Scilight

Pragmatic Cybersecurity
https://www.sciltp.com/journals/pc

Article

Investigating Oversampling Techniques to Mitigate Class Imbalance in Network Intrusion Detection Datasets

Huy Minh Dinh *, Wei Zong and Yang-Wai Chow

Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, Australia
* Correspondence: hmd819@uowmail.edu.au

How To Cite: Dinh, H.M.; Zong, W.; Chow, Y.-W. Investigating Oversampling Techniques to Mitigate Class Imbalance in Network Intrusion Detection Datasets. *Pragmatic Cybersecurity* **2026**, *1*(1), 4.

Received: 14 November 2025
Revised: 21 December 2025
Accepted: 9 January 2026
Published: 27 January 2026

Abstract: Network Intrusion Detection Systems (NIDS) play a crucial role in safeguarding computer networks against increasingly sophisticated cyber threats. However, the performance of machine learning–based NIDS is often constrained by severe class imbalance, in which benign traffic dominates and rare attack types are underrepresented, resulting in biased learning and reduced detection of minority classes that are critical to identify. This study presents a comprehensive comparative analysis of traditional and deep learning–based oversampling methods to mitigate class imbalance and enhance classification performance in NIDS. The evaluation is conducted on the UNSW-NB15 and TON_IoT benchmark datasets using a range of machine learning and deep learning classifiers, with performance assessed using metrics suitable for imbalanced data. Results show that traditional and hybrid oversampling methods provide stable and interpretable improvements, whereas deep generative approaches exhibit strong potential but greater variability across classifiers. In the UNSW-NB15 dataset, severe class imbalance and class overlap limit performance gains from resampling, while in the TON_IoT dataset, classifiers achieve strong baselines even without oversampling. XGBoost consistently demonstrates robust and reliable performance across datasets. Overall, KMeans-SMOTE, SMOTE-NC, and CVAE emerge as the most effective oversampling techniques under varying conditions. This study highlights the trade-offs between interpretability, stability, and detection performance, offering practical guidance for selecting oversampling strategies to improve rare attack detection in practical cybersecurity applications.

Keywords: network intrusion detection; class imbalance; oversampling methods; machine learning; cybersecurity

1. Introduction

In this day and age, where every device is connected to the network, the security of computer networks is of paramount importance. Network Intrusion Detection Systems (NIDS) are critical security mechanisms that monitor network traffic to detect potential intrusions or anomalous activities indicative of security breaches in networks, computer systems, and Internet of Things (IoT) devices. According to García-Teodoro et al. [1], anomaly detection techniques can be classified into three main categories: statistical-based, knowledge-based, and machine learning–based, depending on the approach used to model normal and abnormal behavior.

Signature-based intrusion detection, also referred to as knowledge-based intrusion detection, identifies known attacks by by matching network activity against predefined patterns, or signatures. While this approach is highly effective at identifying previously known threats with a low false positive rate, it has notable limitations in detecting sophisticated and novel cyberattacks [2]. The fundamental weakness of this approach lies in its reactive nature, as it requires prior knowledge of attack patterns to create effective detection rules, rendering it ineffective against previously unseen or evolving attack variants until corresponding signatures are developed [3]. Moreover, the proliferation

of attack variants and the emergence of zero-day threats have made the maintenance of signature repositories both computationally expensive and time-consuming [4,5].

Anomaly-based NIDS, also referred to as statistical-based NIDS, creates a baseline of normal network activity, flagging any deviations from this baseline as potential anomalies or abnormal behaviour [6]. These systems continuously monitor and analyze network traffic, including metrics such as traffic volume, IP addresses, and service ports, to construct a statistical model representing normal or legitimate behaviour [7]. While capable of detecting novel attacks, anomaly-based approaches often produce high false positive rates, as unfamiliar yet benign traffic in dynamic network environments can be misclassified as malicious [7,8].

To mitigate these issues, researchers have explored the integration of machine learning (ML) and deep learning (DL) techniques. Notably, DL-based NIDSs have garnered significant attention due to their capability to automatically learn intricate patterns from raw network traffic without relying on feature engineering [9]. The adoption of artificial intelligence (AI) into intrusion detection has enhanced automated monitoring capabilities and improved detection performance.

Despite significant progress in applying AI to network intrusion detection, a key challenge persists: severe class imbalance between normal and malicious network traffic, where the volume of normal traffic significantly exceeds that of attack traffic [10]. In real-world environments, malicious activities typically represent only a small fraction of total traffic, with legitimate behavior often exceeding 90%. This imbalance is not merely a technical artifact but an intrinsic characteristic of operational network environments that intrusion detection systems must effectively address. Similarly, benchmark network security datasets exhibit this imbalance, where normal traffic vastly outnumbers attack traffic and certain attack types appear extremely infrequently. As a result, models trained on such data tend to become biased toward the majority (benign) class, leading to poor detection of rare yet critical attack instances that pose substantial security threats [11,12].

The class imbalance problem can be addressed through three primary approaches, which Al-Qarni and Al-Asmari categorize as data-level, algorithm-level, and hybrid techniques [13]. Among these, data-level methods such as oversampling have been widely adopted to balance class distributions by generating additional samples for minority classes. Traditional techniques, including Synthetic Minority Oversampling Technique (SMOTE) and its variants, create synthetic examples through interpolation, while more recent deep generative models such as Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN) enable the generation of more realistic and diverse synthetic data. Although extensive research has been conducted on evaluating resampling methods to address class imbalance in NIDS [14–17], most studies have primarily focused on traditional oversampling techniques such as SMOTE, Adaptive Synthetic Sampling (ADASYN), and Random Oversampling (ROS). The potential of deep learning–based approaches derived from VAE and GAN architectures, however, has not been thoroughly explored. These advanced methods can generate more complex and realistic synthetic samples, yet their comparative effectiveness and suitability in the NIDS domain remain largely underexamined.

Another important limitation in prior studies relates to the selection and evaluation of classifiers. Many investigations of oversampling methods consider only a single classifier, either a traditional machine learning model or a deep learning model [16–18]. Some studies focus exclusively on a limited subset of well-performing machine learning algorithms [14,19], leaving out potentially powerful deep learning approaches. Consequently, there is a notable gap in comparative research that systematically examines the performance differences between machine learning and deep learning classifiers under conditions of class imbalance. Given the demonstrated representational capacity of deep learning models, addressing this gap presents a valuable opportunity to improve detection performance in network intrusion detection systems.

In this paper, we investigate the impact of various oversampling methods on machine learning– and deep learning–based NIDSs. The study evaluates how different oversampling techniques affect overall classification performance for both minority and majority classes and identifies which combinations of oversampling approaches and classifiers achieve the most reliable and balanced detection.

Our main contributions are summarized as follows:

- We analyze and compare multiple oversampling techniques applied to network intrusion detection datasets and evaluate their impact on both minority and majority class detection, using appropriate metrics for overall classification performance.
- We determine the optimal oversampling ratio that enhances minority class detection while minimizing noise and overfitting.
- We evaluate and compare the performance of machine learning-based and deep learning-based classifiers trained on oversampled network intrusion detection datasets using suitable metrics for imbalanced data.

2. Related Work

Oversampling has emerged as a widely adopted strategy for addressing the challenges of imbalanced datasets, particularly in classification tasks where minority classes are underrepresented. This section offers an overview of the main categories of oversampling strategies, highlighting their underlying principles and reviewing selected studies that examine these techniques. Broadly, these techniques can be divided into three groups: traditional oversampling methods, hybrid approaches that integrate oversampling with other techniques, and more recent deep learning-based oversampling strategies.

2.1. Traditional Methods

2.1.1. Synthetic Minority Oversampling Technique for Nominal and Continuous (SMOTE-NC)

SMOTE-NC is a variant of SMOTE designed to handle datasets containing a mix of nominal (categorical) and continuous (numerical) features. For numerical features, SMOTE-NC operates on the same principle as the original SMOTE: it generates synthetic samples by interpolating between an instance of the underrepresented class and one of its k-nearest neighbors (KNN). For categorical features, where interpolation is not feasible, SMOTE-NC assigns the most frequent value (mode) among the k-nearest neighbors to the generated sample [20]. Except for SMOTE-NC, most oversampling methods do not natively support categorical features and therefore require these features to be encoded into a numerical format prior to application. Derhab et al. [21] developed a temporal convolutional neural network (TCNN) for IoT-based intrusion detection, achieving up to 99% multiclass classification accuracy through the application of SMOTE-NC to balance the Bot-IoT dataset. In the study by Bulavas et al. [22] on multiclass classification using highly imbalanced network intrusion datasets, SMOTE-NC was combined with fixed random undersampling to mitigate class imbalance in the LITNET-2020 dataset, enabling more reliable detection of rare malicious classes. Although not directly related to NIDS, a study on phishing website detection compared three resampling methods using the Extreme Gradient Boosting (XGBoost) classifier, with SMOTE-NC proving most effective in enhancing detection performance and efficiency [23].

2.1.2. Adaptive Synthetic Sampling (ADASYN)

ADASYN builds on SMOTE's core idea of generating synthetic minority class samples through interpolation but introduces an adaptive mechanism to focus on regions where the minority class is underrepresented or difficult to classify. Using a KNN approach, it estimates the density of minority samples and computes a density ratio based on the proportion of majority class neighbors for each minority sample. Low-density (hard) regions are assigned higher weights, generating more synthetic samples, while high-density (easy) regions produce fewer. Synthetic samples are created by interpolating between a minority sample and its neighbors, effectively rebalancing the dataset [24]. Liu et al. [25] leveraged the computational efficiency and high accuracy of Light Gradient Boosting Machine (LightGBM) in combination with ADASYN to address class imbalance in the NSL-KDD, UNSW-NB15, and CIC-IDS2017 datasets, resulting in improved overall accuracy across all three test sets. In [26], ADASYN was employed to address the class imbalance problem, enhancing the feature extraction capability of a sparse autoencoder and the classification and detection performance of a Random Forest (RF), resulting in superior outcomes compared to other approaches in the study. Pan and Xie [27] aimed to reduce the false positive rate (FPR) caused by class imbalance and feature redundancy in the KDD-Cup'99 dataset by applying principal component analysis (PCA) for dimensionality reduction and ADASYN for oversampling, achieving significantly lower FPR and improved F1-scores.

2.2. Hybrid Approaches

2.2.1. K-Means Synthetic Minority Over-Sampling Technique (KMeans-SMOTE)

KMeans-SMOTE enhances traditional SMOTE by integrating k-means clustering to mitigate issues with class overlap. The feature space is first partitioned into k clusters, highlighting regions where the minority class is most concentrated. Clusters with a high proportion of minority samples are selected for oversampling, while those dominated by the majority class or containing significant overlap are avoided. Within the selected clusters, SMOTE is applied to generate synthetic minority samples through interpolation between existing minority samples and their nearest neighbors. This strategy minimizes the creation of synthetic samples in noisy or overlapping areas, resulting in more effective and reliable oversampling [28]. Wu et al. [29] proposed a network intrusion detection algorithm combining an enhanced Random Forest with KMeans-SMOTE to address class imbalance in the NSL-KDD dataset, achieving improvements over related works despite not being fully optimized. A recent study proposes a hybrid machine learning model that combines KMeans-SMOTE for data balancing with PCA for dimensionality reduction

in wireless sensor network intrusion detection, achieving outstanding classification performance while reducing training and prediction times for real-time applicability [30]. In another approach to the KDD-Cup'99 dataset, Priyadarsini et al. [31] proposed an ensemble classification model that combines ordered weighted averaging (OWA) for feature selection with KMeans-SMOTE to address class imbalance, achieving superior performance over other oversampling-classifier combinations.

