Journal of King Saud University Computer and Information Sciences (2025) 37:7
https://doi.org/10.1007/s44443-025-00015-4

RESEARCH ARTICLE

# ADHS-EL: Dynamic ensemble learning with adversarial augmentation for accurate and robust network intrusion detection

Huajuan Ren[1] · Yonghe Tang[1] · Shuai Ren[1] · Ruimin Wang[1] · Weiyu Dong[1]

Received: 12 July 2024 / Accepted: 31 January 2025


**Abstract**
Machine learning techniques have been increasingly applied to network intrusion detection. Class imbalance and adversarial attacks are two pivotal challenges that hinder the accuracy and robustness of machine learning-based network traffic detection models. However, current research efforts focus solely on addressing one of these issues, which limits the improvement of detection performance. To jointly tackle these challenges, we propose a dynamic ensemble learning method called ADHS-EL. Specifically, a novel sampling mechanism is integrated into the Boosting ensemble framework, providing a balanced data subset mixed with adversarial samples for each iteration to yield a strong ensemble. This sampling mechanism includes dynamic hybrid sampling for balancing benign and malicious traffic, as well as boundary malicious traffic adversarial augmentation for introducing adversarial samples to the training dataset. A boundary samples importance enhancement strategy is adopted in hybrid sampling and adversarial augmentation, which contributes to improving intrusion detection accuracy and preventing robust overfitting. Experimental evaluations on two public datasets demonstrate that the proposed method exhibits significant advantages in terms of accuracy and robustness. In the non-adversarial test, ADHS-EL obtains the highest F1 score on both datasets. In the adversarial test involving four types of adversarial attacks, ADHS-EL achieves the average detection rates of 74.35% and 80.30% on the two datasets, respectively, marking improvements of 10.63% and 13.26% over the optimal baseline methods.

**Keywords** Intrusion detection · Class imbalance · Adversarial attacks · Dynamic hybrid sampling · Adversarial augmentation · Ensemble learning

## 1 Introduction

Cybersecurity incidents occur frequently with the rapid development of network technology, making it increasingly necessary to develop advanced network intrusion detection schemes to mitigate security threats. Machine learning is widely applied to facilitate innovations in network intrusion detection (Li and Abdallah 2023). Utilizing machine learning enables automatic learning and network traffic detection, which offers better scalability and adaptability compared to traditional rule-based detection methods.

The performance of machine learning-based network intrusion detection models is affected by the network traffic data and detection algorithms used in training (Engelen et al. 2021). The class imbalance of network traffic data biases the decision boundaries obtained by standard machine learning, resulting in low detection accuracy for malicious traffic. Additionally, adversarial machine learning attacks (referred to as adversarial attacks) against malicious traffic may bypass detection. Figure 1 illustrates these two problems. As in Fig. 1(a), when the traffic data is imbalanced, the malicious records are lacking in the training dataset, and so the ground truth decision boundary ( shown as a dashed line) deviates from the learned boundary (shown as a solid line). Due to the decision bias, the adversary applies

✉ Huajuan Ren
huajuanren@163.com

Yonghe Tang
tyh_983@126.com

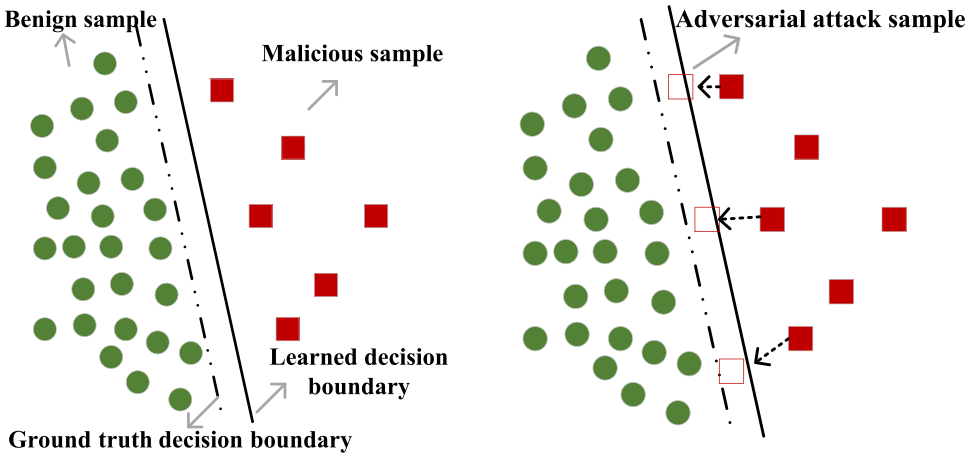
Shuai Ren
renshuai94@outlook.com

Ruimin Wang
wrm2013nian@163.com

Weiyu Dong
dongxinbaoer@163.com

[1] Information Engineering University, Zhengzhou 450001, China

Published online: 24 March 2025




(a) Class imbalance. (b) Adversarial attacks.

**Fig. 1** Illustration of Class imbalance and adversarial attacks

small perturbations to the malicious samples to cross the actual decision boundary with imperceptible changes, resulting in the malicious samples being misclassified as benign, as shown in Fig. 1(b). These natural problems make constructing accurate and robust intrusion detection models from network traffic datasets challenging.

Some solutions have been proposed to address the class imbalance and adversarial attack problems in network intrusion detection. The approaches to solving the class imbalance problem include resampling, feature representation, and cost-sensitive learning. Resampling, which utilizes undersampling, oversampling, or hybrid sampling to balance the original traffic distribution, is simple to implement and flexible in applying different classifiers, having been used in several research efforts (Khan et al. 2019; Liu et al. 2021; Shin et al. 2024; Ren et al. 2023). Feature representation and cost-sensitive learning usually require expert assistance (e.g., careful construction of features and cost matrices), making it difficult to generalize across different tasks (Liu et al. 2020). The methods for defending against adversarial attacks include adversarial training, model integration, and adversarial detection. Adversarial training incorporates a mixture of original and adversarial samples into the training process, where the detection model gradually improves its robustness against adversarial attacks by learning the adversarial characteristics during training (Abou Khamis et al. 2020; Roshan et al. 2024; Usama et al. 2019). Model integration achieves adversarial attack defense by employing sophisticated classification mechanisms (Jiang et al. 2022; Paya et al. 2024). Adversarial detection detects adversarial samples by training additional models (Pawlicki et al. 2020; Mohanty et al. 2022). Among these methods, adversarial training is more straightforward than other methods in teaching adversarial patterns and is an effective way to enhance model robustness (Zhang et al. 2020a).

