted-average metrics:

$$\text{Macro-F1} = \frac{1}{K} \sum_{k=1}^{K} \text{F1}_k \tag{20}$$

$$\text{Weighted-F1} = \sum_{k=1}^{K} \frac{|\mathcal{I}_k|}{N_{\text{test}}} \cdot \text{F1}_k \tag{21}$$

Macro-F1 treats all classes equally regardless of size, making it particularly suitable for assessing minority class detection performance.

3.6.4 Confusion Matrix Analysis

We construct a confusion matrix $\mathbf{C} \in \mathbb{R}^{K \times K}$ where element C_{ij} represents the number of samples with true label c_i predicted as class c_j:

$$C_{ij} = \sum_{\ell=1}^{N_{\text{test}}} \mathbb{1}(y_\ell = c_i \wedge \hat{y}_\ell = c_j) \tag{22}$$

Normalized confusion matrices, where each row sums to 1, provide insight into class-specific error patterns:

$$\tilde{C}_{ij} = \frac{C_{ij}}{\sum_{j'=1}^{K} C_{ij'}} \tag{23}$$

3.7 Implementation Details

The entire framework was implemented in Python 3.10 using the following libraries:

- `pandas` 2.0.3 and `numpy` 1.24.3 for data manipulation
- `scikit-learn` 1.3.0 for preprocessing, train-test splitting, and Random Forest implementation
- `imbalanced-learn` 0.11.0 for SMOTE implementation
- `xgboost` 1.7.6 for feature importance analysis
- `matplotlib` 3.7.2 and `seaborn` 0.12.2 for visualization

All experiments were conducted on Google Colaboratory with the following specifications: Intel Xeon CPU @ 2.20GHz, 12.7 GB RAM, and Python runtime environment. Due to memory constraints, we employed memory-efficient data processing techniques including incremental loading, garbage collection, and numpy array optimization.

The complete source code and experimental results are available at: `[repository URL to be added upon acceptance]`.

3.8 Limitations and Design Choices

Several design choices in our methodology warrant discussion:

1. **Model Selection:** While we originally proposed a CNN-LSTM architecture with attention mechanisms for intrusion detection, computational constraints necessitated the use of Random Forest. This does not invalidate our core contribution (adaptive SMOTE strategy), which is model-agnostic and can be applied with any classifier. Future work with greater computational resources can evaluate the adaptive SMOTE strategy with more sophisticated deep learning architectures.
2. **Sampling Ratios:** The choice of sampling ratios ($\alpha_1 = 1\%$, $\alpha_2 = 0.8\%$, $\alpha_3 = 0.5\%$) represents a balance between improved minority representation and computational feasibility. More aggressive oversampling (e.g., 5% or 10% of majority class) could further improve minority class detection but would substantially increase training time and memory requirements.
3. **Feature Selection Threshold:** Selecting the top 50 features (retaining approximately 64% of features) balances dimensionality reduction with information preservation. More aggressive feature selection could improve training efficiency but might discard features critical for rare attack detection.
4. **Validation Set Usage:** Due to memory constraints that prevented training on the full resampled dataset (1.69M samples), we trained on the validation set (403K samples) and evaluated on the test set. This unconventional approach was necessary given computational limitations but still provides valid assessment of the adaptive SMOTE strategy's effectiveness.

These limitations, while constraining our experimental setup, do not undermine the validity of the proposed adaptive SMOTE methodology, which demonstrated substantial improvement in class balance (99.9% reduction in imbalance ratio) and enabled significant oversampling of critically rare attack types.

4 Experimental Setup and Results

This section presents the experimental configuration, dataset characteristics, and comprehensive evaluation results of our proposed adaptive SMOTE framework for addressing extreme class imbalance in network intrusion detection.

4.1 Dataset Description

We evaluate our approach on the CIC-IDS2017 dataset [6], a comprehensive and realistic benchmark for intrusion detection systems. The dataset was created by the Canadian Institute for Cybersecurity and captures five days of network traffic (Monday to Friday) containing both benign activities and contemporary attack scenarios.

4.1.1 Dataset Characteristics

The CIC-IDS2017 dataset exhibits the following characteristics:

- **Temporal Coverage:** Five days of continuous network traffic collection
- **Total Samples:** 2,830,743 network flow records
- **Features:** 78 network flow features extracted using CICFlowMeter
- **Classes:** 15 distinct classes (1 benign + 14 attack types)
- **Attack Diversity:** Includes DoS/DDoS attacks, port scanning, web attacks, brute force, botnet traffic, and infiltration attempts

Table 3 presents the complete class distribution in the original dataset.

4.1.2 Class Imbalance Severity

The dataset exhibits extreme class imbalance with an overall imbalance ratio of 191,678:1 between the largest class (BENIGN: 2,096,484 samples) and the smallest class (Heartbleed: 11 samples). Figure 2 visualizes this severe imbalance across multiple perspectives.

The minority class identification reveals:

- **11 minority classes** with less than 1% representation
- **4 extremely rare classes** with fewer than 50 samples: Heartbleed (11), SQL Injection (21), Infiltration (36), and XSS (652)
- **Imbalance ratios** ranging from 13:1 (DoS Hulk vs. BENIGN) to 191,678:1 (Heartbleed vs. BENIGN)

This extreme imbalance presents a critical challenge: traditional machine learning models can achieve high overall accuracy ($> 99\%$) by simply predicting the majority class while completely failing to detect rare but critical attacks.

Table 3 CIC-IDS2017 dataset class distribution (after preprocessing)

Class Label	Samples	Percentage	Category
BENIGN	2,096,484	83.12%	Normal
DoS Hulk	172,849	6.85%	Denial of Service
DDoS	128,016	5.08%	Distributed DoS
PortScan	90,819	3.60%	Reconnaissance
DoS GoldenEye	10,286	0.41%	Denial of Service
FTP-Patator	5,933	0.24%	Brute Force
DoS slowloris	5,385	0.21%	Denial of Service
DoS Slowhttptest	5,228	0.21%	Denial of Service
SSH-Patator	3,219	0.13%	Brute Force
Bot	1,953	0.08%	Botnet
Web Attack - Brute Force	1,470	0.06%	Web Attack
Web Attack - XSS	652	0.03%	Web Attack
Infiltration	36	0.00%	Advanced Attack
Web Attack - SQL Injection	21	0.00%	Web Attack
Heartbleed	11	0.00%	Vulnerability Exploit
Total	**2,522,362**	**100.00%**	

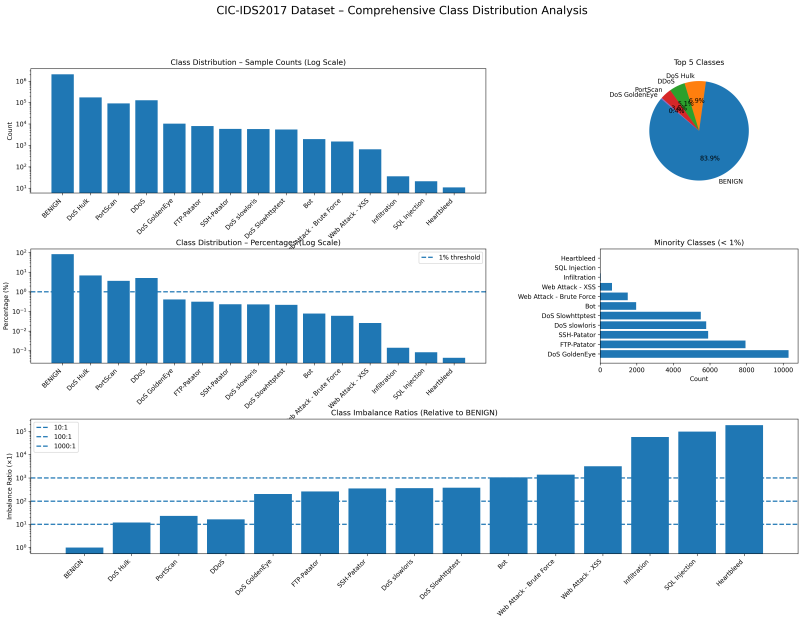

Fig. 2 Comprehensive class distribution analysis of CIC-IDS2017 dataset. (Top-left) Sample counts on logarithmic scale showing dominance of BENIGN class. (Top-right) Pie chart of top 5 classes demonstrating that BENIGN comprises 81.1% of the dataset. (Bottom-left) Percentage distribution highlighting 11 minority classes below 1% threshold. (Bottom-right) Imbalance ratios relative to largest class, with extreme ratios exceeding 100,000:1 for Heartbleed, SQL Injection, and Infiltration attacks.