2.2.2. Borderline-SMOTE (B-SMOTE)

Borderline-SMOTE is an enhanced SMOTE variant that targets minority class samples near the decision boundary, where they are most likely to be misclassified. Using a KNN approach, minority samples are categorized as "safe" (mostly minority neighbors), "danger" (mixed minority and majority neighbors), or "noise" (mostly majority neighbors). Only "danger" samples in borderline regions are oversampled by interpolating between them and their nearest neighbors. There are two variants: Borderline1-SMOTE generates synthetic samples only from minority neighbors to keep them within minority-dominated regions, while Borderline2-SMOTE incorporates majority neighbors, placing samples closer to the decision boundary but increasing the risk of noise [32]. In the study by Zhang et al. [33], the ReliefF algorithm and B-SMOTE were employed for feature selection and oversampling, respectively, on the NSL-KDD dataset. Tested across three base classifiers, the approach significantly improved the detection accuracy of minority class samples. Similarly, Sun et al. [34] evaluated different feature subsets based on information gain rate, using B-SMOTE for data balancing, across three basic machine learning algorithms to determine the optimal feature selection strategy for the CIC-IDS2017 dataset.

2.3. Deep Learning-Based Techniques

2.3.1. Conditional Variational Autoencoder (CVAE)

An autoencoder (AE) is a neural network designed to learn efficient representations of input data by compressing it into a lower-dimensional latent space and subsequently reconstructing the data to approximate the original input as closely as possible [35]. An AE consists of two components: an encoder and a decoder. The encoder transforms the input data into a compact latent representation by progressively reducing its dimensionality, thereby capturing the most important features of the input. The decoder then takes this latent representation and attempts to reconstruct the original data, minimizing the reconstruction error between the input and the output. Autoencoders have diverse applications, including data denoising, data generation, dimensionality reduction, and feature learning.

The VAE [36,37] is a probabilistic generative framework that extends the standard autoencoder for tasks such as synthetic data generation. Like an AE, the VAE consists of an encoder and a decoder; however, the encoder outputs the parameters of a probability distribution, typically fully-factorized Gaussian, including the mean and standard deviation. A latent vector is then sampled from this distribution using the reparameterization trick, which ensures differentiability during training. This latent vector is fed into the decoder to reconstruct the original input. The VAE is trained to minimize both the reconstruction loss, which measures the similarity between the input and output, and the Kullback–Leibler (KL) divergence, which encourages the learned latent distribution to remain close to a prior (usually a standard normal distribution). By optimizing these objectives, the VAE learns a structured and continuous latent space suitable for generative tasks. The CVAE [38] is a variant of the VAE that incorporates auxiliary information, enabling the generation of data conditioned on specific attributes or categories. Xu et al. [39] proposed a novel loss function based on the log hyperbolic cosine (log-cosh) to better capture the discrete characteristics of intrusion data in a CVAE, naming the resulting model the Log-Cosh Conditional Variational Autoencoder (LCVAE). The LCVAE was employed to generate diverse synthetic samples for underrepresented classes in the NSL-KDD dataset, while a convolutional neural network (CNN) served as the classifier. Their experiments demonstrated that this approach not only outperformed other methods cited in their study but also exhibited remarkable computational efficiency. Yang et al. [40] leveraged the CVAE to assign weights to the hidden layers of a deep neural network (DNN) classifier via the encoder, achieved by embedding intrusion labels exclusively in the decoder, and termed this model the Improved Conditional Variational Autoencoder (ICVAE). Compared with well-known and state-of-the-art classifiers, ICVAE achieved superior performance, demonstrating higher accuracy and detection rates for minority and unknown attacks, along with a lower false positive rate. Another study proposed the Conditional Variational Laplace Autoencoder (CVLAE), an extension of the Variational Laplace Autoencoder (VLAE) that conditions on class labels [41]. CVLAE employs a full-covariance Gaussian as the posterior distribution, enhancing posterior expressiveness and reducing the difference between the true and learned approximate posterior through Laplace approximation. Although its classification performance on minority classes was less competitive compared to other methods, the approach demonstrated promising potential for further improvements through refinement and optimization.

2.3.2. Conditional Wasserstein Generative Adversarial Network with Gradient Penalty (CWGAN-GP)

GANs are among the most widely used deep generative models, with applications spanning text, image, and video generation, style transfer, and data synthesis. First introduced by Goodfellow et al. [42], GANs are capable of learning the underlying distribution of training data to address the generative modeling problem. The architecture is composed of two neural networks, a generator and a discriminator, which are trained to compete in an adversarial manner. The generator learns to capture the latent distribution of real samples and produces realistic synthetic data, while the discriminator attempts to distinguish between real and generated samples. This interaction forms a zero-sum game, where the discriminator seeks to maximize its ability to differentiate real from fake, and the generator simultaneously strives to minimize this objective by producing increasingly convincing synthetic data. The vanilla GAN uses Jensen–Shannon (JS) divergence as the minimization objective, aiming to reduce the divergence between the real data distribution and the generator's distribution during training.

Arjovsky et al. [43] proposed the Wasserstein GAN (WGAN) to address key issues present in the original GAN, including convergence instability, mode collapse, and vanishing gradients. Unlike vanilla GANs, which rely on the JS divergence to measure the difference between real and generated distributions, WGAN employs the Wasserstein distance. When the real and generated distributions have little or no overlap, the JS divergence provides limited information, whereas the Wasserstein distance can still accurately quantify the discrepancy between the two distributions. WGAN replaces the discriminator with a critic, which does not perform binary classification of real versus fake samples but instead assigns a score reflecting how "real" a sample is. The critic is trained to satisfy the 1-Lipschitz constraint, initially enforced through weight clipping. However, weight clipping can cause slow convergence and vanishing gradients; to address this, Gulrajani et al. [44] replaced it with a gradient penalty (WGAN-GP), which enforces the 1-Lipschitz constraint by penalizing the norm of the critic's gradient with respect to its input. Similar to CVAE, the addition of "conditional" denotes the inclusion of conditioning data, enabling the generation of samples based on specific attributes or categories. Mu et al. [45] proposed the use of WGAN-GP in the intrusion detection domain to enhance the detection of zero-day attacks on the NSL-KDD dataset. Both the original and generated samples were evaluated across multiple classifiers, with CNN achieving the greatest improvement, approximately 2% for both binary and multiclass classification tasks. Although the detection rate for certain categories decreased in multiclass tasks due to class overlap in the generated data, the approach still demonstrates promising potential. Zhang et al. [46] introduced a pretraining procedure into WGAN-GP to address class imbalance in Industrial IoT (IIoT) networks. The model is first trained on normal network traffic, after which the imbalanced data is fed into the pretrained model for retraining and synthetic data generation. NSL-KDD and CIC-IDS2018 were used as benchmark datasets for evaluation with the LightGBM classifier, resulting in relatively high F1-scores.

3. Problem Definition

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be a labeled intrusion detection dataset containing N samples, where $x_i \in \mathbb{R}^d$ represents a feature vector of d network traffic attributes, and $y_i \in \{1, 2, \ldots, K\}$ denotes one of K classes (including normal traffic). Due to class imbalance, some attack classes (minority classes) have significantly fewer samples than majority classes (normal or frequent attacks), which can bias classifier performance.

The classification task is to learn a function $f_\theta : \mathbb{R}^d \to \{1, 2, \ldots, K\}$, parameterized by θ, that achieves high performance across both minority and majority classes.

To address class imbalance, an oversampling method $\mathcal{O}(\cdot)$ is applied to generate synthetic minority samples: $\mathcal{D}' = \mathcal{O}(\mathcal{D})$, where $|\mathcal{D}'| > |\mathcal{D}|$ and the class distributions are more balanced.

In this study, classification is performed on both the original dataset $\mathcal{D}$ and the oversampled dataset $\mathcal{D}'$ to evaluate the effect of oversampling on classifier performance. Performance is evaluated using the metrics described in Section 4.3 to ensure a fair assessment across all classes. The objective of this study is to evaluate the effectiveness of traditional and deep learning–based oversampling techniques in enhancing classifier performance across both minority and majority classes.

4. Methodology

4.1. Framework Overview

4.1.1. Synthetic Data Generation Framework

The framework for the synthetic data generation process is illustrated in Figure 1. An NIDS training set first undergoes a series of preprocessing steps, including removal of the binary and multiclass target columns and record identifiers, creation of a 20% validation set for early stopping in the CVAE, label encoding of nominal features

for traditional and hybrid oversampling methods, encoding of target variables for deep generative models, one-hot encoding of nominal features, and normalization of numerical features. It should be noted that the preprocessing pipeline differs depending on the oversampling method applied. Once preprocessing is complete, the dataset is processed by the selected oversampling technique to generate synthetic samples, which are then combined with the original dataset and exported as a CSV file to form a new training dataset for evaluation.

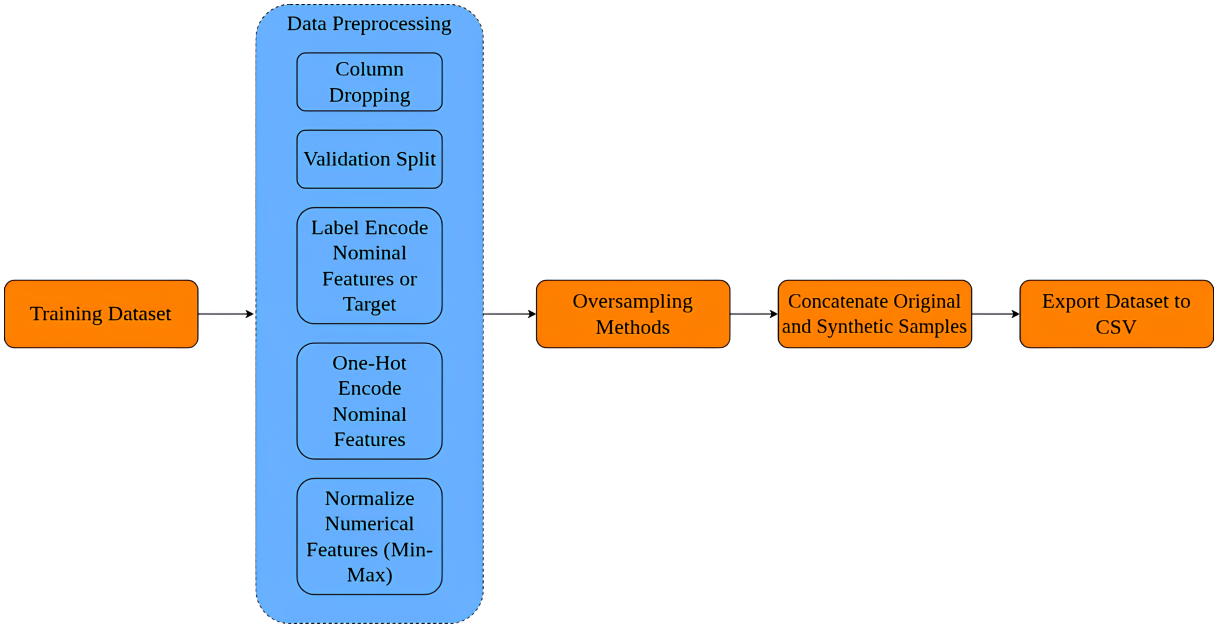

Figure 1. Flowchart of the proposed synthetic data generation process.