It is worth noting that the class imbalance and adversarial attack problems are considered in isolation in the current research, meaning that model accuracy and robustness may not be obtained together. In this paper, we focus on improving the accuracy and robustness of machine learning-based network intrusion detection models by addressing these two challenges. We consider combining resampling and adversarial training to solve the problems at the data level, which has the benefit of being easily scalable to intrusion detection models constructed by different algorithms. Specifically, we propose ADHS-EL (an Ensemble Learning based on Dynamic Hybrid Sampling with Adversarial augmentation). In this method, a dynamic hybrid sampling with adversarial augmentation is designed to be involved in the training and integration of the intrusion detection models. First, classification hardness is employed to identify the sample types, and dynamic hybrid sampling, i.e., dynamic undersampling for benign traffic and dynamic oversampling for malicious traffic, is implemented with the guiding principle of enhancing the importance of boundary samples. Then, for the boundary malicious samples obtained from the dynamically hybrid sampled dataset, their adversarial versions are generated using a well-trained generator based on the improved AdvGAN. AdvGAN (Xiao et al. 2018) is a technique that utilizes Generative Adversarial Networks (GAN) (Goodfellow et al. 2014b) to generate adversarial examples by training a generator to produce samples that can deceive the target model while maintaining visual similarity to the original samples. Finally, the dynamic hybrid sampling datasets with adversarial samples are fed into the Boosting framework for iterative training. The main contributions of this paper are as follows:

- A boundary malicious traffic adversarial augmentation strategy is designed. Based on this strategy, adversarial samples crucial for decision-making are added to the

training set, which helps mitigate the robust overfitting caused by introducing all adversarial samples.

- A dynamic hybrid sampling method with adversarial augmentation is proposed. It provides a balanced dataset mixed with adversarial samples, mitigating the low detection accuracy caused by the class imbalance in network traffic while benefiting the learning of adversarial attack characteristics and thus improving model robustness.
- An ensemble learning approach is suggested based on the proposed sampling method. The ensemble method learns complex network traffic distributions by sampling and training iteratively, enabling accurate and robust detection models.
- The model is tested in non-adversarial and adversarial environments, where five accuracy metrics and adversarial sample detection rates (under four adversarial attacks) are employed for evaluation, respectively. Extensive experimental analysis illustrates the advantages of ADHS-EL in terms of accuracy and robustness compared to the baseline methods.

The rest of the paper is organized as follows. Section 2 introduces the related works on class imbalance handling and adversarial attack defense in machine learning-based network intrusion detection. Section 3 describes the proposed methodology in detail. Section 4 presents the experimental results and analyzes the effectiveness of the proposed method. Section 5 discusses the practical application and limitations of the proposed method. Finally, we conclude the paper in Sect. 6.

## 2 Related works

This section presents related works on machine learning algorithms for building intrusion detection models, solutions to the class imbalance problem, adversarial attacks faced by network intrusion detection models based on machine learning, and adversarial attack defense methods.

### 2.1 Machine learning-based network intrusion detection

Network intrusion detection is generally abstracted as the classification problem in machine learning, where labeled traffic samples are utilized to train the detection model that identifies the type of unlabeled traffic samples. Some classical machine learning algorithms have been used for network intrusion detection with good performance, such as Decision Tree (DT) (Zhao et al. 2021), Random Forest (RF) (Korium et al. 2024), K-Nearest Neighbor (KNN) (Li et al. 2014), and Multi-Layer Perceptron (MLP) (Tian et al. 2021). Deep learning has also been applied to network intrusion detection, improving the accuracy of models to some extent due to its powerful characterization capabilities, such as Deep Neural Network (DNN) (Vinayakumar et al. 2019), Convolutional Neural Network (CNN) (Vinayakumar et al. 2017a), Recurrent Neural Network (RNN) (Vinayakumar et al. 2017b) and their variants (Wang et al. 2023).

Deep learning solutions are not consistently outperforming classical machine learning-based solutions in the cybersecurity field, and classical machine learning algorithms such as DT or RF are not only lightweight in computational requirements but also adequate in classification performance (Lichy et al. 2023; Casas et al. 2019). The existing research efforts have explored the robustness of deep learning models to adversarial attacks, neglecting the research of classical machine learning in this direction. This paper focuses on enhancing the robustness of classical machine learning detection models, which is of practical value in cybersecurity applications.

### 2.2 Class imbalance problem and solutions

The class imbalance problem exists in real network traffic, i.e., benign traffic tends to overwhelm malicious traffic in quantity. Traditional machine learning tends to optimize for overall accuracy, resulting in malicious traffic often being ignored due to the excessive number of benign traffic samples, thereby affecting the accuracy of intrusion detection.

Some approaches have been proposed to alleviate the class imbalance problem in network traffic, mainly including resampling, feature representation, and cost-sensitive learning. Resampling balances the traffic distribution by undersampling, oversampling, or hybrid sampling. Feature representation improves model accuracy in class-imbalanced situations by designing more relevant features for the detection task. The main idea of cost-sensitive learning is to set the misclassification cost of minority class samples to be greater than that of majority class samples, mitigating the preference of the classifier. The latter two solutions usually require experts to construct task-specific features and cost matrices beforehand, which is inflexible in real-world problems.

Resampling is simple to implement and flexible in applying different classifiers, making it the preferred method in most works. (Khan et al. 2019) balanced the distributions of the malicious and benign classes by the oversampling method, Synthetic Minority Oversampling Technique (SMOTE). Liu et al. (2021) utilized Adaptive Synthetic Sampling (ADASYN) to oversample malicious class samples. Shin et al. (2024) employed hybrid sampling, SMOTE-Tomek (SMOTE with TomekLinks), to append minority class samples and remove pairs of indistinguishable majority and minority class samples in the feature space. Zhao et al. (2024) utilized Wasserstein GAN (WGAN) (Arjovsky

et al. 2017) and Conditional Tabular GAN (CTGAN) (Xu et al. 2019) to generate network traffic samples targeting anomalous activities and integrated these newly generated samples into the original dataset to improve the performance of intrusion detection. Dynamic undersampling-based ensemble (DUEN) was proposed in Ren et al. (2023), which utilizes training feedback information to execute dynamic undersampling for handling traffic imbalance, achieving high detection accuracy. In our proposed ADHS-EL, this dynamic undersampling technique is improved, and dynamic hybrid sampling is introduced to handle benign and malicious traffic.