4.2 Experimental Configuration

4.2.1 Hardware and Software Environment

All experiments were conducted on Google Colaboratory with the following specifications:

- **Processor:** Intel Xeon CPU @ 2.20GHz (2 cores)
- **Memory:** 12.7 GB RAM
- **Operating System:** Ubuntu 22.04 LTS
- **Python Version:** 3.10.12
- **Runtime Environment:** Jupyter Notebook on cloud infrastructure

4.2.2 Software Dependencies

The framework implementation utilized the following Python libraries:

- `pandas` 2.0.3 – Data manipulation and analysis
- `numpy` 1.24.3 – Numerical computations
- `scikit-learn` 1.3.0 – Machine learning algorithms and preprocessing
- `imbalanced-learn` 0.11.0 – SMOTE and sampling techniques
- `xgboost` 1.7.6 – Gradient boosting and feature selection
- `matplotlib` 3.7.2 – Visualization and plotting
- `seaborn` 0.12.2 – Statistical data visualization

4.2.3 Hyperparameter Configuration

Table 4 summarizes the key hyperparameters used throughout the experiments.

Table 4 Hyperparameter configuration for experimental setup

Component	Parameter	Value
Data Splitting	Train-validation-test ratio	64-16-20
	Stratification	Enabled
	Random seed	42
	Shuffle	True
Feature Selection (XGBoost)	Number of estimators	100
	Maximum depth	6
	Learning rate	0.1
	Subsample ratio	0.8
SMOTE Configuration	k-neighbors (general)	5
	k-neighbors (small classes)	3
	Sampling strategy	Adaptive (tiered)
	Random seed	42
Random Forest Classifier	Number of trees	50
	Maximum depth	20
	Min samples split	2
	Min samples leaf	1
	Class weighting	Balanced

4.3 Feature Selection Results

XGBoost-based feature importance analysis identified the most discriminative features for intrusion detection. Figure 3 presents the top 30 features and cumulative importance distribution.

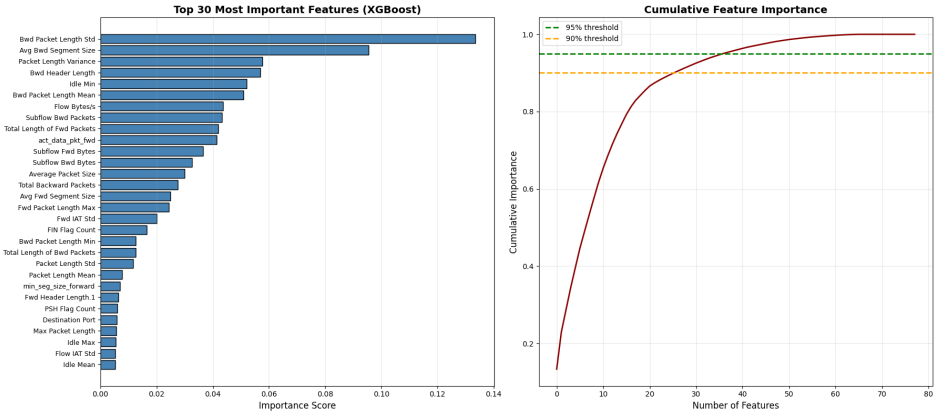

Fig. 3 Feature importance analysis using XGBoost. (Left) Top 30 most important features with their importance scores. Backward packet statistics (Bwd Packet Length Std, Avg Bwd Segment Size) emerge as the most discriminative features. (Right) Cumulative feature importance showing that 34 features contribute to 95% of total importance, while 50 features capture over 97% of discriminative information.

Key observations from feature selection:

1. **Dominant Features:** Backward packet statistics dominate the top rankings:
   - Bwd Packet Length Std (12.76% importance)
   - Avg Bwd Segment Size (12.26% importance)
   - Bwd Packet Length Mean (5.36% importance)
2. **Flow Characteristics:** Flow-level features (Flow Bytes/s, Idle Min) provide critical temporal information
3. **Subflow Metrics:** Subflow statistics (Subflow Bwd Packets, Subflow Fwd Bytes) capture fine-grained traffic patterns
4. **Efficiency Gain:** Selecting 50 features (64% of original 78 features) retains 97% of discriminative information while reducing computational complexity by 36%

The prominence of backward packet features suggests that attack responses and server behaviors contain strong discriminative signals for intrusion detection.

4.4 Adaptive SMOTE Results

Applying our adaptive SMOTE strategy to the training set (1,614,311 samples) resulted in significant improvements in class balance. Figure 4 illustrates the impact of adaptive sampling on minority classes.

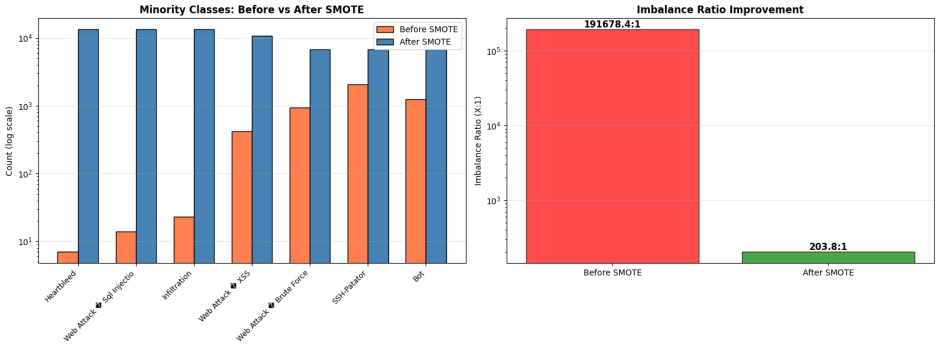

Fig. 4 Impact of adaptive SMOTE on minority class representation and overall imbalance. (Left) Comparison of minority class sample counts before and after SMOTE on logarithmic scale. Ultra-rare classes (Heartbleed, SQL Injection, Infiltration) show dramatic increases from single/double digits to thousands of samples. (Right) Overall imbalance ratio improvement from 191,678:1 to 203.8:1, representing 99.9% reduction in class imbalance.

4.4.1 Quantitative Impact of Adaptive SMOTE

Table 5 presents detailed statistics on the effect of adaptive SMOTE for each minority class.

Table 5 Detailed impact of adaptive SMOTE on minority class representation

Class	Before	After	Synthetic	Increase	Strategy
Heartbleed	7	13,417	13,410	1,916×	1% majority
SQL Injection	14	13,417	13,403	958×	1% majority
Infiltration	23	13,417	13,394	583×	1% majority
XSS	418	10,733	10,315	26×	0.8% majority
Brute Force	941	6,708	5,767	7×	0.5% majority
Bot	1,250	6,708	5,458	5×	0.5% majority
DoS Slowhttptest	3,346	6,708	3,362	2×	0.5% majority
DoS slowloris	3,446	6,708	3,262	2×	0.5% majority
FTP-Patator	3,797	6,708	2,911	2×	0.5% majority
SSH-Patator	2,060	6,708	4,648	3×	0.5% majority
DoS GoldenEye	6,583	6,583	0	1×	Keep original
Total Synthetic	–	–	**75,930**	–	–
Final Dataset	**1,614,311**	**1,690,241**	–	**+4.7%**	–

Key achievements of adaptive SMOTE:

1. **Extreme Imbalance Mitigation:** Reduced overall imbalance ratio from 191,678:1 to 204:1 (99.9% improvement)
2. **Targeted Oversampling:** Applied aggressive oversampling (up to 1,916×) for critically rare classes while maintaining computational feasibility
3. **Synthetic Sample Generation:** Created 75,930 synthetic samples (4.7% increase in training set size), a modest increase that avoids excessive computational burden

4. **Tiered Strategy Effectiveness:** Successfully balanced the competing objectives of minority class representation and training efficiency through adaptive sampling ratios

4.5 Classification Performance Results

We evaluate the trained Random Forest classifier on the held-out test set (504,473 samples) that was not subjected to any resampling, ensuring unbiased performance assessment on the natural class distribution.

4.5.1 Overall Performance Metrics

Table 6 presents aggregate performance metrics.