4.1.2. Classification Framework

Figure 2 presents the flowchart of the classification process. Both the training and testing datasets first undergo preprocessing, which includes removing unnecessary columns to retain only the features, applying label encoding to the target variable, performing one-hot encoding for nominal features, and applying min–max normalization to numerical features. Additionally, a 10% validation set is created from the processed training data using the `StratifiedShuffleSplit` function from the scikit-learn library, to support hyperparameter tuning and early stopping for deep learning–based classifiers. The training and validation sets are then used to train the model and optimize its hyperparameters, while the testing set is reserved for evaluating the model and obtaining the final classification results.

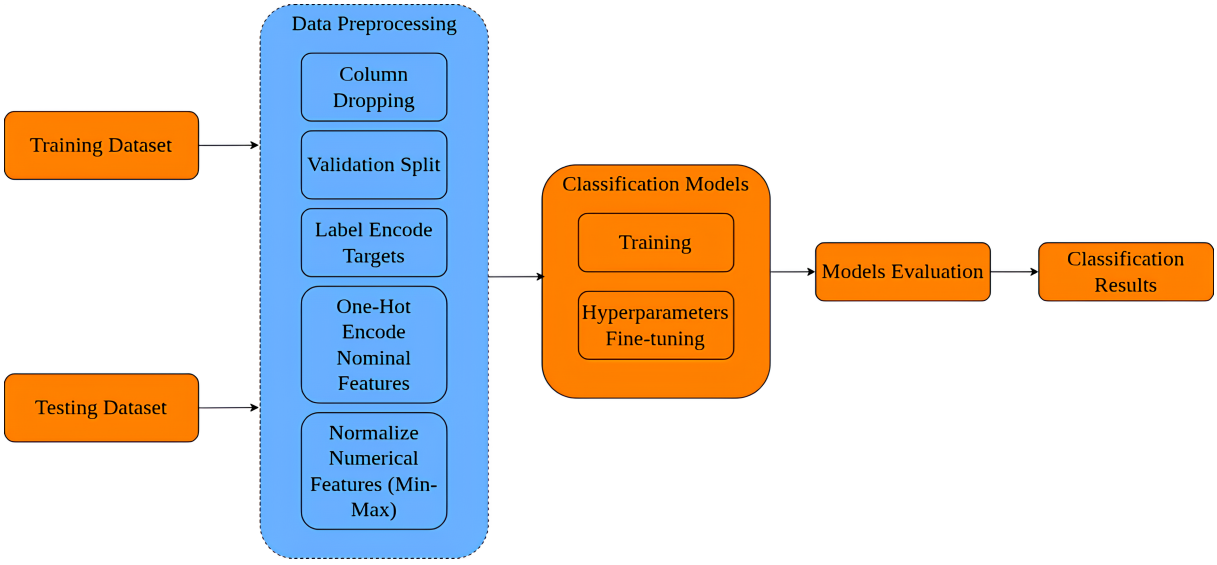

Figure 2. Flowchart of the proposed classification process.

4.2. Classifiers Selection

4.2.1. Ensemble Learning Models

Ensemble learning is a machine learning technique that combines multiple models, often referred to as base learners or weak learners, to produce a single model that is more accurate and robust. This technique can be applied to both traditional machine learning methods and deep learning models, the latter commonly known as deep ensemble learning. Two widely used tree-based algorithms, RF and XGBoost, are employed in this study, each utilizing a different ensemble strategy to perform multiclass classification. In this context, these two classifiers will hereafter be collectively referred to as ML-based models for simplicity.

RF is an ensemble method that utilizes the bagging technique, also known as bootstrap aggregating, to construct a collection of decision trees. Each tree is trained on a random subset of the training data generated through sampling with replacement, and their predictions are then aggregated. The final result is obtained by majority voting for classification tasks or averaging for regression tasks. By combining the outputs of multiple diverse trees, RF reduces variance in the base algorithm, thereby lowering the risk of overfitting and enhancing model accuracy. In addition, the ensemble nature of Random Forest enables it to be robust to noise and outliers, as averaging across the ensemble minimizes the influence of any individual tree trained on subsets containing noisy or extreme data points.

XGBoost is an ensemble learning algorithm that constructs decision trees sequentially, where each tree aims to correct the errors of its predecessors. This sequential error-correction mechanism is known as boosting, an alternative ensemble learning technique. XGBoost employs gradient descent on a specified loss function to guide the learning process. At each iteration, a new tree is fitted to the residuals (gradients) of the current model, and the predictions are updated accordingly, resulting in progressively improved accuracy. The final prediction is obtained by aggregating the outputs of all trees: for classification tasks, the sum is transformed into probabilities using functions such as sigmoid or softmax, while for regression tasks, the sum provides the prediction directly. XGBoost employs several regularization techniques, such as L1 (Lasso), L2 (Ridge) regularization, shrinkage, and tree pruning, to mitigate the overfitting commonly observed in standard boosting algorithms. XGBoost's nature renders it less robust to noise and outliers compared to RF, as it is designed to correct the mistakes of previous models. When an outlier or noisy data point produces a large error, subsequent trees may place excessive emphasis on correcting it, thereby increasing the risk of overfitting.

4.2.2. Deep Learning Models

DL is a subset of ML in which models are composed of multiple hidden layers, commonly referred to as DNNs. These models are more powerful than traditional ML models due to their deep architectures, which enable them to capture complex, non-linear patterns in the data. This study examines two neural network architectures: the one-dimensional convolutional neural network (1D-CNN) and the multilayer perceptron (MLP), both of which are well-suited for structured input data. For simplicity, these classifiers will from now on be collectively referred to as DL-based classifiers.

CNNs are widely recognized in the field of computer vision, particularly for image recognition and classification tasks. In the NIDS domain, they are employed for supervised feature extraction and classification. While CNNs are traditionally designed for two-dimensional inputs such as images, which include height and width, network traffic data is inherently one-dimensional. To address this, some studies reshape the one-dimensional feature vector into a two-dimensional matrix for compatibility with CNNs, while others design one-dimensional CNNs specifically tailored for 1D data. A typical CNN architecture for classification consists of an input layer, multiple convolutional and pooling layers for feature extraction, and a fully connected layer followed by a softmax classifier for prediction. For applications involving one-dimensional or three-dimensional data, the input, convolutional, and pooling layers are modified accordingly to accommodate different input dimensionalities.

MLP is a type of feedforward neural network made up of fully connected layers of neurons, through which information flows unidirectionally from the input layer to the output layer. An MLP architecture typically consists of an input layer, one or more hidden layers, and an output layer. The hidden and output layers apply non-linear activation functions, enabling the network to model complex relationships. An MLP performs classification by learning to map input features to class probabilities through layers of weighted connections and activation functions. Training is carried out using backpropagation, which involves two main steps: the forward pass and the backward pass. During a forward pass, the input data is propagated through the network layer by layer, with each layer computing activations, until a final output is produced. This output is then evaluated using a loss function, which quantifies the difference between the predicted and actual target values. During the backward pass, the gradients of the loss function with respect to the network's weights are calculated and used to update the weights, thereby minimizing the loss.

4.3. Performance Metrics

To assess the effectiveness of the oversampling methods, three main performance metrics are reported: F1-score, area under the precision-recall curve (AUPRC), and G-mean, while precision and recall are used in their calculation. These measures are derived from the confusion matrix in the context of network attack classification. Table 1 presents the 2×2 structure of a confusion matrix in binary classification. A multiclass confusion matrix follows the same principle but expands into an $N \times N$ arrangement, with N corresponding to the total number of classes. In the table, the notations are as follows: TP (True Positive) denotes the number of instances where positive samples are correctly classified; TN (True Negative) represents the number of instances where negative samples are correctly classified; FP (False Positive) refers to the number of negative samples incorrectly classified as positive; and FN (False Negative) indicates the number of positive samples incorrectly classified as negative. In the case of a multiclass confusion matrix, these concepts are applied on a per-class basis rather than to the matrix as a whole.

Table 1. Confusion matrix for binary classification.

	Predicted Positive	**Predicted Negative**
Actual Positive	TP	FN
Actual Negative	FP	TN

In the context of NIDS, precision measures the proportion of records predicted as attacks that are indeed actual attacks. A higher precision value indicates stronger model performance in correctly identifying attack traffic. It is defined as follows in Equation (1):

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

Recall, also known as the detection rate (DR), represents the proportion of actual attack records that are correctly identified by the classifier. It is often referred to as the true positive rate (TPR) or sensitivity. A higher recall indicates stronger performance in identifying attacks. The metric is defined as follows in Equation (2):

$$\text{Recall} = \text{DR} = \text{Sensitivity} = \text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

F1-score is the harmonic mean of recall and precision. Unlike accuracy, which can give a misleading assessment in imbalanced datasets by favoring majority classes, the F1-score provides a more reliable and comprehensive evaluation of model performance. It balances both precision and recall, penalizing models that achieve high precision but low recall, or vice versa. By accounting for both false positives and false negatives, the F1-score emphasizes the correct detection of minority classes, which is crucial for intrusion detection. A higher F1-score signifies that the model maintains an effective balance between precision and recall, accurately detecting positive instances while minimizing both false positives and false negatives. It is defined as shown in Equation (3):

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{3}$$

The precision–recall (PR) curve is a two-dimensional plot that illustrates the trade-off between precision and recall across different classification thresholds, with precision on the y-axis and recall on the x-axis. The area under this curve, referred to as the area under the precision–recall curve (AUPRC), provides a single-value summary of a model's ability to detect positive instances, making it particularly valuable for imbalanced classification tasks. In comparison, the area under the receiver operating characteristic curve (AUC-ROC) measures the relationship between the TPR and the FPR at various thresholds. Although widely used, AUC-ROC can be misleading in imbalanced datasets because it incorporates true negatives, which may artificially inflate performance when the majority class dominates. Therefore, AUPRC is often the preferred metric in such scenarios, as it focuses on the positive class and offers a more informative evaluation of model performance on rare or minority instances. A higher AUPRC represents better model performance in distinguishing positive instances from negative ones, especially in imbalanced datasets. In the context of multiclass classification, the AUPRC is computed using the one-vs-rest (OVR) strategy, where each class is considered the positive class in turn, and all remaining classes are treated as negative. The AUPRC for each class is calculated as shown in Equation (4):

$$\text{AUPRC}_i = \int_0^1 \text{Precision}_i(r)\, d(\text{Recall}_i) \approx \sum_{k=1}^{n-1} \left(\text{Recall}_{k+1}^{(i)} - \text{Recall}_k^{(i)}\right) \cdot \frac{\text{Precision}_{k+1}^{(i)} + \text{Precision}_k^{(i)}}{2} \tag{4}$$

where i indexes the class being evaluated, r denotes the recall variable, n is the total number of sampled classification thresholds, and k indexes the sorted operating points along the precision–recall curve.

The G-mean is defined as the geometric mean of the true positive rate (sensitivity) and the true negative rate (specificity), offering an overall measure of performance in binary classification. In multiclass classification, the G-mean is defined as the geometric mean of the recall values for all classes, reflecting the classifier's balanced ability to correctly identify instances across every class. This metric is particularly well-suited for evaluating oversampling and undersampling techniques, as it ensures that gains in minority class detection are not achieved at the cost of majority class performance. Consequently, the G-mean is highly effective in assessing classifiers on imbalanced datasets, where maintaining predictive accuracy across both minority and majority classes is critical. The formula for G-mean in multiclass classification is expressed as follows in Equation (5):

$$\text{G-mean} = \left(\prod_{i=1}^{N} \text{Recall}_i \right)^{\frac{1}{N}} = \left(\prod_{i=1}^{N} \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i} \right)^{\frac{1}{N}} \tag{5}$$

with N representing the total number of classes and i indexing the individual classes.