### 2.3 Adversarial attacks in network intrusion detection

Machine learning-based network intrusion detection models are faced with threats from adversarial attacks where the model's accuracy, integrity, or security is compromised. Depending on the stage at which the attack is launched, adversarial attacks can be categorized as evasion, poisoning, extraction, and inference attacks. In this paper, we focus on improving the robustness of the model against evasion attacks.

Based on the attacker's level of knowledge about the target model, adversarial attacks can be classified into white-box attacks and gray/black-box attacks. White-box attacks assume that the attacker has complete knowledge about the target model and training data, allowing the attacker to perform complex attacks and thus avoid detection. Some of the common white-box attack methods are Fast Gradient Sign Method (FGSM) (Goodfellow et al. 2014a), Projected Gradient Descent (PGD) (Aleksander et al. 2019), Carlini & Wagner (C&W) (Carlini and Wagner 2017), DeepFool (Moosavi-Dezfooli et al. 2016), DecisionTreeAttack (DTA) (Papernot et al. 2016), and so on. The fact that a comprehensive knowledge of the system and its vulnerabilities is often impossible makes white-box attacks very unrealistic in many practical scenarios. Moreover, these attacks are effective in computer vision, but have certain limitations in network intrusion detection (Zhang et al. 2020a).

Gray/black-box attacks do not require knowledge about the target model and are commonly known as ZooAttack (ZOO) (Chen et al. 2017), HotSkipJumpAttack (HSJA) (Chen et al. 2020), and BoundaryAttack (BA) (Brendel et al. 2017). ZOO is a score-based attack that estimates the gradient to create adversarial traffic in a gray/black-box setup. BA and HSJA are decision-based attacks that use only binary feedback to create adversarial inputs. In addition, GAN is an effective tool for generating adversarial attacks due to its powerful ability to imitate real data distributions. GAN-based attacks can be classified as gray/black box attacks when the binary feedback from the intrusion detection model is used to train the adversarial samples generator. Lin et al. (2022) proposed generating adversarial samples based on WGAN, where the discriminator simulates a black-box intrusion detection model, and the generator is used to convert the original malicious traffic samples into adversarial malicious traffic samples, and experiments show that the generated adversarial attacks are effective. Two variants of WGAN and AdvGAN were explored in Duy et al. (2023) for their ability to generate attack samples to bypass detectors, where experimental results show that AdvGAN retains more similar features to the original samples and easily evades detection. Comparatively, gray/black-box attacks are more practical, but the effectiveness of the attacks is also restricted.

### 2.4 Defense methods for adversarial attacks

Adversarial defenses are employed to mitigate the security threat posed by adversarial attack samples. Representative defense methods are adversarial training, model integration, and adversarial detection, where the first two work in the model training phase and the latter in the testing phase.

Adversarial training improves model robustness against adversarial samples by adding pre-constructed adversarial samples to the training set. Abou Khamis et al. (2020) and Roshan et al. (2024) utilized classical white-box adversarial attacks (such as FGSM, PGD) to generate adversarial examples for adversarial training. Usama et al. (2019) merged GAN-based adversarial attacks with clean samples to co-train the intrusion detection model for enhancing robustness. These studies do not perform a targeted selection of the adversarial samples involved in model training, which may result in robust overfitting and thus degrade the detection performance on clean samples.

Model integration combines multiple detection models to improve the robustness of the detection model, thereby defending against adversarial samples. Feature grouping and multiple model fusion were presented in Jiang et al. (2022) to defend against adversarial attacks. Paya et al. (2024) proposed an integration method called Apollon, which uses multi-armed bandits to select the best classifier or combination of classifiers for enabling adversarial attack defense without compromising the original network traffic identification performance. These model integration methods increase model complexity, and the training strategies tend to be specific to the models constructed by certain algorithms.

Adversarial detection refers to placing an additional detector before the target model to detect adversarial samples. Pawlicki et al. (2020) trained the detector to recognize adversarial samples on an artificial neural network-based intrusion detection model using all neuron outputs as training samples. Mohanty et al. (2022) detected adversarial examples based on the reconstruction error obtained from an autoencoder with a fixed threshold. Adversarial detection can maintain the general performance of the model, but it increases the cost of

**Table 1** Comparison of related intrusion detection methods

| Reference | Dataset | Detection model | Class imbalance handling | Adversarial attack | Adversarial defense | Performance evaluation |
|---|---|---|---|---|---|---|
| Khan et al. (2019) | KDD99, UNSW-NB15 | DL | ✓ | – | – | No-adv |
| Liu et al. (2021) | NSL-KDD, CIC-IDS2017 | ML | ✓ | – | – | No-adv |
| Shin et al. (2024) | IoTID20, CHADC2020 | ML | ✓ | – | – | No-adv |
| Zhao et al. (2024) | CIC-IDS2017 | ML | ✓ | – | – | No-adv |
| Ren et al. (2023) | NSL-KDD, UNSW-NB15 | ML | ✓ | – | – | No-adv |
| Abou Khamis et al. (2020) | UNSW-NB15 | DL | – | White-box | ✓ | Adv |
| Roshan et al. (2024) | CIC-IDS2017 | DL | – | White-box | ✓ | Adv |
| Usama et al. (2019) | NSL-KDD | ML | – | White-box | ✓ | Adv |
| Jiang et al. (2022) | MedBIoT, IoTID | DL | – | White-box | ✓ | No-adv, Adv |
| Paya et al. (2024) | NSL-KDD | ML | – | Black-box | ✓ | No-adv, Adv |
| Pawlicki et al. (2020) | CIC-IDS2017 | ML | – | White-box | ✓ | Adv |
| Mohanty et al. (2022) | CIC-Darknet2020 | ML | – | White-box | ✓ | Adv |
| This paper | NSL-KDD, CIC-IDS2017 | ML | ✓ | Black, White-box | ✓ | No-adv, Adv |

* Note: ML: Machine Learning, DL: Deep Learning, No-adv: Testing in non-adversarial environment, Adv: Testing in adversarial environment

training the additional detection model and may potentially impact the efficiency and practicality of the target detection model.

These adversarial defense methods can reduce the impact of adversarial attacks to some extent. Among them, adversarial training is simpler in teaching adversarial patterns and can be extended to different models. However, the existing adversarial training methods suffer from robust overfitting, causing decreased detection rates of retrained models on clean samples. In our proposed method, this problem is mitigated, enhancing model robustness while ensuring accuracy.