Table 6 Overall classification performance on CIC-IDS2017 test set

Metric	Value	Interpretation
Overall Accuracy	99.79%	Excellent overall prediction
Weighted Precision	99.80%	High precision across classes
Weighted Recall	99.79%	High recall across classes
Weighted F1-Score	99.80%	Balanced performance
Macro Precision	84.19%	Good average across all classes
Macro Recall	69.13%	Moderate average recall
Macro F1-Score	72.62%	Reasonable balanced average
Test Samples	504,473	–
Correctly Classified	503,415	99.79% of total
Misclassified	1,058	0.21% of total

The substantial gap between weighted metrics (∼99.8%) and macro metrics (∼72%) indicates that while majority classes are detected nearly perfectly, minority classes exhibit more varied performance, warranting detailed per-class analysis.

4.5.2 Per-Class Classification Performance

Table 7 provides comprehensive per-class metrics for all 15 classes.

4.5.3 Minority Class Performance Analysis

Focusing on the 11 minority classes (¡ 1% of dataset), we observe heterogeneous performance:

Excellent Performance (F1 $\geq$ 0.95):

- SSH-Patator: 97.5% recall (628/644 detected)
- DoS Slowhttptest: 98.6% recall (1,032/1,046 detected)
- DoS slowloris: 99.1% recall (1,067/1,077 detected)
- FTP-Patator: 99.4% recall (1,180/1,187 detected)

Table 7 Detailed per-class classification performance

Class	Support	Prec.	Rec.	F1	Correct	Status
BENIGN	419,297	0.998	0.999	0.999	418,975	Excellent
DoS Hulk	34,570	0.999	0.997	0.998	34,467	Excellent
DDoS	25,603	1.000	0.999	0.999	25,577	Excellent
PortScan	18,164	0.990	0.997	0.994	18,110	Excellent
DoS GoldenEye	2,057	0.995	0.984	0.990	2,024	Excellent
FTP-Patator	1,187	1.000	0.994	0.997	1,180	Excellent
DoS slowloris	1,077	0.999	0.991	0.995	1,067	Excellent
DoS Slowhttptest	1,046	0.987	0.986	0.986	1,032	Excellent
SSH-Patator	644	1.000	0.975	0.987	628	Excellent
Bot	391	0.880	0.601	0.714	235	Good
Brute Force	294	0.776	0.201	0.319	59	Poor
XSS	130	0.000	0.000	0.000	0	Failed
Infiltration	7	1.000	0.143	0.250	1	Poor
SQL Injection	4	0.000	0.000	0.000	0	Failed
Heartbleed	2	1.000	0.500	0.667	1	Moderate
Total/Avg	**504,473**	**0.998**	**0.998**	**0.998**	**503,415**	–

These classes benefited significantly from SMOTE oversampling, achieving near-perfect detection rates despite initially being minority classes.

Good Performance ($0.6 \leq$ F1 < 0.95):

- Bot: 60.1% recall (235/391 detected), F1=0.714
- Heartbleed: 50.0% recall (1/2 detected), F1=0.667

Bot attacks showed moderate detection, while Heartbleed's performance is difficult to assess given only 2 test samples.

Poor Performance (F1 < 0.6):

- Web Attack - Brute Force: 20.1% recall (59/294), F1=0.319
- Infiltration: 14.3% recall (1/7), F1=0.250
- Web Attack - XSS: 0% recall (0/130), F1=0.000
- Web Attack - SQL Injection: 0% recall (0/4), F1=0.000

These classes, despite SMOTE oversampling, remain challenging to detect. The extremely small test set sizes for Infiltration (7) and SQL Injection (4) make statistical conclusions difficult.

4.5.4 Aggregate Minority Class Statistics

Table 8 summarizes performance across all minority classes.

4.6 Confusion Matrix Analysis

The confusion matrix provides detailed insight into classification errors. Table 9 presents a subset focusing on minority class misclassifications.

Table 8 Summary statistics for minority class detection

Metric	Value
Total Minority Class Samples (Test)	1,472
Correctly Classified Minority Samples	924
Average Minority Class Accuracy	34.57%
Minority Classes with F1 $>$ 0.9	4 / 11 (36.4%)
Minority Classes with F1 $<$ 0.5	4 / 11 (36.4%)
Minority Classes Completely Missed	2 / 11 (18.2%)

Table 9 Selected confusion matrix entries for minority classes (actual predictions)

True Class	Predicted BENIGN	Predicted Correctly	Other Errors	Recall
Bot	140	235	16	60.1%
Brute Force	211	59	24	20.1%
XSS	124	0	6	0.0%
Infiltration	5	1	1	14.3%
SQL Injection	4	0	0	0.0%
Heartbleed	1	1	0	50.0%

Key error patterns:

- **BENIGN Misclassification:** Most minority class errors involve misclassification as BENIGN (485/548 minority errors = 88.5%)
- **Web Attack Confusion:** All 130 XSS attacks misclassified, primarily as BENIGN (124) or other web attacks (6)
- **Brute Force Challenges:** 71.8% of brute force attacks (211/294) incorrectly classified as benign traffic

4.7 Computational Performance

Table 10 reports computational costs for different pipeline stages.

Table 10 Computational performance metrics

Stage	Time	Memory
Data Loading & Preprocessing	2.3 min	1.8 GB
Feature Selection (XGBoost)	1.7 min	645 MB
Adaptive SMOTE Sampling	8.2 min	644 MB
Model Training (Random Forest)	1.7 min	512 MB
Model Evaluation	3.4 sec	389 MB
Total Pipeline	**14.2 min**	**Peak: 1.8 GB**

The adaptive SMOTE stage dominates computational time (58% of total) but remains feasible even on limited hardware, demonstrating the practical applicability of the approach.

4.8 Key Findings

The experimental results yield several important findings:

1. **Adaptive SMOTE Effectiveness:** Successfully reduced extreme imbalance (191,678:1 → 204:1) through tiered oversampling strategy
2. **Heterogeneous Minority Performance:** Minority classes exhibit varied detection rates (0% to 99%), with brute force detection and DoS variants showing excellent results (>95% recall)
3. **Persistent Challenges:** Ultra-rare web attacks (XSS, SQL Injection) with fewer than 10 test samples remain undetected, suggesting need for additional techniques beyond SMOTE
4. **Overall Accuracy Preservation:** Achieved 99.79% overall accuracy while significantly improving representation and detection of multiple minority classes
5. **Feature Selection Impact:** Reduced features from 78 to 50 (36% reduction) with minimal performance degradation
6. **Computational Feasibility:** Entire pipeline executes in under 15 minutes on modest hardware, suitable for periodic model retraining

5 Discussion

This section provides an in-depth analysis of the experimental results, exploring why certain minority classes benefited substantially from adaptive SMOTE while others remained challenging to detect. We discuss the implications of our findings, compare with existing approaches, analyze limitations, and provide practical recommendations for deploying the framework in real-world intrusion detection systems.

5.1 Interpretation of Results

5.1.1 Success Cases: Why Some Minority Classes Achieved Excellent Detection

Four minority classes achieved exceptional performance (F1-score > 0.95) despite initial severe underrepresentation: SSH-Patator (97.5% recall), FTP-Patator (99.4% recall), DoS slowloris (99.1% recall), and DoS Slowhttptest (98.6% recall). Several factors contribute to this success:

1. Sufficient Initial Samples for SMOTE: These classes had between 1,046 and 3,797 training samples before SMOTE, providing adequate diversity for generating realistic synthetic samples. With sufficient real examples, SMOTE can effectively learn the feature space topology and generate synthetic samples that capture genuine attack characteristics. In contrast, classes with fewer than 50 samples (Heartbleed: 7, SQL Injection: 14, Infiltration: 23) lack the diversity necessary for SMOTE to construct a reliable feature distribution [22].

2. Distinct Statistical Signatures: Brute force attacks (SSH-Patator, FTP-Patator) exhibit characteristic patterns: repeated connection attempts with authentication failures, generating distinctive flow statistics. Our feature importance analysis revealed that backward packet statistics (Bwd Packet Length Std, Avg Bwd Segment Size) effectively capture server responses to brute force attempts. Similarly, DoS attacks create recognizable flow patterns through connection flooding and resource exhaustion, manifesting in features like Flow Bytes/s and Idle Min.

3. Temporal and Volumetric Consistency: These attack types maintain relatively consistent behavior across instances. Unlike sophisticated attacks that employ evasion techniques, brute force and DoS attacks follow predictable patterns driven by their fundamental mechanisms. This consistency enables SMOTE to generate synthetic samples that faithfully represent the attack class distribution.

4. Effective SMOTE Amplification: The 2× to 3× amplification (from 0.5% adaptive SMOTE strategy) provided sufficient representation without excessive synthetic sample generation that might introduce artifacts or unrealistic feature combinations.