The metrics are initially computed for each class individually and then averaged using the macro strategy to yield an overall performance score. Macro averaging is preferred over micro and weighted averaging, as it assigns equal importance to both majority and minority classes. In contrast, micro averaging aggregates TPs, FPs, and FNs across all classes, causing the results to be biased toward classes with larger sample sizes. In highly imbalanced datasets, this often produces misleadingly high and uniform values for precision, recall, and F1-score. Weighted averaging, on the other hand, accounts for class size when computing averages. As a result, if a model performs well on the majority class but poorly on minority classes, the weighted metrics may still appear high, masking poor performance on underrepresented data. Although macro metrics may yield lower values than micro or weighted metrics due to weaker performance on minority classes, they provide a more accurate reflection of the classifier's overall performance and are therefore adopted in this study.

5. Experimental Setup

This section outlines the datasets employed in our experiments, the data preprocessing procedures, the oversampling ratios applied, and the implementation details of the study.

5.1. Overview of Datasets

Most existing studies rely on older benchmark datasets such as KDD-Cup'99 and NSL-KDD for evaluating NIDS performance. Although NSL-KDD addresses several limitations of KDD-Cup'99, such as the redundancy of records in the training set [47], it is outdated and does not adequately represent the characteristics of a modern attack environment. In this study, the UNSW-NB15 [48] and TON_IoT [49] datasets are selected to evaluate the performance of various oversampling techniques, as they provide more diverse, up-to-date, and realistic representations of current network traffic and attack scenarios.

5.1.1. UNSW-NB15

A group of cybersecurity researchers developed a new NIDS dataset in the Cyber Range Lab of the Australian Cyber Security Centre (ACSC), aimed at overcoming the limitations of older datasets [50]. The resulting UNSW-NB15 dataset was generated using the IXIA PerfectStorm tool to simulate a mix of realistic modern normal traffic and synthetic contemporary malicious traffic.

The tcpdump tool was used to capture network packets containing nine types of simulated modern cyberattacks including Analysis, Backdoor, Denial of Service (DoS), Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms, as well as normal traffic. These packets were then processed using Argus, Bro-IDS, and twelve C# algorithms to extract 49 distinct features.

The dataset comprises over two million records, distributed across four CSV files: UNSW-NB15_1.csv, UNSW-NB15_2.csv, UNSW-NB15_3.csv and UNSW-NB15_4.csv, which can be merged and randomly split for training and testing purposes. In this study, the publicly available partitioned training and testing sets, containing 175,341 and 82,332 records respectively, are used. Table 2 presents the sample distribution for each class in the datasets.

Table 2. Distribution of the UNSW-NB15 dataset by category.

Category	Training Set	Training Set (%)	Testing Set	Testing Set (%)
Normal	56,000	31.94	37,000	44.94
Generic	40,000	22.81	18,871	22.92
Exploits	33,393	19.05	11,132	13.52
Fuzzers	18,184	10.37	6062	7.36
DoS	12,264	6.99	4089	4.97
Reconnaissance	10,491	5.98	3496	4.25
Analysis	2000	1.14	677	0.82
Backdoor	1746	1.00	583	0.71
Shellcode	1133	0.65	378	0.46
Worms	130	0.07	44	0.05
Total	175,341	100.00	82,332	100.00

5.1.2. TON_IoT

A few years later, the creators of UNSW-NB15 introduced the next generation of a dataset for the IoT and IIoT to facilitate the evaluation of AI-based cybersecurity solutions [51]. The dataset comprises heterogeneous data collected from multiple sources such as IoT sensor telemetry, network traffic packets, and Linux and Windows logs, representing the interconnected three-layer architecture of IoT, Cloud, and Edge/Fog systems.

This report focuses on the network subset of the TON_IoT dataset, where network traffic is captured and processed similarly to the UNSW-NB15 dataset. Among the nine attack types included, seven are new, comprising Password, Cross-Site Scripting (XSS), Ransomware, Distributed Denial of Service (DDoS), Scanning, Injection, and Man-in-the-Middle (MITM) attacks.

The TON_IoT network dataset contains over 20 million records distributed across 23 CSV files. For this study, the train_test_network.csv subset is used to evaluate and compare various oversampling methods. Since this subset is not pre-split into training and testing sets, the `train_test_split` function from the scikit-learn library is employed, with stratification by attack type to preserve class distribution, resulting in an 80% training set and a 20% testing set. Table 3 shows the distribution of samples for each class across the training and testing sets.

Table 3. Distribution of the TON_IoT dataset by category.

Category	Training Set	Training Set (%)	Testing Set	Testing Set (%)
Normal	40,000	23.67	10,000	23.66
Password	16,000	9.48	4000	9.48
Backdoor	16,000	9.48	4000	9.48
XSS	16,000	9.48	4000	9.48
Ransomware	16,000	9.48	4000	9.48
DDoS	16,000	9.48	4000	9.48
Scanning	16,000	9.48	4000	9.48
DoS	16,000	9.48	4000	9.48
Injection	16,000	9.48	4000	9.48
MITM	834	0.49	209	0.50
Total	168,834	100.00	42,209	100.00

5.2. Data Preprocessing

The preprocessing steps for synthetic data generation and classification are largely identical, with minor adjustments made to accommodate different scenarios. The overall procedure is outlined below:

- For both datasets, the target columns corresponding to binary and multiclass classification are removed. In addition, the UNSW-NB15 dataset has an ID column, representing the record number, which is also dropped. The remaining columns serve as the features of the datasets and are used for further processing.
- In the CVAE, the `train_test_split` function is applied to create a 20% validation set from the training data, which is used for early stopping to prevent overfitting.

- During the synthetic data generation step, categorical (nominal) features in the datasets are transformed using label encoding, since most traditional and hybrid oversampling methods cannot natively handle such feature types. For deep learning–based synthetic generation and classification models, label encoding is also applied to the multiclass target values. The `LabelEncoder` function from the scikit-learn library is employed to convert string labels into numerical representations.
- In deep learning–based synthetic generation and classification models, categorical (nominal) features are converted into numerical representations using one-hot encoding prior to model training. This approach generates new binary columns corresponding to the number of distinct categories in the original feature, with each column representing a single category. A value of 1 is assigned to the column matching the category of a given record, while all other category columns are assigned a value of 0. The `OneHotEncoder` function from the scikit-learn library is employed for this transformation.
- Min–max normalization is applied to the numerical features to scale them to the [0, 1] range. This scaling is performed using the `MinMaxScaler` function from the scikit-learn library, which follows the mathematical formula shown in (6). In this formula, x is the original value, x_{min} and x_{max} are the minimum and maximum values of the feature respectively, and x' is the normalized value scaled to the range [0, 1].

$$x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{6}$$

5.3. Oversampling Application

5.3.1. UNSW-NB15

The oversampling methods outlined in Section 2 are applied exclusively to the training set of the UNSW-NB15 dataset, while the testing set is preserved for the evaluation of classification models. Two oversampling ratios are considered. In the first approach, the classes that account for less than five percent of the dataset, namely Analysis, Backdoor, Shellcode, and Worms (Table 2), are increased to 10,000 samples each, combining both original and synthetic instances. In the second approach, all classes are expanded to match the size of the Normal class, which contains 56,000 samples. The resulting distributions of the UNSW-NB15 training dataset under both oversampling strategies are presented in Table 4.

Table 4. Distributions of the UNSW-NB15 training datasets by category (after upsampling).

Category	Training Set Minority Class Upsampled	Training Set Minority Class Upsampled (%)	Training Set Class-Balanced Upsampled	Training Set Class-Balanced Upsampled (%)
Normal	56,000	26.63	56,000	10.00
Generic	40,000	19.02	56,000	10.00
Exploits	33,393	15.88	56,000	10.00
Fuzzers	18,184	8.65	56,000	10.00
DoS	12,264	5.83	56,000	10.00
Reconnaissance	10,491	4.99	56,000	10.00
Analysis	10,000	4.75	56,000	10.00
Backdoor	10,000	4.75	56,000	10.00
Shellcode	10,000	4.75	56,000	10.00
Worms	10,000	4.75	56,000	10.00
Total	210,332	100.00	560,000	100.00

5.3.2. TON_IoT

In the case of the TON_IoT dataset, oversampling methods are again applied solely to the training set, with the testing set left intact for classification evaluation. Two oversampling ratios are examined. The first increases the MITM class to 16,000 samples, aligning it with the size of other attack classes. The second balances the dataset by expanding every class to 40,000 samples, which corresponds to the size of the Normal class. Table 5 shows the distributions of the TON_IoT training sets after applying the two oversampling ratios.

Table 5. Distributions of the TON_IoT training datasets by category (after upsampling).

Category	Training Set MITM Upsampled	Training Set MITM Upsampled (%)	Training Set Class-Balanced Upsampled	Training Set Class-Balanced Upsampled (%)
Normal	40,000	21.70	40,000	10.00
Password	16,000	8.70	40,000	10.00
Backdoor	16,000	8.70	40,000	10.00
XSS	16,000	8.70	40,000	10.00
Ransomware	16,000	8.70	40,000	10.00
DDoS	16,000	8.70	40,000	10.00
Scanning	16,000	8.70	40,000	10.00
DoS	16,000	8.70	40,000	10.00
Injection	16,000	8.70	40,000	10.00
MITM	16,000	8.70	40,000	10.00
Total	184,000	100.00	400,000	100.00

5.4. Hyperparameters and Implementation Details

The implementation of oversampling techniques and classification models is carried out using Python 3.11.9 and PyTorch 2.7.1. A fixed random seed of 42 is applied to all functions involving randomness, as well as to PyTorch's CPU and GPU operations where not explicitly stated. Additionally, deterministic behavior in cuDNN is enforced and its auto-tuner is disabled to minimize stochasticity and enhance reproducibility.

5.4.1. Oversampling Implementation

For classical and hybrid oversampling methods, including SMOTE-NC, ADASYN, KMeans-SMOTE, and B-SMOTE, implementations from the imbalanced-learn library are employed and applied directly to the datasets for both minority-class and class-balanced upsampling. The hyperparameters used for these methods are summarized in Table 6. In the case of SMOTE-NC, an array of integers is specified to indicate the indices of categorical features in the dataset, enabling the algorithm to handle them automatically. The `sampling_strategy` parameter is set according to the upsampling ratios defined in Section 5.3 for each dataset, while the `random_state` is fixed at 42 to ensure reproducibility.

Table 6. Hyperparameters for traditional and hybrid oversampling methods on UNSW-NB15 and ToN_IoT.

Method	Parameter	Value (UNSW-NB15/ToN_IoT)

SMOTE-NC	`categorical_features`
	`sampling_strategy`
	`random_state`
		categorical_indices
		sampling_strategy
		42

ADASYN	`sampling_strategy` `random_state`	sampling_strategy 42

KMeans-SMOTE	`kmeans_estimator`
	`cluster_balance_threshold`
	`sampling_strategy`
	`n_jobs`
	`random_state`
		20/N/A
		0.001/0.005
		sampling_strategy
		−1
		42

B-SMOTE	`sampling_strategy` `random_state`	sampling_strategy 42

In contrast, CVAE and CWGAN-GP are implemented manually using the PyTorch library, with their architectures and hyperparameters provided in Tables 7 and 8, respectively.

Table 7. CVAE architecture and training hyperparameters. D_{in} denotes the input feature dimension, y the conditional label, z the latent variable, μ the latent mean, $\log \sigma^2$ the latent variance, Z_{dim} the latent dimension.