### 2.5 Summary

The relevant intrusion detection methods are summarized in Table 1, highlighting the differences between our proposed method and existing research. We focus on improving the accuracy and adversarial robustness of machine learning-based detection models. This is achieved by jointly addressing class imbalance and adversarial attacks, which distinguishes our approach from previous studies. We employed two datasets widely applied in intrusion detection research and conducted a comprehensive performance evaluation. This evaluation encompassed the model's accuracy and robustness in both non-adversarial and adversarial environments, with the latter involving black-box and white-box attacks.

## 3 Methodology

In this section, we introduce the proposed ADHS-EL, with its overall workflow shown in Fig. 2.

ADHS-EL consists of two stages, namely performing traffic sampling and training intrusion detection models. The first stage includes dynamic hybrid sampling and boundary malicious traffic adversarial augmentation, both performed based on classification hardness to achieve targeted traffic sampling and obtain the hybrid traffic set. In the second stage, the hybrid traffic set output from the first stage is taken as input to the base classification algorithm and trained to get the current detection model, which is cumulatively averaged with the previously obtained ensemble detection model as the current ensemble. Note that these two stages proceed dynamically during the iteration. The prediction errors of the current ensemble serve as feedback information to provide the hardness distribution calculation, guiding the sampling process for the next round of model training.

### 3.1 Classification hardness calculation

Classification hardness (Liu et al. 2020) can be used to measure the difficulty of correctly discriminating a traffic sample using a detection model. The greater the classification hardness, the more difficult the sample is to be classified correctly.

The symbol $H$ represents the classification hardness function, which refers to any decomposable error function capable of determining the total error based on the aggregated errors of individual samples. Suppose $F$ is a trained classifier, and the classification hardness for a sample $(x, y)$ relative to $F$ is defined by $H(x, y, F)$. Our proposed approach utilizes the absolute error to measure classification hardness, specifically expressed as $H(x, y, F) = 1 - p_y$, where $p_y$ signifies the predicted likelihood of $x$ being classified as $y$ according to $F$, ranging from 0 to 1. The computation of the absolute error is straightforward, and its value spans from 0 to 1, easily corresponding to the sample type.

Given that the traffic dataset has $m$ classes and the classifier determines the class based on the maximum predictive probability, we analyze the correspondence between sample type and classification hardness. Samples with classification hardness in [0, 0.5) tend to be accurately classified and are

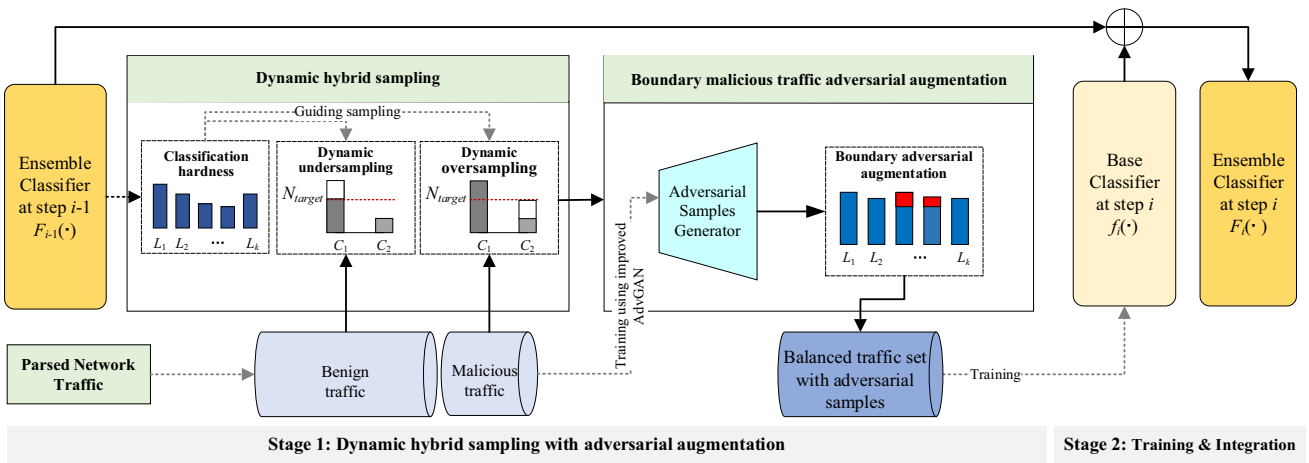



**Fig. 2** Proposed ADHS-EL framework

of the simple type. Conversely, when the hardness value falls in $[0.5, (m-1)/m]$, the classification outcomes may be correct or incorrect, with a higher likelihood of being boundary samples. Finally, samples whose classification hardness falls in $((m-1)/m, 1]$ are inevitably misclassified and are more likely to be noise samples.

Furthermore, the importance of the data in the classification task is demonstrated by means of the hardness contribution, i.e., the cumulative classification hardness of multiple samples. For a specified class, the samples are partitioned into $k$ distinct groups based on their respective classification hardness values. Here, $k$ is a hyperparameter that determines the number of groups. Specifically, let $L_j$ represent the $jth$ group comprising samples as defined in (1), where $j$ ranges from 1 to $k$ as an integer. $C_j$ denotes the hardness contribution of the samples in the $jth$ group, as shown in (2).

$$L_j = \left\{ (x, y) \middle| \frac{j-1}{k} \leq H(x, y, F) < \frac{j}{k} \right\} \text{ w.l.o.g. } H \in [0, 1] \tag{1}$$

$$C_j = \sum_{s \in L_j} H\left(x_s, y_s, F\right) \tag{2}$$

Simple samples are already well classified, so only smaller and smaller hardness contributions need to be retained during iterative training. Noisy samples arising from indistinguishable overlaps or anomalies are steadily present in the iterative process, so less attention should be paid to such samples to prevent overfitting. Boundary samples are crucial for decision-making, so increasing the importance of these samples is beneficial for the model, both in maintaining generalization ability and in improving learning ability for difficult samples. This principle will be followed to guide dynamic hybrid sampling for targeted sampling.

## 3.2 Dynamic hybrid sampling

To balance the traffic distribution, dynamic hybrid sampling is implemented, i.e., dynamic undersampling for benign traffic to reduce the number of samples and dynamic oversampling for malicious traffic to increase the number of samples. The same guiding principle is applied in both undersampling and oversampling, i.e., reducing the focus on simple and noisy samples while increasing the importance of boundary samples in the iteration.

### 3.2.1 Sampling weight calculation

We assign different sampling probabilities $w_j$ to $L_j$ to achieve targeted sampling for different types of samples.