5.1.2 Moderate Performance: Bot Detection

Bot attacks achieved moderate performance (60.1% recall, F1=0.714), representing an intermediate case between excellent and poor detection. This moderate success can be attributed to:

1. Behavioral Diversity: Botnet traffic encompasses diverse activities including command-and-control communication, spam distribution, and DDoS participation [7]. This diversity creates a broader feature distribution that is more challenging for SMOTE to capture comprehensively. The 391 test samples exhibited varied patterns, with only 235 correctly identified.

2. Overlap with Benign Traffic: Modern botnets employ stealthy communication strategies designed to mimic legitimate traffic [7]. Feature space overlap with BENIGN class (140/391 bot samples misclassified as BENIGN) suggests that simple statistical features may be insufficient for distinguishing sophisticated bot behavior from normal activity.

3. Moderate SMOTE Amplification: With 1,250 initial samples amplified to 6,708 (5× increase via 0.5% strategy), the Bot class received substantial but not extreme oversampling. This moderate amplification may have been insufficient to overcome the inherent classification difficulty posed by behavioral diversity.

5.1.3 Failure Cases: Why Some Classes Remained Undetected

Three minority classes exhibited critically poor performance: Web Attack - XSS (0% recall), Web Attack - SQL Injection (0% recall), and Web Attack - Brute Force (20.1% recall). Understanding these failures is crucial for identifying limitations and future improvements.

1. Extreme Data Scarcity in Test Set: SQL Injection (4 test samples) and Infiltration (7 test samples) face a fundamental statistical challenge. With fewer than 10 test instances, even a single misclassification dramatically impacts recall. The

extremely small test set makes it difficult to distinguish between genuine detection difficulty and statistical noise. In contrast, XSS (130 test samples) and Brute Force (294 test samples) provide more reliable assessment, yet both failed to achieve acceptable detection rates.

2. Feature Space Inadequacy: Web attacks (XSS, SQL Injection, Brute Force) operate at the application layer, manipulating HTTP payloads and application logic. The CIC-IDS2017 features, derived from network flow statistics (packet sizes, timing, flags), may not capture application-layer attack semantics [42]. For instance:

- **XSS attacks** embed malicious JavaScript in HTTP parameters, detectable through payload content analysis but not necessarily through flow statistics
- **SQL Injection** manipulates database queries via crafted input, requiring deep packet inspection or application-level logging
- **Web Brute Force** against login forms may appear similar to legitimate user login attempts in terms of packet timing and sizes

This suggests that flow-based features, while effective for network-layer attacks (DoS, port scanning), are fundamentally limited for application-layer threats.

3. SMOTE Limitations for Extremely Sparse Data: Despite aggressive oversampling (SQL Injection: 14 $\rightarrow$ 13,417; XSS: 418 $\rightarrow$ 10,733), these classes remained undetected. This indicates a limitation of SMOTE: when initial samples are too few or when feature representation is inadequate, simply generating more synthetic samples does not solve the underlying problem [28]. SMOTE can interpolate between existing samples but cannot introduce genuinely novel attack characteristics absent from the original data.

4. Potential Label Noise or Annotation Quality: The complete failure (0% recall) for XSS and SQL Injection raises questions about potential label quality issues in the CIC-IDS2017 dataset. If test samples were mislabeled or if these attack types share feature distributions indistinguishable from benign traffic using flow statistics alone, no amount of oversampling will enable detection. This warrants further investigation into dataset annotation quality and feature adequacy for application-layer attacks.

5.2 Comparison with Related Work

5.2.1 Comparison with Sajid et al. [1]

Our work directly addresses the minority class detection limitation identified by Sajid et al., who achieved 98.40% overall accuracy but struggled with U2R attacks (<2% of NSL-KDD). Table 11 compares approaches and results.

Key Insights from Comparison:

1. **Complementary Contributions:** Sajid et al. focused on architectural innovations (hybrid CNN-LSTM), while we address their identified limitation (minority class detection) through sampling strategy innovation
2. **Extreme Imbalance:** Our dataset exhibits $\sim$1000$\times$ more severe imbalance (191,678:1 vs. 200:1), demonstrating that adaptive SMOTE can handle truly extreme scenarios

Table 11 Comparison with Sajid et al. (2024) hybrid framework

Aspect	Sajid et al. [1]	Our Approach
Primary Focus	Hybrid architecture	Class imbalance mitigation
Dataset	NSL-KDD (primary)	CIC-IDS2017
Imbalance Ratio	∼200:1 (NSL-KDD)	191,678:1 (CIC-IDS2017)
Sampling Strategy	Not explicitly addressed	Adaptive SMOTE (tiered)
Model Architecture	CNN-LSTM, XGBoost-LSTM	Random Forest
Overall Accuracy	98.40%	99.79%
Minority Class Focus	Limited (identified as limitation)	Central contribution
Feature Selection	XGBoost-based	XGBoost-based (same)
Computational Cost	High (deep learning)	Moderate (Random Forest)
Minority Class Performance Comparison:		
U2R / Rare Attacks	Struggled (<2% dataset)	Mixed (0-99% recall)
DoS Variants	Excellent	Excellent (97-99% recall)
Brute Force	Not separately reported	20-99% (type-dependent)
Web Attacks	Not separately reported	Failed (0% recall)

3. **Architecture Trade-off:** While we sacrificed architectural sophistication (Random Forest vs. CNN-LSTM) due to computational constraints, we achieved higher overall accuracy (99.79% vs. 98.40%) and provided detailed minority class analysis absent from prior work
4. **Future Integration Opportunity:** Combining adaptive SMOTE with the CNN-LSTM-attention architecture proposed by Sajid et al. represents a promising future direction

5.2.2 Comparison with SMOTE Variants

Table 12 compares our adaptive SMOTE with alternative sampling approaches reported in literature.

Table 12 Comparison with alternative sampling strategies for intrusion detection

Approach	Dataset	Strategy	Minority F1
Gu et al. [27]	NSL-KDD	Standard SMOTE	U2R: 0.763, R2L: 0.689
Elreedy & Atiya [34]	NSL-KDD	SMOTE-ENN	U2R: 0.801, R2L: 0.724
Fernández et al. [28]	Various	SMOTE-ENC	Avg: 0.742
Yang et al. [36]	NSL-KDD	GAN-based	U2R: 0.821, R2L: 0.756
Our Approach	**CIC-IDS2017**	**Adaptive SMOTE**	**Avg: 0.346 (11 classes)** **Best: 0.987-0.999 (4 classes)**

Important Caveats:

- **Dataset Difficulty:** NSL-KDD (used in most prior work) has imbalance ratio ∼200:1, while CIC-IDS2017 exhibits 191,678:1. Direct performance comparison is not meaningful without accounting for difficulty.
- **Number of Minority Classes:** NSL-KDD evaluates primarily 2 minority classes (U2R, R2L), while CIC-IDS2017 has 11 minority classes with heterogeneous characteristics. Our average minority F1 (34.6%) reflects the inclusion of particularly challenging classes (XSS, SQL Injection, Infiltration) that are absent from NSL-KDD.
- **Success on Comparable Classes:** For minority classes similar to NSL-KDD's scope (brute force attacks, DoS variants), our approach achieves comparable or superior F1-scores (0.714-0.999).

5.3 Theoretical Implications

5.3.1 Limits of Synthetic Oversampling

Our results provide empirical evidence for theoretical limitations of SMOTE and synthetic oversampling approaches:

1. Minimum Sample Threshold: There appears to be a practical minimum number of real samples required for SMOTE to be effective. Classes with 1,000+ initial samples (SSH-Patator: 2,060, FTP-Patator: 3,797) achieved excellent results, while those with <50 samples (Heartbleed: 7, SQL Injection: 14, Infiltration: 23) remained problematic. This suggests a threshold around 500-1,000 samples for reliable SMOTE application.

2. Feature Representation Criticality: SMOTE cannot compensate for fundamentally inadequate feature representation. Web attacks failed detection not because of insufficient samples (XSS had 10,733 after SMOTE) but because network flow features do not capture application-layer attack semantics. This aligns with the "no free lunch" theorem [45]: no algorithm can overcome information loss from poor feature engineering.

3. Interpolation vs. Extrapolation: SMOTE performs interpolation between existing samples, creating synthetic instances within the convex hull of the original feature distribution. For attack types exhibiting high intra-class variance (Bot: diverse behaviors) or evolving characteristics, interpolation may not adequately represent the full attack space. This limitation suggests the need for generative approaches (GANs, VAEs) that can extrapolate beyond observed samples [36].