Component	Specification/Value
Architecture Details	
Encoder	Input ($D_{in} + y$) $\rightarrow$ Linear(512) $\rightarrow$ Linear(256) $\rightarrow$ Linear(128)
Latent Space	μ, $\log \sigma^2 = \text{Linear}(128 \rightarrow Z_{dim})$
Decoder	Latent ($z + y$) $\rightarrow$ Linear(128) $\rightarrow$ Linear(256) $\rightarrow$ Linear(512) $\rightarrow$ Sigmoid
Activation	ReLU
Normalization	BatchNorm1d
Regularization	Dropout (0.3)
Latent Dimension (Z_{dim})	64
Training Hyperparameters	
Loss Function	Binary Cross-Entropy (BCE) + KL Divergence
Optimizer	Adam (Weight Decay: 1×10^{-5})
Learning Rate	0.001
Batch Size	64
Max Epochs	100
Early Stopping	10
Learning Rate Scheduler (ReduceLROnPlateau)	
Mode	`min` (on validation loss)
Factor/Patience	0.5/5 epochs

Table 8. CWGAN-GP architecture and training hyperparameters. z denotes the noise vector, y the conditional label, y_{embed} the embedded representation of y, x the real data input, D_{in} the dimensionality of real data (critic input), D_{out} the dimensionality of generated data (generator output), n_{critic} the number of critic updates per generator update, λ the gradient penalty coefficient, and β_1, β_2 the Adam optimizer parameters.

Component	Specification/Value
Architecture Details	
Generator	$z \odot y_{embed} \rightarrow$ Linear(256) $\rightarrow$ Linear(512) $\rightarrow$ Linear(1024) $\rightarrow$ Linear(D_{out}) $\rightarrow$ Sigmoid
Critic	$x \odot y_{embed} \rightarrow$ Linear(1024) $\rightarrow$ Linear(512) $\rightarrow$ Linear(256)
Embeddings	$y \rightarrow$ Vector (size 100 for Gen, size D_{in} for Critic)
Activation	LeakyReLU (0.2)
Regularization	Dropout (0.3) [Generator only]
Noise Dimension (z)	100
Training Hyperparameters	
Loss Function	Wasserstein Loss with Gradient Penalty
Optimizer	Adam ($\beta_1 = 0.5, \beta_2 = 0.9$)
Learning Rate	0.0001
Batch Size	256
Max Epochs	100
Critic Updates	$n_{critic} = 5$ per generator update
GP Coefficient	$\lambda = 10$

5.4.2. Classification Implementation

Random Forest and XGBoost are utilized directly from their respective libraries for classification, while MLP and 1D-CNN are implemented from scratch using PyTorch. The hyperparameters and architectures of these models are presented in Tables 9–11.

Table 9. Hyperparameters used for Random Forest and XGBoost classifiers.

Classifier	Parameter	Value
Random Forest	`random_state`	42
	`n_jobs`	-1
XGBoost	`random_state`	42
	`n_jobs`	-1
	`use_label_encoder`	`False`
	`eval_metric`	`'mlogloss'`

Table 10. MLP architecture and training hyperparameters. D_{out} denotes the number of classes for classification.

Component	Specification/Value
Architecture Details	
Hidden Layers	Linear(512) $\rightarrow$ Linear(256) $\rightarrow$ Linear(128)
Output Layer	Linear(D_{out})
Activation	ReLU
Regularization	Dropout (0.2)
Training Hyperparameters	
Loss Function	Cross-Entropy Loss
Optimizer	Adam
Learning Rate	0.001
Batch Size	64
Max Epochs	50
Early Stopping	5
Learning Rate Scheduler (ReduceLROnPlateau)	
Mode	`min` (on validation loss)
Factor/Patience	0.1/3 epochs

Table 11. 1D-CNN architecture and training hyperparameters. D_{out} denotes the number of classes for classification.

Component	Specification/Value
Architecture Details	
Convolutional Layers	Conv1D(64) $\rightarrow$ Conv1D(128) $\rightarrow$ Conv1D(256)
Filter Settings	Kernel Size: 3; Padding: 1; Stride: 1
Pooling Layers	MaxPool1D (Size: 2) after each Conv layer
Fully Connected	Linear(256) $\rightarrow$ Linear(128) $\rightarrow$ Linear(D_{out})
Activation	ReLU
Regularization	Dropout (0.3)
Training Hyperparameters	
Loss Function	Cross-Entropy Loss
Optimizer	Adam
Learning Rate	0.001
Batch Size	128
Max Epochs	100 (UNSW-NB15)/50 (TON_IoT)
Early Stopping	10 (UNSW-NB15)/5 (TON_IoT)
Learning Rate Scheduler (ReduceLROnPlateau)	
Mode	`min` (on validation loss)
Factor/Patience (UNSW-NB15)	0.5/5 epochs
Factor/Patience (TON_IoT)	0.1/3 epochs

6. Experimental Results

6.1. UNSW-NB15

Table 12 presents the classification results of the RF model on the UNSW-NB15 dataset, before and after applying oversampling. The baseline G-mean of 0.2889 is critically low, indicating that the classifier is heavily

biased toward the majority class and confirming the severity of the class imbalance problem. The baseline F1-score and AUPRC are also modest, reflecting the model's overall difficulty in handling minority classes.

Most traditional and hybrid oversampling techniques yield substantial improvements in G-mean, indicating that minority class recall benefits significantly from resampling. However, this gain comes with a trade-off between enhancing minority detection and preserving overall classification performance. This trade-off is evident in the inverse relationship between metrics: methods that substantially improve G-mean lead to only marginal shifts in F1-score and AUPRC, which may be positive or negative, affecting either one metric or both simultaneously. This pattern confirms that the improvement in minority class recall comes at the expense of precision, ranking ability, or both.

Under the minority-class upsampling strategy, KMeans-SMOTE achieves the highest F1-score, while CWGAN-GP performs best under the balanced upsampling strategy. ADASYN yields the largest gain in G-mean but experiences a decline in both F1-score and AUPRC, illustrating that improving recall for rare classes often reduces overall precision and ranking performance.

In contrast, deep learning-based methods perform poorly in balanced detection, with CVAE (minority) yielding a 0 G-mean, indicating a complete failure to predict any true positives for the minority class. This suggests the synthetic samples generated by these methods were either noisy or non-representative, thereby confusing the model's decision boundary. Despite this classification failure, these methods achieve higher macro AUPRC scores than the others, suggesting they have a strong ranking ability but their optimal classification threshold is severely misaligned.

Overall, upsampling the minority classes is generally preferred, as it outperforms balanced upsampling in four out of six cases across the evaluated oversampling methods. This holds true whether the focus is on minority class detection, measured by G-mean, or overall classification performance, measured by F1-score. These findings are supported by evaluating the average performance of all oversampling techniques under their respective oversampling ratios. However, it is important to recognize the inherent trade-off between these two metrics.

Table 12. RF classification results for the UNSW-NB15 dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced
Macro F1-score	0.4643	0.4638	0.4518	0.4652	0.4579	0.4717	0.4627	0.4706	0.4334	0.4611	0.4645	0.4606	0.4676	0.4655	0.4563
Macro AUPRC	0.5417	0.5192	0.4991	0.5200	0.4925	0.5288	0.5031	0.5095	0.4883	0.5377	0.5358	0.5435	0.5370	0.5265	0.5093
G-mean	0.2889	0.4853	0.4831	0.5020	0.5203	0.3986	0.3713	0.4770	0.4360	0.2716	0.0000	0.2696	0.3034	0.4007	0.3524

Table 13 presents the classification results of XGBoost on the UNSW-NB15 dataset, before and after oversampling. The baseline performance of XGBoost is relatively strong for an imbalanced dataset, confirming its robustness to class imbalance and highlighting the effectiveness of the boosting strategy, in contrast to RF. Nevertheless, the baseline G-mean remains low, indicating a noticeable bias toward the majority class.

Traditional and hybrid oversampling methods substantially improve the model's ability to balance predictions between majority and minority classes, as reflected in consistently higher G-mean scores, thereby confirming their effectiveness in boosting minority class recall without severely harming majority performance.

Deep learning–based models (CVAE and CWGAN-GP) achieve the highest F1-scores overall when combined with minority-class upsampling. However, these results require careful interpretation: both methods fail catastrophically in terms of G-mean, with most cases recording a score of 0 across both upsampling strategies. In the balanced upsampling strategy, KMeans-SMOTE achieves the highest macro F1-score, followed by CWGAN-GP and CVAE. This outcome indicates a complete collapse in detecting one of the minority classes. As a result, the elevated F1-scores are somewhat misleading, since they are largely sustained by strong majority-class performance with only minimal gains from some of the other minority classes.

Despite this, CVAE and CWGAN-GP achieve higher average AUPRC scores compared to other oversampling methods, suggesting that while their synthetic samples are too noisy for effective classification, the models still retain strong ranking ability. The issue lies not in the models' potential, but in the misaligned classification threshold that produces the zero G-mean.

These classification results highlight the inherent trade-off between metrics: maximizing G-mean often leads to a reduction in F1-score, AUPRC, or both, primarily due to an increase in false positives. For XGBoost, balanced upsampling is preferred when the goal is to optimize G-mean, with SMOTE-NC (balanced) achieving the highest score. In contrast, minority upsampling is generally more effective for maximizing F1-score, outperforming balanced upsampling in four out of six comparisons. This approach also proves safer for generative methods, as it limits the introduction of noise into the dataset. These findings are supported by the observation that the average metrics of all oversampling methods under the minority strategy are higher than those under balanced upsampling.

Table 13. XGBoost classification results for the UNSW-NB15 dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced
Macro F1-score	0.5084	0.4818	0.4792	0.4857	0.4665	0.5098	0.5112	0.4971	0.4965	0.5154	0.4921	0.5154	0.5059	0.5009	0.4919
Macro AUPRC	0.5661	0.5598	0.5463	0.5559	0.5354	0.5594	0.5384	0.5534	0.5404	0.5839	0.5761	0.5820	0.5629	0.5657	0.5499
G-mean	0.3965	0.4199	0.5443	0.4746	0.5273	0.4668	0.4657	0.4501	0.5055	0.3806	0.0000	0.0000	0.0000	0.3653	0.3405

Table 14 reports the classification results of the 1D-CNN on the UNSW-NB15 dataset, both before and after oversampling. At baseline, the model performs substantially worse than the two tree-based classifiers, exhibiting the weakest performance overall, as it completely fails to detect one of the minority classes, resulting in an overall G-mean of 0. Specifically, the baseline model is unable to classify any samples from the Analysis class (see Section 6.2 for details). Additionally, with CVAE (balanced) and CWGAN-GP (balanced), the Worms class is entirely misclassified, resulting in a G-mean of 0. This failure highlights the sensitivity of CNN architectures and emphasize the need for careful architectural design and hyperparameter tuning.

For the 1D-CNN, resampling proves essential for functionality. After oversampling, traditional and hybrid methods successfully mitigate these issues, with ADASYN and SMOTE-NC emerging as the best performers under the balanced upsampling strategy. In contrast, deep learning–based oversampling methods remain ineffective, as shown by persistent 0 G-mean scores. Because 1D-CNNs rely heavily on extracting local patterns, slightly noisy or mismatched generative samples from CVAE or GANs may obscure rare-class signals, resulting in recall of 0 and, consequently, a zero G-mean. Moreover, 1D-CNNs are highly sensitive to temporal or sequential artifacts introduced by generative models due to their dependence on local receptive fields and weight sharing. Generative samples may also differ in statistical properties (e.g., variance, range, or distribution moments), further disrupting recognition.