First, the average hardness contribution $\bar{C}_j$ of $L_j$ is calculated as Eq. (3).

$$\bar{C}_j = \frac{C_j}{|L_j|} \tag{3}$$

Then, a dynamic factor $\delta$ is introduced to control the hardness contribution of different types of samples in the $ith$ iteration. The dynamic factor $\delta_j$ associated with $L_j$ is defined as in (4).

$$\delta_j = \begin{cases} \alpha_i, 0 \leq \bar{C}_j < 0.5 \text{ or } (\mathrm{m}-1)/\mathrm{m} < \bar{C}_j \leq 1 \\ \beta_i \times \alpha_i, \quad 0.5 \leq \bar{C}_j \leq (\mathrm{m}-1)/\mathrm{m} \end{cases} \tag{4}$$

$\delta$ is based on a self-paced factor $\alpha$ and a boundary sample weighting factor $\beta$. $\alpha$ controls the overall change in the hardness contribution of the samples, with samples of greater hardness contributing more and more to the sampled data as the iteration progresses. However, using $\alpha$ alone may result in overfitting noisy samples. Thus, $\beta$ is utilized to specifically weight the hardness contribution of boundary samples. In the $ith$ iteration, $\alpha$ and $\beta$ are defined in (5) and (6), where $n$ denotes the iterations.

$$\alpha_i = \tan \frac{(i-1)\pi}{2n} \tag{5}$$

$$\beta_i = \begin{cases} \frac{n-i}{n}, 0 \leq i \leq \lfloor \frac{n}{2} \rfloor \\ \frac{i}{n}, \lfloor \frac{n}{2} \rfloor < i \leq n \end{cases} \tag{6}$$

$\beta$ is updated in stages with iterations, and the hardness contribution of boundary samples is dynamically enhanced. In the early iterations, the enhanced magnitude slowly increases so as not to interfere with the model's learning on the simple samples. In the later iterations, the enhanced magnitude is gradually reduced to avoid overfitting complex boundaries.

Finally, the sampling weight $w_j$ for $L_j$ is computed based on $\bar{C}_j$ and $\delta_j$, as in (7), and the normalized sampling weight $w_j^{nor}$ can be obtained according to (8).

$$w_j = \frac{1}{\bar{C} + \delta_j}, \forall j = 1, 2, \ldots, k \tag{7}$$

$$w_j^{nor} = \frac{w_j}{\sum_{u=1}^{k} w_u} \tag{8}$$

### 3.2.2 Hybrid sampling process

After obtaining the normalized sampling weights for each group, the sampling number for each group can be calculated as shown in (9), where $N_{target}$ is the target sampling number.

$$N_j = N_{target} \times w_j^{nor} \tag{9}$$

For benign traffic, dynamic undersampling (DUS) is performed, where non-repetitive random sampling is conducted based on the sampling number of each group to reach the target sample size. For malicious traffic, dynamic oversampling (DOS) is executed, where repetitive random sampling is carried out to achieve the target sample size. Unlike the undersampling of benign traffic, we get more duplicates in the sampled set of malicious traffic.

## 3.3 Adversarial augmentation for boundary malicious traffic

ADHS-EL utilizes the adversarial training idea to add adversarial samples to the training dataset to train the model and enhance its adversarial robustness. Notably, most works show that standard adversarial training reduces the accuracy of intrusion detection models on the original samples. This problem may be related to the inclusion of extremely adversarial samples in the training process (Zhang et al. 2020b), where adversarial samples that cross the ground truth decision boundary are added to the training process, affecting the determination of the original samples. In light of this, we propose the boundary malicious traffic adversarial augmentation (BMTAA) strategy for the following two reasons.

(1) The attackable data is closer to the class boundary, while the guarded data is further away from the class boundary (Zhang et al. 2020c). For boundary samples, it is easier to generate adversarial samples that bypass detection after imposing a small perturbation, which is closer to the characteristics of adversarial attacks crafted by attackers.
(2) Boundary samples play a crucial role in model decision-making. Adding the adversarial version of these samples for adversarial training is equivalent to increasing the weight of the boundary samples, which is helpful to alleviate the reduced accuracy of standard adversarial training on the original test data.

### 3.3.1 Adversarial settings

In generating adversarial samples for malicious traffic, we assume a white-box attack scenario at the feature space level of the traffic, where we train the adversarial sample generator using the predictive probability feedback from the intrusion detection model. Adversarial samples generated at the feature space level can help enhance model robustness (Han et al. 2021). Due to the relaxed constraints on the real traffic space, generating samples at the feature space level encompasses a greater range of possible adversarial attacks, which is meaningful for defense.

### 3.3.2 Adversarial generator training

GAN is capable of generating adversarial samples against any intrusion detector supported by machine learning algorithms. In ADHS-EL, adversarial samples of malicious traffic are generated using AdvGAN, guiding the generator training by directly utilizing the feedback from the intrusion detection model. The original research applies AdvGAN to image-type samples, yet traffic is table-type data. We modify the loss function to work with traffic data, and it ensures that the generated samples are as close as possible to malicious features while mimicking the distribution of benign features to bypass the detection model. The structure of the improved AdvGAN model is shown in Fig. 3, which mainly consists of the base intrusion detection model $f_0$, the discriminator $D$, and the generator $G$.

***A. Base intrusion detection model***

The base intrusion detection model can be a classification model constructed by any machine learning algorithm, which is the target model to be bypassed by the generated samples in the framework shown in Fig. 3. The feedback of this model for the generated samples, i.e., the adversarial loss $L_{adv}$, is utilized to guide the training of the generator.

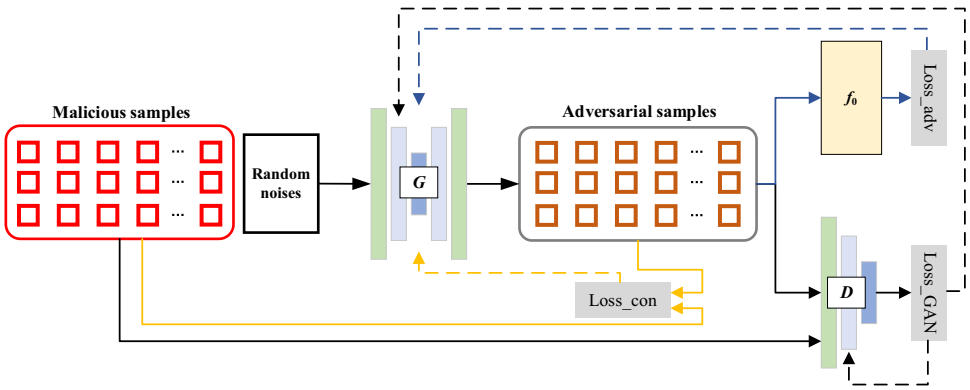



**Fig. 3** Improved AdvGAN for malicious traffic adversarial sample generation

***B. Generator***

The generator $G$ is a feed-forward neural network that aims to convert malicious traffic into its adversarial version. It combines a malicious traffic sample vector and a random noise vector from a uniform distribution $U(0, 1)$ as input and outputs a generated adversarial sample vector, denoted as $G(x, z)$. This sample should have characteristics that are close to the original malicious distribution and capable of deceiving the target intrusion detection model, so the feedback from the discriminator $D$ and the target intrusion detection model $f_0$ are utilized to guide the generator training.