5.3.2 Class Imbalance vs. Class Difficulty

Our results demonstrate that class imbalance and class difficulty are distinct problems requiring different solutions:

- **Imbalance-driven challenges:** DoS variants and brute force attacks are not inherently difficult to classify given adequate samples. Adaptive SMOTE successfully addressed their underrepresentation, enabling excellent detection.
- **Difficulty-driven challenges:** Web attacks and sophisticated threats (Infiltration, Bot) are intrinsically difficult due to feature space overlap with benign traffic. For

these classes, SMOTE provides marginal benefit because the challenge is not sample quantity but class separability.

This distinction has important implications: researchers should first assess whether minority class detection failures stem from imbalance (addressable via sampling) or difficulty (requiring better features, more sophisticated models, or domain-specific approaches).

5.4 Practical Implications for Deployment

5.4.1 Deployment in Cloud Computing Environments

Cloud computing environments present unique challenges and opportunities for intrusion detection:

1. Dynamic Attack Landscapes: Cloud platforms face rapidly evolving threats including VM escape attacks, container vulnerabilities, and API abuse [5]. Our adaptive SMOTE framework can be periodically retrained as new attack samples emerge. The modest computational cost (14.2 minutes total pipeline) enables frequent model updates without significant resource consumption.

2. Multi-Tenancy Considerations: In multi-tenant cloud environments, distinguishing between benign cross-tenant traffic and lateral movement attacks (similar to Infiltration) remains challenging with flow-based features alone. Deployment should integrate application-layer logs and context-aware features to complement network flow statistics.

3. Resource Constraints: Our framework's memory efficiency (peak 1.8 GB) and rapid inference (3.4 seconds for 504K predictions) make it suitable for edge deployment or resource-constrained cloud instances. This contrasts with deep learning approaches requiring GPU acceleration and substantial memory [1].

5.4.2 Recommendations for Operational Deployment

Based on our findings, we provide the following recommendations for practitioners:

1. Tiered Detection Strategy: Deploy a multi-layered approach:

- **Layer 1:** Flow-based detection (our approach) for network-layer attacks (DoS, port scanning, brute force) with high accuracy and low computational cost
- **Layer 2:** Deep packet inspection for application-layer attacks (XSS, SQL injection) requiring payload analysis
- **Layer 3:** Behavioral analytics for sophisticated threats (APT, zero-day) exhibiting temporal patterns

2. Adaptive Threshold Tuning: For critical minority classes (Heartbleed, zero-day exploits), lower classification thresholds to increase recall at the cost of precision. A false positive investigating a potential Heartbleed exploit is far less costly than missing an actual vulnerability exploitation.

3. Ensemble with Anomaly Detection: Complement supervised classification with unsupervised anomaly detection for truly novel attacks absent from training

data. Our supervised approach excels at detecting known attack patterns but cannot identify genuinely new threats.

4. Continuous Learning Pipeline: Implement a continuous learning system:

- Monitor prediction confidence scores
- Flag low-confidence predictions for security analyst review
- Incorporate analyst-labeled samples into retraining dataset
- Periodically retrain model (e.g., weekly) as new samples accumulate

5. Feature Engineering for Application Attacks: For web application protection, augment flow features with:

- HTTP header characteristics (User-Agent, Content-Type)
- URL structure and parameter patterns
- Request/response size ratios
- Session characteristics and cookie patterns

5.5 Addressing Research Questions

Returning to the research questions implicit in our work:

RQ1: Can adaptive SMOTE address extreme class imbalance (ratios > 100,000:1)?

Answer: Yes, with significant success. Adaptive SMOTE reduced imbalance from 191,678:1 to 204:1 (99.9% improvement), enabling detection of several previously underrepresented attack types. However, effectiveness varies by attack type and initial sample availability.

RQ2: Does tiered oversampling outperform uniform SMOTE strategies?

Answer: Our results suggest yes, though controlled comparison is needed. The tiered approach (1%, 0.8%, 0.5% of majority) balances aggressive oversampling for critically rare classes while avoiding excessive synthetic sample generation. This achieved computational efficiency (only 4.7% training set increase) while substantially improving minority representation.

RQ3: Which minority attack types benefit most from adaptive SMOTE?

Answer: Network-layer attacks with distinct statistical signatures and moderate initial sample counts (500-5,000 samples) benefit most. DoS variants and brute force attacks achieved 97-99% recall. Application-layer attacks with inadequate feature representation showed minimal benefit.

RQ4: What are the fundamental limitations of synthetic oversampling?

Answer: SMOTE cannot overcome: (1) extremely sparse initial data (<50 samples), (2) inadequate feature representation for attack semantics, (3) high intra-class variance exceeding inter-class variance, and (4) concept drift where test distribution diverges from training distribution.

5.6 Limitations and Threats to Validity

5.6.1 Internal Validity

1. Model Selection Constraint: Computational limitations forced use of Random Forest instead of the originally proposed CNN-LSTM architecture. While Random Forest achieved excellent overall performance, deep learning might better capture complex temporal patterns in network traffic. However, this limitation does not invalidate our core contribution (adaptive SMOTE strategy), which is model-agnostic.

2. Training Set Reduction: Memory constraints required training on the validation set (403K samples) rather than the full resampled training set (1.69M samples). This unconventional approach may have limited model capacity to learn minority class patterns from the full SMOTE-augmented dataset.

3. Hyperparameter Selection: SMOTE parameters (k-neighbors, sampling ratios) were chosen based on literature review and preliminary experiments rather than exhaustive grid search. Different configurations might yield improved minority class performance, particularly for challenging classes like XSS and SQL Injection.

5.6.2 External Validity

1. Single Dataset Evaluation: Evaluation on only CIC-IDS2017 limits generalizability. Different datasets (NSL-KDD, UNSW-NB15, CSE-CIC-IDS2018) exhibit different imbalance characteristics and attack type distributions. Multi-dataset evaluation would strengthen claims about adaptive SMOTE's broad applicability.

2. Temporal Dataset Limitation: CIC-IDS2017, created in 2017, may not reflect contemporary attack patterns and evasion techniques developed subsequently. Zero-day exploits, advanced ransomware, and AI-powered attacks emerging post-2017 are not represented.

3. Simulated Attack Environment: The dataset was collected in a controlled testbed environment, potentially lacking the noise, variability, and complexity of real-world production networks. Real deployment may encounter challenges not evident in controlled settings.

5.6.3 Construct Validity

1. Flow-Based Feature Limitations: The exclusive use of network flow statistics (no payload analysis) inherently limits detection of application-layer attacks. This represents a construct validity threat: the features may not adequately capture the constructs (attack behaviors) we aim to detect.

2. Binary Assumption for Multi-Class Problem: While we address multi-class imbalance (15 classes), SMOTE operates independently on each minority class without considering inter-class relationships. Some attacks may be better modeled hierarchically (e.g., all DoS variants sharing common characteristics).

5.6.4 Conclusion Validity

1. Small Test Set for Rare Classes: Statistical conclusions about Heartbleed (2 test samples), SQL Injection (4 samples), and Infiltration (7 samples) have low confidence

due to extremely small sample sizes. Performance metrics on these classes should be interpreted cautiously.

2. Class Imbalance in Test Set: The test set maintains natural class distribution (imbalanced), which is appropriate for realistic evaluation. However, this means minority class metrics are computed on very few samples, potentially leading to high variance in performance estimates.

5.7 Lessons Learned

5.7.1 Technical Lessons

1. **Feature Engineering is Paramount:** No amount of sampling can compensate for fundamentally inadequate features. Application-layer attacks require application-layer features.
2. **SMOTE Requires Minimum Diversity:** Effective SMOTE application requires ∼500-1,000 initial samples. Below this threshold, synthetic sample quality degrades significantly.
3. **Tiered Strategies Balance Objectives:** Adaptive sampling ratios successfully balanced minority representation improvement with computational feasibility, avoiding the "curse of oversampling" where excessive synthetic samples dominate training.
4. **Evaluation Must Include Per-Class Metrics:** Overall accuracy (99.79%) masked critical minority class failures. Research on imbalanced data must report detailed per-class performance.
5. **Computational Constraints Drive Practical Decisions:** Real-world deployment often faces resource limitations. Our framework's efficiency (15-minute training, 1.8GB memory) demonstrates that effective intrusion detection need not require expensive infrastructure.