Under minority-only sampling, CWGAN-GP achieves the highest macro F1-score, followed by KMeans-SMOTE and CVAE. Under balanced sampling, KMeans-SMOTE attains the top macro F1-score, with ADASYN and SMOTE-NC performing next best. However, balanced upsampling can cause generative models to produce unrealistic samples that distort 1D-CNN decision boundaries, resulting in lower F1-scores and G-mean. In contrast, traditional and hybrid methods generate synthetic samples by interpolating between existing real instances, which may still introduce some noise or overlap but better preserve realistic feature correlations. This makes them more reliable than generative approaches, particularly for 1D-CNNs, which are highly sensitive to such artifacts.

The macro F1-score remains closely clustered between 0.4088 and 0.4512, indicating that resampling provides limited overall improvement for 1D-CNN models. At the same time, AUPRC generally declines compared to the baseline for most oversampling methods, suggesting that changes to the decision boundary can impair the model's ranking ability. In this case, the trade-off is particularly pronounced, as both F1-score and AUPRC decrease across all oversampling techniques despite gains in G-mean. For the 1D-CNN, the balanced strategy proves most effective for enhancing minority class detection, as measured by G-mean, whereas the minority strategy remains the preferred choice for improving overall classification performance, as reflected in F1-score averages across the two strategies.

Table 14. 1D-CNN classification results for the UNSW-NB15 dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced
Macro F1-score	0.4435	0.4211	0.4271	0.4216	0.4275	0.4464	0.4285	0.4158	0.4194	0.4403	0.4088	0.4512	0.4165	0.4327	0.4213
Macro AUPRC	0.5149	0.5044	0.4865	0.5052	0.4886	0.4887	0.4560	0.4745	0.4644	0.5016	0.4763	0.5107	0.4837	0.4975	0.4759
G-mean	0.0000	0.3857	0.5080	0.4047	0.5178	0.3622	0.3678	0.3383	0.4918	0.1917	0.0000	0.0000	0.0000	0.2804	0.3142

Table 15 summarizes the MLP's classification results on the UNSW-NB15 dataset under baseline and oversampled conditions. With its simpler architecture compared to the 1D-CNN, the MLP achieves comparable baseline performance, successfully detecting minority classes with a non-zero G-mean.

Consistent with previous findings, traditional and hybrid oversampling techniques substantially enhance G-mean as they synthesize new data points through linear interpolation between existing minority-class instances. This process expands the effective feature space of rare classes, reduces sparsity, and compels the classifier to lower its decision boundary, thereby boosting recall and improving G-mean. Moreover, these methods generate synthetic samples in a controlled manner, minimizing noise and avoiding significant overlap with the majority-class region.

By contrast, deep learning–based oversampling methods perform poorly in terms of G-mean, with CVAE (balanced) even failing to detect one class entirely. The MLP's F1-scores remain tightly clustered around its already low baseline, suggesting that oversampling contributes minimal overall performance improvement for this model. In addition, results from deep learning–based oversampling confirm the trade-off: techniques that maximize G-mean often yield slightly lower AUPRC and F1-scores than the baseline, indicating that the boundary shift required for

recall improvement adversely impacts overall precision and ranking ability.

For the MLP, the balanced upsampling strategy is most effective for maximizing G-mean, whereas minority-only upsampling is generally preferable for optimizing F1-score. This is supported by the higher average F1-score achieved under the minority strategy compared to balanced upsampling. By introducing fewer synthetic samples, the minority-only approach helps reduce false positives and better preserves the model's overall precision.

Table 15. MLP classification results for the UNSW-NB15 dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced	Minority	Balanced
Macro F1-score	0.4464	0.4321	0.4328	0.4374	0.4328	0.4497	0.4525	0.4413	0.4358	0.4414	0.4242	0.4434	0.4413	0.4409	0.4366
Macro AUPRC	0.5050	0.5097	0.4848	0.5054	0.4939	0.4919	0.4730	0.4884	0.4677	0.4932	0.4962	0.4897	0.4827	0.4964	0.4831
G-mean	0.2821	0.3439	0.4924	0.4529	0.5366	0.3966	0.4138	0.4307	0.4976	0.2313	0.0000	0.2217	0.1969	0.3462	0.3562

This analysis highlights the severity of combined class imbalance and overlap, as evident from baseline performance across multiple classifiers. Resampling helped mitigate the class overlap issue, enabling the 1D-CNN to function properly. However, without specialized frameworks or additional discriminative features, resampling offered limited improvement in overall model performance. Traditional and hybrid oversampling methods consistently proved to be the most reliable and effective approach, substantially increasing G-mean by expanding the minority feature space and alleviating class bias across all four classifiers. In contrast, deep learning–based generative methods were highly variable, frequently resulting in catastrophic failures (G-mean = 0), particularly in robust models such as XGBoost and sensitive architectures like 1D-CNN, despite occasionally achieving high F1-scores. Among the evaluated methods, KMeans-SMOTE, CVAE, and CWGAN-GP performed well under different classifiers, with CVAE and CWGAN-GP consistently achieving near-top F1-scores across most scenarios, even when the G-mean remained at zero. Considering the average performance across classifiers and oversampling strategies, XGBoost stands out as the most robust and reliable model. Minority-only oversampling is generally preferred because it introduces less noise into the classification process. With an appropriate framework to address class overlap, XGBoost's performance could be further enhanced.

These results reinforce the inherent trade-off in imbalanced learning: maximizing G-mean often comes at the expense of overall F1-score and AUPRC, primarily due to increased false positives from aggressive decision boundaries. Consequently, the balanced upsampling strategy is generally optimal for maximizing G-mean and minority-class detection, whereas the less aggressive minority-only strategy is preferable for optimizing F1-score and reducing the risk of catastrophic misclassification, as it introduces fewer overlapping synthetic samples.

6.2. Data Overlap in UNSW-NB15

As observed in the results above, certain cases yield a G-mean of 0. For multi-class problems, G-mean is computed as the n-th root of the product of the recall values for all classes (Equation (5)). Consequently, if any class has a recall of 0, the entire product collapses to 0, leading to a G-mean of 0, even when the F1-score remains non-zero. This behavior is observed in certain scenarios, irrespective of whether the G-mean is computed manually or using the `geometric_mean_score` function from the imbalanced-learn library with `average='multiclass'`. Although the function offers an option to assign a value to the recall of unrecognized classes to prevent a zero G-mean, this was not applied here, as it could misrepresent the actual performance.

This phenomenon is observed only in the UNSW-NB15 dataset, revealing that it is affected not only by class imbalance but also by significant data overlap [52]. Data overlap arises when multiple samples share highly similar feature representations or mimic the behavior of Normal or other classes, making it difficult for classifiers to establish clear decision boundaries. In network traffic, several classes can exhibit similar characteristics, for example, Analysis and Exploits may both involve probing or exploiting vulnerabilities in web applications or systems, resulting in overlapping features. Zoghi and Serpen [52] proposed a visualization-based approach to study this issue, showing that attack samples frequently overlap with Normal traffic and with other attack classes, with the degree of overlap varying between certain class pairs. Salman et al. [53] further observed that DoS, Exploits, Analysis, and Backdoor exhibit high similarities and often cluster together, which they refer to as Category 1 (Cat. 1) attacks. In their experiments, a classifier distinguishing Exploits from non-Exploits consistently misclassified nearly all DoS, Analysis, and Backdoor traffic as Exploits, underscoring the difficulty of separating highly similar attack behaviors. They further noted that Reconnaissance, Shellcode, and Worms are also frequently misclassified into Cat. 1. Even after removing certain fixed features, high false alarm rates persisted, likely due to either feature bias or intrinsic similarities across classes. In the context of evaluating oversampling methods, balanced oversampling can worsen this issue by amplifying data overlap, which may reduce overall model performance unless addressed through strategies such as attack categorization. Without introducing additional discriminative

features or adopting specialized techniques, the combined effects of class imbalance and overlap remain severely detrimental to classification performance.

The G-mean score of 0 observed in the UNSW-NB15 dataset arises from the zero recall (sensitivity) of the Analysis and/or Worm classes. The baseline performance for the Analysis class is particularly poor, as all of its samples are misclassified as Exploits, resulting in a recall score of 0, as shown in both the confusion matrix and the performance metrics (Tables 16 and 17). In the confusion matrix, the first four rows, which correspond to Analysis, Backdoor, DoS, and Exploits, reveal substantial misclassifications among these classes, reflecting their strong feature similarities. The remaining three rows, which correspond to Reconnaissance, Shellcode, and Worms, also indicate that the majority of their samples are misclassified as Exploits.

When CVAE-based balanced upsampling is applied, the problem worsens. Both the Analysis and Worms classes achieve a recall of 0, directly resulting in a G-mean of 0 (Table 18). The corresponding confusion matrix (Table 19) confirms that none of the samples from these classes are correctly classified. This indicates that introducing a large number of synthetic samples can exacerbate the data overlap issue, severely impairing classifier performance.

Table 16. Confusion matrix for 1D-CNN classification on the UNSW-NB15 dataset.

	Analysis	**Backdoor**	**DoS**	**Exploits**	**Fuzzers**	**Generic**	**Normal**	**Reconnaissance**	**Shellcode**	**Worms**
Analysis	0	0	0	677	0	0	0	0	0	0
Backdoor	0	11	2	560	5	0	0	1	4	0
DoS	0	6	224	3711	89	0	5	9	44	1
Exploits	16	10	126	10,657	192	0	51	24	55	1
Fuzzers	0	12	4	1831	3152	0	670	119	274	0
Generic	0	2	56	502	123	18,163	2	17	13	3
Normal	270	1	13	1179	6937	1	28,323	114	162	0
Reconnaissance	0	2	7	846	22	3	6	2603	7	0
Shellcode	0	0	0	79	52	0	3	31	213	0
Worms	0	0	0	34	2	0	0	0	2	6

Table 17. Class-level performance metrics for 1D-CNN classification on UNSW-NB15 (affected classes).

Class	**Precision**	**Recall**	**F1-Score**	**AUPRC**
Analysis	0.0000	0.0000	0.0000	0.0421
Worms	0.5455	0.1364	0.2182	0.2369

Table 18. Class-level performance metrics for 1D-CNN with CVAE (balanced) on UNSW-NB15 (affected classes).

Class	**Precision**	**Recall**	**F1-Score**	**AUPRC**
Analysis	0.0000	0.0000	0.0000	0.0419
Worms	0.0000	0.0000	0.0000	0.1019

Table 19. Confusion matrix for 1D-CNN with CVAE (balanced) on the UNSW-NB15 dataset.

	Analysis	**Backdoor**	**DoS**	**Exploits**	**Fuzzers**	**Generic**	**Normal**	**Reconnaissance**	**Shellcode**	**Worms**
Analysis	0	1	5	669	0	0	2	0	0	0
Backdoor	0	14	4	551	5	0	1	5	3	0
DoS	0	44	206	3679	62	0	7	52	39	0
Exploits	22	45	70	10,600	174	3	68	104	46	0
Fuzzers	0	17	7	1954	2881	0	650	496	55	2
Generic	0	3	228	560	52	17,999	4	15	8	2
Normal	611	2	12	1087	6174	0	28,428	614	52	20
Reconnaissance	0	7	9	804	27	0	20	2628	1	0
Shellcode	0	1	0	77	19	0	4	140	135	2
Worms	0	0	0	39	2	0	0	1	2	0

The models evaluated, including RF, XGBoost, MLP, and 1D-CNN, demonstrate significant architectural differences in their ability to handle class overlap, which is pronounced in this dataset. XGBoost emerged as the most robust classifier, using its adaptive boosting strategy to build highly complex decision boundaries that

effectively manage mixed feature regions, resulting in the highest baseline G-mean. In contrast, both RF and MLP struggled with overlap, producing conservative and biased predictions with low baseline G-mean scores. The 1D-CNN was the most sensitive model because its reliance on local pattern extraction makes it highly vulnerable to noise and artifacts in overlapping regions. This sensitivity contributed to the catastrophic failure of generative oversampling methods, resulting in a baseline G-mean of 0 and indicating that the CNN's local feature learning was severely disrupted by feature overlap and sample sparsity.