The sample generated by $G$ is fed to $D$, which calculates the probability that the input sample is considered to be a real sample. The loss value $L_{GAN}$ fed back to $G$ from $D$ is calculated by (10), where $X_{mal}$ denotes the malicious sample set.

$$L_{GAN} = \mathbb{E}_{x \in X_{mal}, z \sim U(0,1)} \left[ \log \left( 1 - D \left( G \left( x, z \right) \right) \right) \right] \tag{10}$$

In parallel, this generated sample is fed into the target intrusion detection model $f_0$ to predict its probability of being a target class (i.e., benign class). The loss value $L_{adv}$ for spoofing $f_0$ is given by (11), where $l_{f_0}$ is the loss function for training a target model.

$$L_{adv} = \mathbb{E}_{x \in X_{mal}, z \sim U(0,1)} \left[ l_{f_0} \left( G \left( x, z \right), t \right) \right] \tag{11}$$

We improve the soft hinge loss used to measure the distance between images in the original AdvGAN. The Root Mean Square Error (RMSE) is used to measure the constructive loss between the original samples and the adversarial samples, and the formula is shown in (12), where $n_d$ is the feature dimension of the input and output samples.

$$L_{con} = \sqrt{\sum_{i=1}^{n_d} \left( x^i - G \left( x, z \right)^i \right) / n_d} \tag{12}$$

To this point, the complete loss function $L_G$ of the generator is shown in (13), and the training objective is to minimize this loss function.

$$L_G = L_{GAN} + L_{adv} + L_{con} \tag{13}$$

Among them, $L_{GAN}$ helps generate adversarial samples similar to the original sample distribution, $L_{adv}$ contributes to crafting adversarial samples with high successful evasion rates, and $L_{con}$ serves to constrain the perturbation range to ensure sample functionality.

***C. Discriminator***

The discriminator $D$ is also a feed-forward neural network that distinguishes between the original samples and the generated adversarial samples. $D$ is trained to maximize the output of the original malicious samples while minimizing the output of the generated samples. The loss function $L_D$ is defined as in Eq.(14), and its optimization objective is to minimize this loss function.

$$L_D = \mathbb{E}_{x \in X_{mal}, z \sim U(0,1)} \left[ \log D \left( G \left( x, z \right) \right) \right] - \mathbb{E}_{x \in X_{mal}} \left[ \log D \left( x \right) \right] \tag{14}$$

During training, the generator and the discriminator optimize each other with iterations until convergence.

#### 3.3.3 Adversarial samples generation

In the malicious sample set after dynamic oversampling, we select the unduplicated sample set $S_{mal\_bou}$ with classification hardness between $[0.5, (m-1)/m]$, and the rest of the samples may be considered to be the sample set after dynamic oversampling, $S_{mal\_DOS}$. We feed $S_{mal\_bou}$ into the trained adversarial samples generator to obtain the adversarial version of the malicious sample set, $S^{aug}_{mal\_bou}$. $S_{mal\_DOS}$ is mixed with $S^{aug}_{mal\_bou}$ to get the malicious traffic dataset with boundary adversarial augmentation.

### 3.4 Ensemble training process

We formally describe the training process of ADHS-EL, and the details are shown in Algorithm 1. Let $S$ represent the training dataset, $S_{mal}$ denote the malicious class training samples set, and $S_{c_q}$ denote the set of training samples with class $c_q, \forall q = 1, 2, \cdots, m$.

ADHS-EL takes Boosting as its base integration framework and utilizes dynamic hybrid sampling with adversarial augmentation to provide a training subset for the base classifier in each iteration. This sampling process corresponds to lines 6 to 16 in Algorithm 1, where line 7 calculates the sampling weights for a given class of samples. For benign samples, dynamic undersampling is performed (lines 9 and 10), and for malicious samples, dynamic oversampling and boundary adversarial augmentation are executed (lines 12-14). The sampled benign and malicious traffic is mixed to obtain a new training subset, which is used to train the classifier for this iteration (line 17). At the end of the iteration, the classifiers from the $n$ training are cumulatively averaged to get the final ensemble (line 18).

### 3.5 Time complexity analysis

Based on Algorithm 1, the time complexity calculation of the proposed method can be divided into two parts: before the iterative training (lines 1-2) and the iterative training process (lines 3-19).

**Algorithm 1** ADHS-EL

**Input:** Training set $S$, iterations $n$, number of groups $k$, target sampling number $N_{target}$, base classifier $f$
**Output:** Integrated intrusion detector $F(\cdot)$
1: Train the initial base classifier $f_0(\cdot)$ on $S$
2: Train the adversarial sample generator $G$ on $S_{mal}$ (3.3.2)
3: **for** i=1 to n **do**
4: Initialize sampled subset $S^{'} \leftarrow \emptyset$
5: Update current ensemble $F_i(x) = \frac{1}{i}\sum_{v=0}^{i-1} f_v(x)$
6: **for** q=1 to m **do**
7: Calculate the sampling weights $w$ in $S_{c_q}$, involving $k$ (3.2.1)
8: **if** $|S_{c_q}| > N_{target}$ **then**
9: $S^{'}_{c_q} \leftarrow$ Performs dynamic undersampling, sampling $N_{target}$ instances from $S_{c_q}$ based on $w$ (3.2.2)
10: $S^{'} \leftarrow S^{'} \cup S^{'}_{c_q}$
11: **else**
12: $S^{'}_{c_q} \leftarrow$ Performs dynamic oversampling, sampling $N_{target}$ instances from $S_{c_q}$ based on $w$ (3.2.2)
13: $S^{'}_{c_q,adv\_aug} \leftarrow$ Implement adversarial augmentation via $G$ for the boundary malicious traffic from $S^{'}_{c_q}$ (3.3.3)
14: $S^{'} \leftarrow S^{'} \cup S^{'}_{c_q,adv\_aug}$
15: **end if**
16: **end for**
17: Train $f_i(\cdot)$ using $S^{'}$
18: $F(x) = \frac{1}{i}\sum_{v=1}^{i} f_v(x)$
19: **end for**