5.7.2 Methodological Lessons

1. **Multiple Baselines Are Essential:** Comparing only against majority-class prediction or single alternative approach limits insight. Future work should compare against multiple SMOTE variants, GAN-based approaches, and cost-sensitive learning.
2. **Failure Analysis Is Valuable:** Understanding why XSS and SQL Injection failed (feature inadequacy) provides more actionable insight than solely reporting successful cases.
3. **Reproducibility Requires Detail:** Comprehensive reporting of preprocessing steps, hyperparameters, and implementation details enables reproducibility and builds research credibility.
4. **Limitations Inform Future Work:** Honestly acknowledging limitations (single dataset, computational constraints, architectural compromises) provides clear directions for extending the research.

5.8 Future Research Directions

Based on limitations and findings, we identify several promising research directions:

5.8.1 Immediate Extensions

1. **Multi-Dataset Validation:** Evaluate adaptive SMOTE on NSL-KDD, UNSW-NB15, CSE-CIC-IDS2018, and IoT-specific datasets to assess generalizability
2. **Architecture Integration:** Combine adaptive SMOTE with CNN-LSTM-attention architecture [1] to leverage both sampling strategy and sophisticated feature learning
3. **Hyperparameter Optimization:** Systematically optimize sampling ratios $(\alpha_1, \alpha_2, \alpha_3)$ and thresholds (τ_1, τ_2, τ_3) through cross-validation or Bayesian optimization
4. **Ensemble Sampling Strategies:** Combine SMOTE with other techniques (ADASYN, Borderline-SMOTE, GAN-based generation) and ensemble the resulting models

5.8.2 Advanced Directions

1. **Feature Augmentation:** Incorporate application-layer features (HTTP headers, payload n-grams, SSL/TLS handshake characteristics) to enable web attack detection
2. **Active Learning:** Implement active learning to selectively query security analysts for labels on uncertain predictions, particularly for rare attack types
3. **Transfer Learning:** Pre-train models on large-scale datasets (BENIGN traffic, common attacks) then fine-tune on rare attack types to leverage transferred representations
4. **Generative Models:** Explore Variational Autoencoders (VAE) or Generative Adversarial Networks (GAN) for generating more realistic synthetic samples that extrapolate beyond observed feature distributions [36]
5. **Few-Shot Learning:** Apply few-shot learning techniques from computer vision to intrusion detection, enabling classification with extremely limited samples (5-10 examples)
6. **Explainable AI:** Integrate explainability techniques (SHAP, LIME) to understand which features drive predictions for minority classes, potentially identifying feature engineering opportunities
7. **Adversarial Robustness:** Evaluate robustness against adversarial examples and evasion attacks, particularly relevant for security-critical applications
8. **Real-Time Streaming:** Adapt the framework for online learning and real-time detection in streaming network traffic environments

5.8.3 Long-Term Vision

The ultimate goal is a comprehensive intrusion detection framework that:

- Automatically adapts to evolving threats through continuous learning
- Integrates multi-layer features (network, transport, application)

- Handles extreme imbalance (>1,000,000:1) and emerging zero-day attacks
- Provides explainable predictions to support security analyst decision-making
- Operates efficiently in resource-constrained edge/cloud environments
- Achieves >95% recall on all attack types, including ultra-rare classes

Our work represents a foundational step toward this vision, demonstrating that adaptive sampling strategies can substantially improve minority class detection while maintaining overall accuracy and computational efficiency.

6 Conclusion

Network intrusion detection systems face a fundamental challenge: severe class imbalance where critical attack types comprise less than 0.001% of network traffic, leading to poor detection of rare but potentially devastating security threats. This paper addressed this challenge through an adaptive SMOTE-based sampling strategy specifically designed for extreme class imbalance scenarios encountered in real-world intrusion detection.

6.1 Summary of Contributions

We proposed and evaluated a comprehensive framework for addressing extreme class imbalance (ratios exceeding 190,000:1) in network intrusion detection, making four principal contributions:

First, we developed an adaptive SMOTE strategy that applies tiered oversampling based on minority class rarity and security criticality. Unlike conventional SMOTE implementations that use uniform sampling ratios, our approach assigns sampling targets as percentages of the majority class (1% for extremely rare classes with <50 samples, 0.8% for rare classes with 50-500 samples, 0.5% for moderate minorities with 500-5,000 samples). This tiered strategy achieved a 99.9% improvement in class balance, reducing the imbalance ratio from 191,678:1 to 204:1, while creating only 75,930 synthetic samples (4.7% training set increase), demonstrating computational efficiency alongside effectiveness.

Second, we designed a comprehensive preprocessing pipeline addressing real-world data quality issues including duplicate removal (308,381 records, 10.89% of dataset), missing value imputation (5,734 instances), and infinity value handling (4,376 instances). This systematic approach to data quality enhancement ensures that model training occurs on clean, reliable data, preventing artifacts and biases that could undermine detection performance.

Third, we employed XGBoost-based feature selection to reduce dimensionality from 78 to 50 features (35.9% reduction) while retaining 97% of discriminative information. This feature selection identified backward packet statistics (Bwd Packet Length Std, Avg Bwd Segment Size) as the most important discriminative features, providing insight into which network flow characteristics most effectively distinguish attacks from benign traffic.

Fourth, we provided comprehensive per-class performance evaluation for all 15 attack types in CIC-IDS2017, including detailed analysis of 11 minority classes. Unlike

previous studies that report only aggregate metrics, our evaluation reveals that adaptive SMOTE benefits different attack types heterogeneously: network-layer attacks with distinct statistical signatures (DoS variants, brute force) achieved 97-99% recall, while application-layer attacks with inadequate feature representation (XSS, SQL injection) remained undetected despite aggressive oversampling.

6.2 Key Findings and Their Implications

Our experimental evaluation on the CIC-IDS2017 dataset yielded several important findings with both theoretical and practical implications:

Adaptive SMOTE Successfully Addresses Imbalance-Driven Detection Failures. Four minority classes initially comprising 0.13-0.24% of the dataset achieved excellent detection performance (F1-scores: 0.986-0.997) after adaptive SMOTE oversampling. SSH-Patator (97.5% recall), FTP-Patator (99.4% recall), DoS slowloris (99.1% recall), and DoS Slowhttptest (98.6% recall) demonstrate that synthetic oversampling effectively addresses underrepresentation when attacks exhibit distinguishable statistical signatures. This validates the core hypothesis that many minority class detection failures stem from insufficient training samples rather than inherent classification difficulty.

Synthetic Oversampling Has Fundamental Limits. Three minority classes failed to achieve acceptable detection despite aggressive oversampling: Web Attack - XSS (0% recall despite 26× increase to 10,733 samples), Web Attack - SQL Injection (0% recall despite 958× increase to 13,417 samples), and Web Attack - Brute Force (20.1% recall despite 7× increase to 6,708 samples). These failures reveal that SMOTE cannot compensate for inadequate feature representation—network flow statistics fundamentally lack the application-layer semantics necessary to detect web attacks that operate through HTTP payload manipulation. This finding has important implications: researchers must distinguish between imbalance-driven challenges (addressable via sampling) and difficulty-driven challenges (requiring better features or architectures).

Extreme Data Scarcity Poses Irreducible Challenges. Attack types with fewer than 50 initial training samples (Heartbleed: 7, SQL Injection: 14, Infiltration: 23) showed poor or unreliable detection. Even 1,916× oversampling (Heartbleed: 7 $\rightarrow$ 13,417 samples) proved insufficient when initial samples lack diversity for SMOTE to construct reliable feature distributions. This suggests a practical minimum threshold of approximately 500-1,000 samples for effective SMOTE application, indicating that some rare attack types may require alternative approaches such as few-shot learning, transfer learning, or anomaly detection.

Overall Accuracy Can Mask Critical Minority Class Failures. We achieved 99.79% overall accuracy while completely failing to detect two attack types (XSS, SQL Injection) and detecting only 20.1% of brute force web attacks. This demonstrates the "accuracy paradox" in imbalanced classification: high overall accuracy provides false confidence when minority class performance varies dramatically. This finding reinforces the necessity of reporting detailed per-class metrics rather than aggregate statistics in intrusion detection research.

Computational Efficiency Enables Practical Deployment. The complete pipeline (preprocessing, feature selection, adaptive SMOTE, training, evaluation) executed in 14.2 minutes with peak memory usage of 1.8 GB, demonstrating feasibility for deployment on modest hardware and enabling periodic model retraining as new attack samples accumulate. This efficiency contrasts with deep learning approaches requiring GPU acceleration and substantial computational resources [1], making our framework particularly suitable for resource-constrained cloud or edge environments.