6.3. TON_IoT

Table 20 presents the RF classification performance on the TON_IoT dataset under both baseline and upsampled conditions. The baseline results indicate that the RF achieves strong classification performance despite the limited number of MITM samples, suggesting that the dataset's features are inherently more distinctive, which facilitates more accurate class separation. The baseline G-mean, approaching 1, indicates a well-balanced ability to identify both positive and negative instances.

Overall, upsampling leads to only marginal improvements, accompanied by trade-offs across evaluation metrics, similar to the patterns observed with the UNSW-NB15 dataset, albeit on a smaller scale. In terms of macro F1-score, ADASYN, KMeans-SMOTE, and B-SMOTE perform worse than SMOTE-NC, CVAE, and CWGAN-GP, despite achieving slight gains in G-mean. While CVAE and CWGAN-GP yield modest improvements in both F1-score and AUPRC compared to the baseline, their G-mean scores decrease slightly. This outcome suggests that traditional and hybrid oversampling methods primarily enhance recall, thereby improving G-mean, whereas deep generative models tend to strengthen precision and ranking capability but at the expense of a slight reduction in recall. Consistent with prior observations on the UNSW-NB15 dataset, this reflects a fundamental trade-off between optimizing minority-class detection and maintaining overall class balance.

Although balancing all classes yields a slightly higher macro F1-score compared to upsampling only the MITM class, the improvement is minimal and statistically insignificant. Deep learning–based oversampling methods, however, exhibit more stable and consistent performance across metrics relative to traditional approaches. Among them, CWGAN-GP attains the highest F1-score (0.9863) and AUPRC (0.9970) when all classes are balanced. Overall, RF achieves better performance when all classes are oversampled rather than when only the MITM class is upsampled, particularly in terms of F1-score.

Table 20. RF classification results for the TON_IoT dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced
Macro F1-score	0.9858	0.9856	0.9858	0.9839	0.9842	0.9846	0.9851	0.9841	0.9841	0.9856	0.9860	0.9861	0.9863	0.9850	0.9853
Macro AUPRC	0.9968	0.9954	0.9947	0.9956	0.9953	0.9964	0.9962	0.9958	0.9951	0.9969	0.9964	0.9968	0.9970	0.9962	0.9958
G-mean	0.9902	0.9931	0.9929	0.9927	0.9925	0.9918	0.9916	0.9926	0.9922	0.9898	0.9900	0.9899	0.9894	0.9917	0.9914

Table 21 reports the performance of the XGBoost classifier on the TON_IoT dataset across both baseline and upsampled configurations. Leveraging its error-correcting boosting mechanism, XGBoost achieves slightly stronger baseline performance than RF, indicating its enhanced ability to capture subtle patterns that RF may have overlooked. In contrast to the RF results, upsampling only the MITM class yields a higher macro F1-score than balancing all classes across various oversampling methods. This suggests that oversampling non-MITM classes may introduce noise or lead to overfitting, slightly diminishing overall F1 performance. Consequently, focusing on minority-class upsampling appears more effective and also more computationally efficient than balancing the entire dataset.

Among all methods, the CVAE applied under the MITM-only upsampling strategy delivers the best overall performance, showing improvements across all three evaluation metrics and achieving the highest F1-score and AUPRC. SMOTE-NC ranks second, offering a more competitive G-mean compared to CVAE. Deep generative models such as CVAE and CWGAN-GP also offer a marginal advantage in ranking performance, as reflected in their higher AUPRC values. Conversely, ADASYN and B-SMOTE consistently underperform relative to other oversampling approaches under both sampling strategies.

For XGBoost, resampling provides clear benefits, with CVAE and SMOTE-NC achieving the best balance between performance and computational complexity when applied to minority-focused upsampling. If maximizing G-mean is the primary objective, the traditional SMOTE-NC method is preferable. However, if the goal is to enhance overall classification performance and ranking ability, the CVAE-based approach represents the superior choice.

Table 21. XGBoost classification results for the TON_IoT dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced
Macro F1-score	0.9874	0.9880	0.9873	0.9867	0.9842	0.9870	0.9865	0.9854	0.9840	0.9883	0.9882	0.9870	0.9847	0.9871	0.9858
Macro AUPRC	0.9977	0.9977	0.9973	0.9976	0.9966	0.9975	0.9968	0.9975	0.9967	0.9979	0.9976	0.9979	0.9974	0.9977	0.9971
G-mean	0.9916	0.9954	0.9941	0.9934	0.9899	0.9921	0.9903	0.9920	0.9901	0.9917	0.9925	0.9897	0.9886	0.9924	0.9909

Table 22 compares the baseline and upsampled performance of the 1D-CNN classifier on the TON_IoT dataset. The baseline results demonstrate strong overall performance, although not as competitive as RF or XGBoost without extensive hyperparameter tuning and careful architectural optimization. The 1D-CNN generally benefits from MITM-only upsampling across most resampling methods in terms of F1-score, except for SMOTE-NC and KMeans-SMOTE. Specifically, SMOTE-NC generates smoother synthetic samples through interpolation across all classes, while KMeans-SMOTE produces samples that better preserve local data distributions. This balanced and less noisy augmentation is particularly advantageous for CNN models.

Under targeted upsampling of only the MITM class, SMOTE-NC consistently achieves strong results, while CWGAN-GP performs reasonably well but with a slightly lower G-mean. Targeted oversampling tends to outperform balanced oversampling for deep generative methods, suggesting that focusing augmentation on specific minority classes is more effective than uniformly balancing all classes. In contrast, balanced ADASYN and B-SMOTE rank among the weakest performers, as they often introduce noisy or borderline samples that cause CNN filters to capture spurious local patterns, thereby reducing generalization capability.

Given the distinctive nature of the TON_IoT dataset features and the CNN's ability to capture rare local attack signatures through convolutional filters, the model is able to detect rare attacks effectively despite the limited number of samples. While oversampling produces only marginal improvements, these gains remain meaningful. However, CNNs are also more sensitive to unrealistic or borderline synthetic samples, which can amplify local noise and degrade performance. Overall, for 1D-CNN, SMOTE-NC delivers the most effective performance under balanced upsampling and is the preferred method in this scenario.

Table 22. 1D-CNN classification results for the TON_IoT dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced
Macro F1-score	0.9533	0.9551	0.9571	0.9522	0.9412	0.9507	0.9537	0.9517	0.9405	0.9523	0.9432	0.9543	0.9435	0.9527	0.9465
Macro AUPRC	0.9862	0.9865	0.9874	0.9846	0.9797	0.9836	0.9844	0.9855	0.9803	0.9846	0.9804	0.9852	0.9794	0.9850	0.9819
G-mean	0.9543	0.9585	0.9624	0.9580	0.9450	0.9554	0.9588	0.9572	0.9433	0.9544	0.9429	0.9583	0.9457	0.9570	0.9497

Table 23 compares the baseline and upsampled performance of the MLP classifier on the TON_IoT dataset. The baseline MLP slightly outperforms 1D-CNN but does not offer a significant advantage over tree-based classifiers. Similar to 1D-CNN, MLP generally benefits more from MITM-only upsampling than from upsampling all classes, with the exceptions of SMOTE-NC and KMeans-SMOTE. Overall, deep learning–based oversampling methods perform better with MITM-only upsampling, whereas traditional and hybrid methods show mixed results.

Among the evaluated oversampling methods, CWGAN-GP targeting the MITM class achieves the highest F1-score and ranks first across all evaluation metrics, followed by CVAE. Both deep generative methods show a notable drop in F1-score when moving from MITM-only to balanced upsampling, suggesting that balancing all classes introduces noise or artifacts that reduce the MLP's generalization.

Balanced upsampling is detrimental to MLP performance across all metrics and methods compared to the MITM-only strategy. These results suggest that the default MLP architecture is effective, and that excessive synthetic data from fully balancing all classes can introduce harmful noise that reduces classifier performance.

Table 23. MLP classification results for the TON_IoT dataset.

	Baseline	SMOTE-NC		ADASYN		KMeans-SMOTE		B-SMOTE		CVAE		CWGAN-GP		Oversampling Average	
		MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced	MITM	Balanced
Macro F1-score	0.9574	0.9565	0.9591	0.9534	0.9496	0.9552	0.9560	0.9556	0.9468	0.9577	0.9439	0.9605	0.9408	0.9565	0.9494
Macro AUPRC	0.9879	0.9862	0.9863	0.9846	0.9821	0.9852	0.9856	0.9853	0.9804	0.9866	0.9798	0.9886	0.9808	0.9861	0.9825
G-mean	0.9600	0.9591	0.9611	0.9599	0.9537	0.9598	0.9620	0.9608	0.9499	0.9598	0.9422	0.9633	0.9398	0.9605	0.9515

In summary, despite the dataset's inherent class imbalance, all classifiers demonstrate satisfactory performance even without the application of oversampling techniques. This suggests that the dataset's features are well-structured and sufficiently discriminative, resulting in only modest performance improvements from resampling. Among the evaluated classifiers, XGBoost consistently emerges as the most reliable model, achieving strong baseline

performance and superior average metrics compared with the others. Regarding oversampling methods, CVAE produces near-optimal results across most scenarios, while CWGAN-GP demonstrates high potential but remains sensitive to the choice of classifier architecture. SMOTE-NC offers an effective balance between simplicity and accuracy, often achieving the highest G-mean. In contrast, ADASYN and B-SMOTE consistently underperform and are therefore not recommended for this dataset. The minority-only upsampling strategy (MITM-only) is generally preferred, as it surpasses full balancing in most cases while also being more computationally efficient. This observation is further supported by the minority-only upsampling approach consistently achieving higher average performance metrics than the balanced strategy across different classifiers. Although a trade-off between G-mean and F1-score or AUPRC persists, it is less severe than that observed in the UNSW-NB15 dataset.

7. Conclusions and Future Work

7.1. Conclusions

This research contributes to the field of intrusion detection by providing a comprehensive comparative analysis of traditional and deep learning–based oversampling methods across two benchmark NIDS datasets: UNSW-NB15 and TON_IoT. Multiple classifiers and resampling strategies, including deep generative approaches that have not been extensively examined in previous studies, were systematically evaluated to determine their effectiveness in improving minority class detection. The findings highlight practical trade-offs between model interpretability, stability, and minority detection performance, offering guidance for selecting appropriate resampling strategies in real-world imbalanced scenarios.

The results from the UNSW-NB15 dataset showed that both class imbalance and class overlap significantly influence model performance, particularly for overlapping classes that are also affected by imbalance. Although resampling techniques helped mitigate class overlap and improved the stability of certain models, their overall benefits were limited without the inclusion of additional discriminative features or specialized learning frameworks. Among the evaluated methods, KMeans-SMOTE provided the most consistent and interpretable improvements, enhancing both the G-mean and F1-score by expanding the minority feature space and reducing class bias. In comparison, deep generative oversampling methods such as CVAE and CWGAN GP demonstrated strong potential but also high variability. These methods occasionally achieved strong F1-scores but sometimes failed completely in terms of G-mean, indicating instability and possible overfitting to synthetic samples. Nevertheless, CVAE and CWGAN GP frequently ranked among the top performers in terms of F1-score, showing promising results when properly tuned or combined with robust classifiers. The findings also indicate that without appropriate mechanisms to handle class overlap, the performance of oversampling methods is hindered.