In the first part, the complexity comes from the training of both the initial base classifier and the adversarial sample generator. Assuming the training time for the base classifier $f$ on the dataset $S$ is $T^{train}_{f,S}$, the number of training epochs for the improved AdvGAN is $N_e$, the forward/backward propagation times for the generator and discriminator on the malicious class sample set $S_{mal}$ are $T_{G,S_{mal}}$ and $T_{D,S_{mal}}$ respectively, and the prediction time for the target classifier $f$ is $T^{pred}_{f,S_{mal}}$. Ignoring constant factors, the time complexity of the first part is $O\left(T^{train}_{f,S} + N_e \cdot \left(T_{G,S_{mal}} + T_{D,S_{mal}} + T^{pred}_{f,S_{mal}}\right)\right)$.

In the second part, the main contributors to the complexity during the iterative training process are the sampling weight calculation (line 7), the adversarial augmentation of boundary malicious samples (line 13), and the training of the base classifier (line 17). The sampling weight calculation relies on the prediction probabilities estimated by the current ensemble, which can be obtained by combining the cached prediction probabilities from the previous iteration's ensemble and the predictions from the latest trained base classifier. Assuming the prediction time for the latest trained base classifier on traffic samples of class $c_q$ is $T^{pred}_{f,S_{c_q}}$. The adversarial augmentation involves the forward propagation time of the generator $G$ on the boundary malicious sample set of class $c_q$, denoted as $T_{G,S_{c_q,mal\_bou}}$. The training time for the base classifier on the new training subset $S'$ is $T^{train}_{f,S'}$. Therefore, the complexity of the second part is $O\left(n \cdot \left(m \cdot \left(T^{pred}_{f,S_{c_q}} + T_{G,S_{c_q,mal\_bou}}\right) + T^{train}_{f,S'}\right)\right)$, where $n$ is the number of training iterations and $m$ is the number of classes in the traffic dataset. Here, $m \cdot \left(T^{pred}_{f,S_{c_q}} + T_{G,S_{c_q,mal\_bou}}\right)$ can be converted to $T^{pred}_{f,S} + T_{G,S_{mal\_bou}}$.

The time complexity of the two parts added together results in the final complexity of $O\left(T^{train}_{f,S} + N_e \cdot \left(T_{G,S_{mal}} + T_{D,S_{mal}} + T^{pred}_{f,S_{mal}}\right) + n \cdot \left(T^{pred}_{f,S} + T_{G,S_{mal\_bou}} + T^{train}_{f,S'}\right)\right)$. Evidently, the time complexity of ADHS-EL is linearly related to the number of training iterations $n$ and is mainly associated with the training and prediction complexity of the adversarial sample generator $G$ and the base classifier $f$.

## 4 Experiments and analysis

In this section, we show the experimental setup and the experimental results of ADHS-EL in both non-adversarial and adversarial environments. The proposed method is compared with some imbalance handling and adversarial defense methods in terms of both accuracy and adversarial robustness. The effectiveness of each module and the generalization of ADHS-EL are explored.

### 4.1 Experimental setup

#### 4.1.1 Dataset description

The datasets NSL-KDD[1] (Tavallaee et al. 2009) and CIC-IDS2017[2] (Sharafaldin et al. 2018) were employed for experimental evaluation. The datasets involved in the experiment are described as shown in Tables 2 and 3, where IR (Imbalance Ratio) denotes the ratio of the most majority class sample number to the relative minority class sample number.

The NSL-KDD dataset is an enhanced version of the KDD'99 dataset and serves as a publicly available benchmark for evaluating intrusion detection models. It includes benign traffic and four types of malicious traffic, with a total of 41 features extracted from the traffic samples, 31 of which are network-based and 10 are host-based. The dataset consists of four sub-datasets, of which KDDTrain+.txt and KDDTest+.txt were used for this experiment. However, the proportions of sample distributions in these two sub-datasets differ significantly, possibly with distributional inconsisten-

[1] https://www.unb.ca/cic/datasets/nsl.html
[2] https://www.unb.ca/cic/datasets/ids-2017.html

**Table 2** Data distribution of the NSL-KDD dataset

| Class | KDDTrain | | KDDTest(Test Set 2) | | Attribute |
|---|---|---|---|---|---|
| | Samples | IR | Samples | IR | |
| Normal | 67343 | 1 | 9711 | 1 | |
| DoS | 45927 | 1.466 | 7458 | 1.302 | |
| Probe | 11656 | 5.778 | 2421 | 4.011 | 41 |
| U2R | 52 | 1295.058 | 200 | 48.555 | |
| R2L | 995 | 67.681 | 2754 | 3.526 | |
| Total | 125973 | | 22544 | | |

cies. The non-adversarial environment was involved in this experiment, which requires the test set and training set distributions to be essentially the same in order to objectively demonstrate the model performance. For this reason, a portion of the training set provided by KDDTrain+.txt was divided as the test dataset in the non-adversarial environment, referred to as Test Set 1. The test set provided by KDDTest+.txt was used as the test dataset for the models' out-of-distribution generalizability, referred to as Test Set 2.

The CIC-IDS2017 dataset, developed by the Canadian Institute for Cybersecurity (CIC), provides an extensive and diverse collection of real-world network traffic data. It was created in a 5-day simulation environment, consisting of 2.83 million valid network traffic entries. For each flow, 78 features were extracted using the CICFlowMeter tool. The dataset includes benign traffic and 14 common types of attack traffic. We merged the traffic samples from Monday to Friday to form a single dataset, which will be utilized for subsequent model training and testing.