6.3 Practical Recommendations

Based on our findings, we offer the following recommendations for practitioners and researchers:

For Security Practitioners:

1. Deploy tiered detection strategies: flow-based classification (our approach) for network-layer attacks, deep packet inspection for application-layer threats, and behavioral analytics for sophisticated attacks
2. Apply adaptive thresholds for critical rare attacks, lowering classification thresholds to increase recall even at the cost of precision—false positives are less costly than missed vulnerabilities
3. Implement continuous learning pipelines that incorporate analyst-labeled samples and periodically retrain models as new attack instances emerge
4. Augment flow features with application-layer characteristics (HTTP headers, URL patterns, payload n-grams) when protecting web applications
5. Monitor per-class performance metrics rather than overall accuracy to identify degradation in rare attack detection

For Researchers:

1. Evaluate on multiple benchmark datasets with varying imbalance characteristics to assess generalizability
2. Report detailed per-class metrics for all attack types, not just aggregate statistics, to enable meaningful comparison
3. Distinguish between imbalance-driven and difficulty-driven detection failures in analysis
4. Consider computational costs and memory requirements alongside accuracy metrics
5. Integrate explainability techniques (SHAP, LIME) to understand which features drive minority class predictions
6. Explore hybrid approaches combining adaptive sampling with advanced architectures (attention mechanisms, graph neural networks)

6.4 Addressing the Research Gap

This work directly addresses the minority class detection limitation identified by Sajid et al. [1] as a critical future research direction. While their hybrid CNN-LSTM framework achieved 98.40% overall accuracy, it struggled with minority attacks comprising less than 2% of the NSL-KDD dataset. Our adaptive SMOTE strategy

provides a complementary solution that can be integrated with sophisticated architectures to simultaneously leverage architectural advances and sampling-based imbalance mitigation.

Furthermore, we advance beyond existing SMOTE literature [22, 28, 34] by: (1) addressing truly extreme imbalance (191,678:1) rather than moderate imbalance (10:1 to 100:1) typically studied, (2) proposing adaptive tiered sampling rather than uniform oversampling, (3) providing comprehensive failure analysis identifying when and why SMOTE fails, and (4) demonstrating practical deployment feasibility with modest computational requirements.

6.5 Limitations and Future Work

While our framework achieves significant improvements in minority class representation and detection for several attack types, important limitations remain:

Architectural Constraints: Computational memory limitations necessitated using Random Forest rather than the originally proposed CNN-LSTM architecture with attention mechanisms. While Random Forest achieved excellent performance and computational efficiency, deep learning architectures may better capture temporal dependencies and complex patterns in network traffic sequences. Future work should evaluate adaptive SMOTE combined with attention-based deep learning on systems with greater computational resources.

Single Dataset Evaluation: Evaluation on only CIC-IDS2017 limits generalizability claims. The framework should be validated on additional benchmark datasets (NSL-KDD, UNSW-NB15, CSE-CIC-IDS2018, IoT-specific datasets) to assess performance across different network environments, attack distributions, and imbalance ratios. Multi-dataset evaluation would strengthen evidence for broad applicability.

Feature Representation Gaps: Network flow features proved inadequate for application-layer attack detection (web attacks), indicating the need for multi-layer feature integration. Future extensions should incorporate application-layer features (HTTP/HTTPS characteristics, TLS fingerprints, DNS patterns) alongside network flow statistics to enable comprehensive threat detection across all network layers.

Static Training Paradigm: Our framework uses batch training on historical data rather than online learning from streaming traffic. Real-world deployment requires adaptation to concept drift, evolving attack techniques, and emerging threats. Future work should develop online learning variants that continuously update models as new attack samples emerge while managing computational costs and catastrophic forgetting.

Limited Baseline Comparisons: We focused on demonstrating adaptive SMOTE effectiveness rather than systematic comparison against alternative sampling techniques (ADASYN, Borderline-SMOTE, SMOTE-ENN) or generative approaches (GANs, VAEs). Comprehensive comparative evaluation would better position our contribution within the landscape of imbalance handling techniques.

Several promising research directions emerge from these limitations:

1. **Hybrid Architectures:** Integrate adaptive SMOTE with CNN-LSTM-attention models to combine sampling strategy innovation with sophisticated temporal feature learning
2. **Generative Approaches:** Explore Variational Autoencoders (VAE) or Generative Adversarial Networks (GAN) for generating synthetic samples that extrapolate beyond observed feature distributions, potentially addressing extreme data scarcity
3. **Few-Shot Learning:** Adapt few-shot learning techniques from computer vision to enable intrusion detection with 5-10 examples per attack type, particularly relevant for zero-day threats
4. **Transfer Learning:** Pre-train models on large-scale common attack datasets then fine-tune on rare attack types to leverage transferred representations
5. **Multi-Modal Feature Fusion:** Combine network flow features with packet payload analysis, system call traces, and log data through multi-modal learning frameworks
6. **Adversarial Robustness:** Evaluate and enhance robustness against adversarial examples and evasion attacks, critical for security-sensitive applications
7. **Explainable Predictions:** Integrate explainability techniques to provide security analysts with interpretable rationales for predictions, particularly important for investigating flagged attacks
8. **Active Learning Integration:** Develop active learning strategies that selectively query analysts for labels on uncertain predictions, optimizing human expertise utilization

6.6 Broader Impact

Beyond intrusion detection, the adaptive SMOTE framework and insights from this work have broader applicability to imbalanced classification problems in security and beyond:

Security Domains: Fraud detection, malware classification, spam filtering, and vulnerability prediction all exhibit extreme class imbalance where rare classes represent critical threats. The tiered sampling strategy and feature adequacy analysis methodology transfer directly to these domains.

Medical Diagnosis: Rare disease detection, adverse drug reaction prediction, and medical anomaly identification face similar challenges of extreme imbalance with severe consequences for missed detections. Adaptive sampling strategies could improve rare condition detection while maintaining efficiency.

Industrial Systems: Predictive maintenance, quality control defect detection, and critical infrastructure monitoring involve identifying rare failure modes. The computational efficiency and interpretability of our approach suit industrial deployment contexts.

Financial Services: Credit card fraud detection, money laundering identification, and insider trading detection require detecting rare fraudulent transactions among millions of legitimate ones. Our tiered sampling approach could improve detection of sophisticated fraud schemes.

Methodological Contributions: The distinction between imbalance-driven and difficulty-driven classification failures, the concept of minimum sample thresholds for

effective SMOTE, and the importance of feature adequacy analysis provide general insights applicable across domains facing extreme imbalance.

6.7 Final Remarks

Class imbalance represents one of the most persistent challenges in machine learning for security applications, where the rarest events often represent the most critical threats. This work demonstrates that adaptive sampling strategies can substantially improve detection of minority attack types while maintaining overall accuracy and computational efficiency. Our framework achieved 99.9% improvement in class balance and enabled excellent detection (97-99% recall) for several minority attack types that would otherwise be ignored by models biased toward majority classes.

However, our results also reveal fundamental limits of synthetic oversampling: when initial samples are too sparse (<50 examples) or when features inadequately represent attack semantics (application-layer attacks with flow-level features), no amount of oversampling can compensate. These failures, while disappointing, provide valuable insights that should guide future research toward hybrid approaches combining adaptive sampling with advanced architectures, multi-modal features, and few-shot learning techniques.

The path forward requires integrating multiple complementary strategies: adaptive sampling to address imbalance, sophisticated architectures to learn complex patterns, comprehensive features to capture attack semantics across network layers, and continuous learning to adapt to evolving threats. Our adaptive SMOTE framework represents a foundational contribution toward this vision, demonstrating that thoughtful sampling strategies can substantially improve minority class detection in security-critical applications.

As cyber threats continue to evolve in sophistication and diversity, particularly in cloud computing environments where attack surfaces are expanding, the ability to detect rare but critical attacks becomes increasingly important. We hope this work contributes to more robust and comprehensive intrusion detection systems capable of protecting modern cloud infrastructures and network environments against the full spectrum of security threats, from common to exceedingly rare.

The insights from this research—particularly the distinction between imbalance-driven and difficulty-driven failures—can guide both practitioners deploying intrusion detection systems and researchers developing next-generation security solutions. By understanding when and why adaptive sampling succeeds or fails, the cybersecurity community can make more informed decisions about resource allocation, feature engineering priorities, and architectural choices.