In the TON_IoT dataset, the classifiers achieved relatively strong performance even without oversampling, suggesting that the dataset's inherent feature representations are sufficiently discriminative. Among the oversampling methods, CVAE achieved near-optimal F1-scores in most settings, while SMOTE-NC provided a good balance between simplicity, stability, and effectiveness. The minority-only resampling strategy was also found to be advantageous, often outperforming full balancing approaches while remaining computationally efficient. Across both datasets, XGBoost consistently emerged as the most reliable and stable model, demonstrating strong baseline performance and maintaining robustness under different resampling configurations.

These findings highlight the importance of conducting a thorough baseline analysis before applying oversampling to imbalanced datasets. Understanding the dataset's intrinsic properties, such as class overlap or feature redundancy, is essential for selecting appropriate mitigation strategies. Based on this understanding, it is recommended to apply targeted oversampling only to the most underrepresented classes rather than applying full balancing, which may introduce noise and overlapping samples that negatively affect overall performance.

Overall, this analysis emphasizes that the choice of oversampling technique and classifier architecture should be carefully aligned with the characteristics of the dataset. Traditional and hybrid oversampling methods deliver stable and dependable improvements, while deep generative approaches show considerable promise for future research, particularly as their stability and generalization capabilities continue to advance.

7.2. Future Work

Future research can further build upon these findings in several directions:

- Addressing class overlap: Developing a classification framework or incorporating additional discriminative features to mitigate the class overlap issue observed in the UNSW-NB15 dataset.
- Improving feature consistency in deep generative models: Current VAE- and GAN-based oversampling methods often suffer from feature misalignment, where generated samples fail to retain key statistical properties of real network traffic. Investigating the integration of specialized loss functions, such as feature perceptual

loss, could help preserve feature integrity and enhance synthetic data quality.

- Combining VAE and GAN architectures: Since both approaches have distinct strengths and limitations, integrating them into a unified model could leverage the advantages of each, producing higher-quality and more diverse synthetic samples.

By exploring these directions, future studies can further improve the detection of rare and evolving network attacks, leading to more robust and reliable intrusion detection systems capable of adapting to real-world cybersecurity challenges.

Author Contributions

H.M.D.: conceptualization, methodology, investigation, writing—original draft preparation; W.Z.: supervision, writing—reviewing and editing; Y.-W.C.: supervision, writing—reviewing and editing. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The datasets used in this study are publicly available and can be accessed at the following links: UNSW-NB15 and TON_IoT. The source code and datasets produced in this study are publicly available here.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

No AI tools were utilized for this paper.

References

1. Garcia-Teodoro, P.; Diaz-Verdejo, J.; Maciá-Fernández, G.; et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* **2009**, *28*, 18–28.
2. Khraisat, A.; Gondal, I.; Vamplew, P.; et al. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20.
3. Schrötter, M.; Niemann, A.; Schnor, B. A comparison of neural-network-based intrusion detection against signature-based detection in IoT networks. *Information* **2024**, *15*, 164.
4. Ahmad, R.; Alsmadi, I.; Alhamdani, W.; et al. Zero-day attack detection: a systematic literature review. *Artif. Intell. Rev.* **2023**, *56*, 10733–10811.
5. Hwang, R.H.; Peng, M.C.; Huang, C.W.; et al. An unsupervised deep learning model for early network traffic anomaly detection. *IEEE Access* **2020**, *8*, 30387–30399.
6. Ma, W. Analysis of anomaly detection method for Internet of things based on deep learning. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3893.
7. Van, N.T.; Thinh, T.N.; Sach, L.T. An anomaly-based network intrusion detection system using deep learning. In Proceedings of the International Conference on System Science and Engineering (ICSSE), Ho Chi Minh City, Vietnam, 21–23 July 2017; pp. 210–214.
8. Bace, R.G.; Mell, P. Intrusion Detection Systems. 2001. Available online: https://all.net/books/standards/NIST-CSRC/csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf (accessed on 1 November 2025)
9. Dong, B.; Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In Proceedings of the 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016; pp. 581–585.
10. Rodda, S.; Erothi, U.S.R. Class imbalance problem in the network intrusion detection systems. In Proceedings of the

International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, India, 3–5 March 2016; pp. 2685–2688.
11. Rani, M.; Gagandeep. Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications. *Multimed. Tools Appl.* **2022**, *81*, 8499–8518.
12. Longadge, R.; Dongre, S. Class imbalance problem in data mining review. *arXiv* **2013**, arXiv:1305.1707.
13. Al-Qarni, E.A.; Al-Asmari, G.A. Addressing imbalanced data in network intrusion detection: A review and survey. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 136–143.
14. Tran, N.; Chen, H.; Jiang, J.; et al. Effect of class imbalance on the performance of machine learning-based network intrusion detection. *Int. J. Perform. Eng.* **2021**, *17*, 741.
15. Wheelus, C.; Bou-Harb, E.; Zhu, X. Tackling class imbalance in cyber security datasets. In Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 6–9 July 2018; pp. 229–232.
16. Kudithipudi, S.; Narisetty, N.; Kancherla, G.R.; et al. Evaluating the Efficacy of Resampling Techniques in Addressing Class Imbalance for Network Intrusion Detection Systems Using Support Vector Machines. *Ing. Des Syst. Inf.* **2023**, *28*, 1229.
17. Rahma, F.; Rachmadi, R.F.; Pratomo, B.A.; et al. Assessing the Effectiveness of Oversampling and Undersampling Techniques for Intrusion Detection on an Imbalanced Dataset. In Proceedings of the IEEE Industrial Electronics and Applications Conference (IEACon), Penang, Malaysia, 6–7 November 2023; pp. 92–97.
18. Bagui, S.; Li, K. Resampling imbalanced data for network intrusion detection datasets. *J. Big Data* **2021**, *8*, 6.
19. Ahmed, H.A.; Hameed, A.; Bawany, N.Z. Network intrusion detection using oversampling technique and machine learning algorithms. *PeerJ Comput. Sci.* **2022**, *8*, e820.
20. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; et al. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357.
21. Derhab, A.; Aldweesh, A.; Emam, A.Z.; et al. Intrusion detection system for internet of things based on temporal convolution neural network and efficient feature engineering. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 6689134.
22. Bulavas, V.; Marcinkevičius, V.; Rumiński, J. Study of multi-class classification algorithms' performance on highly imbalanced network intrusion datasets. *Informatica* **2021**, *32*, 441–475.
23. Omari, K.; Taoussi, C.; Oukhatar, A. Comparative Analysis of Undersampling, Oversampling, and SMOTE Techniques for Addressing Class Imbalance in Phishing Website Detection. *Int. J. Adv. Comput. Sci. Appl.* **2025**, *16*, 751–757.
24. He, H.; Bai, Y.; Garcia, E.A.; et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008; pp. 1322–1328.
25. Liu, J.; Gao, Y.; Hu, F. A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. *Comput. Secur.* **2021**, *106*, 102289.
26. Wang, Z.; Jiang, D.; Huo, L.; et al. An efficient network intrusion detection approach based on deep learning. *Wirel. Netw.* **2021**. https://doi.org/10.1007/s11276-021-02698-9
27. Pan, L.; Xie, X. Network intrusion detection model based on PCA+ ADASYN and XGBoost. In Proceedings of the 3rd International Conference on E-Business, Information Management and Computer Science, Wuhan, China, 5–6 December 2020; pp. 44–48.
28. Douzas, G.; Bacao, F.; Last, F. Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Inf. Sci.* **2018**, *465*, 1–20.
29. Wu, T.; Fan, H.; Zhu, H.; et al. Intrusion detection system combined enhanced random forest with SMOTE algorithm. *Eurasip J. Adv. Signal Process.* **2022**, *2022*, 39.
30. Talukder, M.A.; Khalid, M.; Sultana, N. A hybrid machine learning model for intrusion detection in wireless sensor networks leveraging data balancing and dimensionality reduction. *Sci. Rep.* **2025**, *15*, 4617.
31. Priyadarsini, P.I.; Leela, P.S.; Jyothi, B. Towards intelligent machine learning models for intrusion detection system. *Turk. J. Comput. Math. Educ.* **2021**, *12*, 643–655.
32. Han, H.; Wang, W.Y.; Mao, B.H. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *Advances in Intelligent Computing*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 878–887.
33. Zhang, J.; Zhang, Y.; Li, K. A network intrusion detection model based on the combination of relieff and borderline-smote. In Proceedings of the 2020 4th High Performance Computing and Cluster Technologies Conference & 2020 3rd International Conference on Big Data and Artificial Intelligence, Qingdao, China, 3–6 July 2020; pp. 199–203.
34. Sun, Y.; Que, H.; Cai, Q.; et al. Borderline smote algorithm and feature selection-based network anomalies detection strategy. *Energies* **2022**, *15*, 4751.
35. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 4 November 2025).
36. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
37. Doersch, C. Tutorial on variational autoencoders. *arXiv* **2016**, arXiv:1606.05908.
38. Sohn, K.; Lee, H.; Yan, X. Learning structured output representation using deep conditional generative models. *Adv. Neural Inf. Process. Syst.* **2015**, *28*.

39. Xu, X.; Li, J.; Yang, Y.; et al. Toward effective intrusion detection using log-cosh conditional variational autoencoder. *IEEE Internet Things J.* **2020**, *8*, 6187–6196.
40. Yang, Y.; Zheng, K.; Wu, C.; et al. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors* **2019**, *19*, 2528.
41. Azmin, S.; Islam, A.M.A.A. Network intrusion detection system based on conditional variational laplace autoencoder. In Proceedings of the 7th International Conference on Networking, Systems and Security, Dhaka, Bangladesh, 22–24 December 2020; pp. 82–88.
42. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; et al. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
43. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 214–223.
44. Gulrajani, I.; Ahmed, F.; Arjovsky, M.; et al. Improved training of wasserstein gans. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
45. Mu, Z.; Shi, X.; Dogan, S. Information system security reinforcement with wgan-gp for detection of zero-day attacks. In Proceedings of the 7th International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 24–27 May 2024; pp. 105–110.
46. Zhang, L.; Jiang, S.; Shen, X.; et al. PWG-IDS: An intrusion detection model for solving class imbalance in IIoT networks using generative adversarial networks. *arXiv* **2021**, arXiv:2110.03445.
47. Tavallaee, M.; Bagheri, E.; Lu, W.; et al. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
48. Moustafa, N.; Slay, J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.
49. Moustafa, N. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustain. Cities Soc.* **2021**, *72*, 102994.
50. Moustafa, N.; Slay, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Inf. Secur. Journal: Glob. Perspect.* **2016**, *25*, 18–31.
51. Booij, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; Den Hartog, F.T. ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet Things J.* **2021**, *9*, 485–496.
52. Zoghi, Z.; Serpen, G. UNSW-NB15 computer security dataset: Analysis through visualization. *Secur. Priv.* **2024**, *7*, e331.
53. Salman, T.; Bhamare, D.; Erbad, A.; et al. Machine learning for anomaly detection and categorization in multi-cloud environments. In Proceedings of the IEEE 4th international conference on cyber security and cloud computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 97–103.