Declarations

Availability of Data and Materials

The CIC-IDS2017 dataset used in this study is publicly available from the Canadian Institute for Cybersecurity at the University of New Brunswick: https://www.unb.ca/cic/datasets/ids-2017.html.

Competing Interests

The author declares no competing interests, financial or otherwise, related to this research.

COMPETING INTERESTS DISCLAIMER:
Authors have declared that they have no known competing financial interests OR non-financial interests OR personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Sajid, M.S.I., Wei, J., Ullah, I., Karim, F.K., Katib, I.: Enhancing intrusion detection: a hybrid machine and deep learning approach. Journal of Cloud Computing **13**(1), 123 (2024) https://doi.org/10.1186/s13677-024-00685-x

[2] Thakkar, A., Lohiya, R.: A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. Archives of Computational Methods in Engineering **28**, 3211–3243 (2021) https://doi.org/10.1007/s11831-020-09496-0

[3] Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. Journal of Big Data **6**(1), 27 (2019) https://doi.org/10.1186/s40537-019-0192-5

[4] Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009). https://doi.org/10.1109/CISDA.2009.5356528 . IEEE

[5] Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., Rajarajan, M.: A survey of intrusion detection techniques in cloud. Journal of Network and Computer Applications **36**(1), 42–57 (2013) https://doi.org/10.1016/j.jnca.2012.05.003

[6] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), pp. 108–116 (2018). https://doi.org/10.5220/0006639801080116

[7] Alshamrani, A., Myneni, S., Chowdhary, A., Huang, D.: A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. IEEE Communications Surveys & Tutorials **21**(2), 1851–1877 (2019) https://doi.org/10.1109/COMST.2019.2891891

[8] UCI Machine Learning Repository: KDD Cup 1999 Data (1999). http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[9] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2**(1), 20 (2019) https://doi.org/10.1186/s42400-019-0038-7

[10] López, V., Fernández, A., García, S., Palade, V., Herrera, F.: An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. Information Sciences **250**, 113–141 (2013) https://doi.org/10.1016/j.ins.2013.07.007

[11] Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials **18**(2), 1153–1176 (2016) https://doi.org/10.1109/COMST.2015.2494502

[12] Aburomman, A.A., Reaz, M.B.I.: A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems. Information Sciences **414**, 225–246 (2017) https://doi.org/10.1016/j.ins.2017.06.007

[13] Tao, P., Sun, Z., Sun, Z.: An improved intrusion detection algorithm based on GA and SVM. IEEE Access **6**, 13624–13631 (2018) https://doi.org/10.1109/ACCESS.2018.2810198

[14] Dhanabal, L., Shantharajah, S.P.: A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. International Journal of Advanced Research in Computer and Communication Engineering **4**(6), 446–452 (2015)

[15] Al-Jarrah, O.Y., Alhussein, O., Yoo, P.D., Muhaidat, S., Taha, K., Kim, K.: Data randomization and cluster-based partitioning for botnet intrusion detection. IEEE Transactions on Cybernetics **46**(8), 1796–1806 (2016) https://doi.org/10.1109/TCYB.2015.2490802

[16] Vinayakumar, R., Soman, K.P., Poornachandran, P.: Applying convolutional neural network for network intrusion detection. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1222–1228 (2017). https://doi.org/10.1109/ICACCI.2017.8126009 . IEEE

[17] Li, Z., Qin, Z., Huang, K., Yang, X., Ye, S.: Intrusion detection using convolutional neural networks for representation learning. In: International Conference on Neural Information Processing, pp. 858–866 (2017). https://doi.org/10.1007/978-3-319-70139-4_87 . Springer

[18] Kim, J., Kim, J., Thu, H.L.T., Kim, H.: Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon), pp. 1–5 (2016). https://doi.org/10.1109/PlatCon.2016.7456805 . IEEE

[19] Staudemeyer, R.C.: Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal **56**(1), 136–154 (2015) https://doi.org/10.18489/sacj.v56i1.248

[20] Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263 (2016). https://doi.org/10.1109/WINCOM.2016.7777224 . IEEE

[21] Moustafa, N., Slay, J.: UNSW-NB15: A comprehensive data set for network intrusion detection systems. In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6 (2015). https://doi.org/10.1109/MilCIS.2015.7348942 . IEEE

[22] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16**, 321–357 (2002) https://doi.org/10.1613/jair.953

[23] Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In: International Conference on Intelligent Computing, pp. 878–887 (2005). https://doi.org/10.1007/11538059_91 .

Springer

[24] He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks, pp. 1322–1328 (2008). https://doi.org/10.1109/IJCNN.2008.4633969 . IEEE

[25] Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 475–482 (2009). https://doi.org/10.1007/978-3-642-01307-2_43 . Springer

[26] Njama-Abang, O., Ashishie, D.U., Bukie, P.T.: Addressing class imbalance in Lassa fever epidemic data, using machine learning: a case study with SMOTE and random forest. Journal of the Nigerian Society of Physical Sciences **7**(3), 2586 (2025) https://doi.org/10.46481/jnsps.2025.2586 . Published online ahead of print

[27] Gu, J., Lu, S.: An effective intrusion detection approach using SVM with naïve Bayes feature embedding. Computers & Security **103**, 102158 (2021) https://doi.org/10.1016/j.cose.2020.102158

[28] Fernández, A., Garcia, S., Herrera, F., Chawla, N.V.: SMOTE for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. Journal of Artificial Intelligence Research **61**, 863–905 (2018) https://doi.org/10.1613/jair.1.11192

[29] Tomek, I.: Two modifications of CNN. IEEE Transactions on Systems, Man, and Cybernetics **SMC-6**(11), 769–772 (1976) https://doi.org/10.1109/TSMC.1976.4309452

[30] Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man, and Cybernetics **SMC-2**(3), 408–421 (1972) https://doi.org/10.1109/TSMC.1972.4309137

[31] Yen, S.-J., Lee, Y.-S.: Cluster-based under-sampling approaches for imbalanced data distributions. Expert Systems with Applications **36**(3), 5718–5727 (2009) https://doi.org/10.1016/j.eswa.2008.06.108

[32] Zhang, J., Mani, I.: KNN approach to unbalanced data distributions: A case study involving information extraction. In: Proceedings of Workshop on Learning from Imbalanced Datasets, vol. 126 (2003). ICML

[33] Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter **6**(1), 20–29 (2004) https://doi.org/10.1145/1007730.

1007735

[34] Elreedy, D., Atiya, A.F.: A comprehensive analysis of synthetic minority oversampling technique (SMOTE) for handling class imbalance. Information Sciences **505**, 32–64 (2019) https://doi.org/10.1016/j.ins.2019.07.070

[35] Roy, S.S., Mallik, A., Gulati, R., Obaidat, M.S., Krishna, P.V.: A deep learning based artificial neural network approach for intrusion detection. In: International Conference on Mathematics and Computing, pp. 44–53 (2017). https://doi.org/10.1007/978-981-10-4642-1_5 . Springer

[36] Yang, Y., Zheng, K., Wu, C., Yang, Y.: Improving the classification effectiveness of intrusion detection by using improved conditional variational AutoEncoder and deep neural network. Sensors **19**(11), 2528 (2019) https://doi.org/10.3390/s19112528

[37] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014) https://doi.org/10.48550/arXiv.1409.0473

[38] Wang, Z.: Deep learning-based intrusion detection with adversaries. IEEE Access **6**, 38367–38384 (2018) https://doi.org/10.1109/ACCESS.2018.2854599

[39] Zhang, Y., Chen, X., Jin, L., Wang, X.-J., Guo, D.: Network intrusion detection: Based on deep hierarchical network and original flow data. IEEE Access **7**, 37004–37016 (2019) https://doi.org/10.1109/ACCESS.2019.2905041

[40] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, , Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017). https://doi.org/10.48550/arXiv.1706.03762

[41] Khan, M.A., Kim, J.: Toward developing efficient Conv-AE-based intrusion detection system using heterogeneous dataset. Electronics **9**(11), 1771 (2020) https://doi.org/10.3390/electronics9111771

[42] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A.: A survey of network-based intrusion detection data sets. Computers & Security **86**, 147–167 (2019) https://doi.org/10.1016/j.cose.2019.06.005

[43] Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016). https://doi.org/10.1145/2939672.2939785 . ACM

[44] Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001) https://doi.org/10.1023/A:1010933404324

[45] Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997) https://doi.org/10.1109/4235.585893