**Table 3** Data distribution of the CIC-IDS2017 dataset

| Class | Samples | IR | Attributes |
|---|---|---|---|
| BENIGN | 2273097 | 1 | 78 |
| FTP-Patator | 7938 | 286.356 | |
| SSH-Patator | 5897 | 385.467 | |
| DoS Hulk | 231073 | 9.837 | |
| DoS GoldenEye | 10293 | 220.839 | |
| DoS slowloris | 5796 | 392.184 | |
| DoS Slowhttptest | 5499 | 413.366 | |
| Heartbleed | 11 | 206645.182 | |
| Web Attack_Brute Force | 1507 | 1508.359 | |
| Web Attack_XSS | 652 | 3486.345 | |
| Web Attack_Sql Injection | 21 | 108242.714 | |
| Infiltration | 36 | 63141.583 | |
| DDoS | 128027 | 17.755 | |
| PortScan | 158930 | 14.303 | |
| Bot | 1966 | 1156.204 | |
| Total | 2830743 | | |

#### 4.1.2 Data preprocessing

The traffic dataset preprocessing consists of removing the task irrelevant or network context information leaking attributes, one-hot encoding of categorical attributes, and standardizing numerical attributes. Irrelevant attributes are not very useful for the detection task and attributes revealing network context tend to make machine learning algorithms learn by shortcuts, so these features are removed. The categorical attributes have discrete values, e.g., the attribute *protocol type* with values such as 'TCP', 'UDP', 'ICMP'. After one-hot encoding, this attribute is encoded as [1, 0, 0], [0, 1, 0], [0, 0, 1]. Data normalization is performed using Z-Score as shown in Eq.(15), where $x$ is the original data, $\sigma$ is the standard deviation of the sample, $\mu$ is the mean of the sample, and $x^*$ is the standardized result.

$$x^* = \frac{x - \mu}{\sigma} \tag{15}$$

For NSL-KDD, first, remove one attribute that has all 0 values, *num_outbound_cmds*. Then, nine attributes *protocol_type*, *service*, *flag*, *land*, *logged_in*, *root_shell*, *su_attempted*, *is_host_login*, and *is_guest_login* were enc oded by one-hot, getting 128-dimensional features for each record. Finally, data standardization was performed.

For CIC-IDS2017, nine features with all zero values were removed, including *Bwd PSH Flags*, *Bwd URG Flags*, *CWE Flag Count*, *Fwd Avg Bytes/Bulk*, *Fwd Avg Packets/Bulk*, *Fwd Avg Bulk Rate*, *Bwd Avg Bytes/Bulk*, *Bwd Avg Packets/Bulk*, and *Bwd Avg Bulk Rate*. In addition, the attribute *Destination Port*, which contains the context of the network environment, was deleted, as well as all records with 'inf' value. Finally, standardize the data. After preprocessing, the dataset with 2,520,798 samples and 68 features was obtained.

#### 4.1.3 Evaluation settings

To evaluate the in-distribution generalizability of the models, we conducted 10 independent experimental runs. In each run, we fixed the random seed and split the dataset into training, validation, and test sets in an 8:1:1 ratio. Stratified sampling

**Table 4** Network structure of the improved AdvGAN

| Module | Layers | Out shape | Activation function |
|---|---|---|---|
| Generator | Linear | (input_dim, input_dim/2) | ReLU |
| | Linear | (input_dim/2, input_dim/2) | ReLU |
| | Linear | (input_dim/2, input_dim/2) | ReLU |
| | Linear | (input_dim/2, input_dim/2) | ReLU |
| | Linear | (input_dim/2, input_dim) | – |
| Discriminator | Linear | (input_dim, input_dim/2) | LeakyReLU |
| | Linear | (input_dim/2, input_dim/2) | LeakyReLU |
| | Linear | (input_dim/2, input_dim/2) | LeakyReLU |
| | Linear | (input_dim/2, input_dim/2) | LeakyReLU |
| | Linear | (input_dim/2, 1) | – |
| Base model | DT/RF/KNN/MLP | | |

was used to ensure that the class distribution in each subset remained consistent with the overall dataset. For the NSL-KDD dataset, we conducted the data split using the dataset provided in KDDTrain+.txt. The final experimental results were obtained by averaging the performance metrics from the test set across 10 runs. To evaluate the out-of-distribution generalization, the models trained in 10 independent runs were evaluated on NSL-KDD's Test Set 2. All experiments were conducted on a Microsoft Windows 10 64-bit operating system, utilizing an Intel (R) Core (TM) i7-8750 H CPU and 32 GB of memory, with Python 3.11.4 serving as the programming environment.

We tested the applicability of the proposed method by selecting different types of machine learning algorithms to build the base intrusion detection model, including DT, RF, KNN, and MLP. These classifiers were implemented via the Python third-party library scikit-learn 1.1.3, with default parameters.

The experiments involved model accuracy and adversarial robustness measurement. Overall Accuracy (ACC), Macro Average F1-score (Macro.F1), and Macro Area Under Curve (Macro.AUC) metrics were used to evaluate the accuracy of the model. The performance in recognizing malicious traffic was evaluated using Precision, Recall, and F1-score. In adversarial robustness evaluation, adversarial samples generated against malicious traffic were applied to test the model performance, and the detection rate of adversarial samples was used as the evaluation metric.

In adversarial robustness testing, three common black-box adversarial machine learning attacks were used to evaluate the robustness of the proposed model, including ZOO, HSJA, and BA, which can act on intrusion detection models constructed by any machine learning algorithm. We also used DTA to test model robustness, given that tree-based algorithms were utilized in constructing the intrusion detection models in our experimental testing. DTA can only target DT, so the same adversarial samples made against DT were leveraged when testing the robustness of other models against this attack. The above attacks were executed with the ART[3] toolkit and set to default parameters.

#### 4.1.4 Parameter settings

ADHS-EL primarily comprises dynamic hybrid sampling and ensemble iteration. The involved parameters include the number of iterations $n$, the number of groups $k$, and the target sampling number $N_{target}$, which were set to 5, 5, and the average number of samples used in training, respectively. During the dynamic hybrid sampling process, adversarial samples of boundary malicious traffic, generated by the generator of an improved AdvGAN, are incorporated into the training set. The network architecture of the improved AdvGAN is detailed in Table 4, where input_dim denotes the preprocessed feature dimension.

In our experiments, the network traffic data is structured in a tabular format. Fully connected networks can capture all possible correlations between columns (Xu et al. 2019), so we utilized fully connected networks to construct the generator and discriminator in the improved AdvGAN. The generator is a neural network containing five fully connected layers, with the first four layers activated by the ReLU function. The discriminator also has five linear layers but uses LeakyReLU as its activation function. Following the settings for the number of hidden layer neurons in Usama et al. (2019) and Duy et al. (2023), which suggest approximately half the input dimension, we set the number of neurons in each hidden layer to input_dim/2, rounded to the nearest integer.

In the training of the improved AdvGAN, the weights of both the generator and discriminator were initialized using Kaiming normal initialization, with Adam serving as the optimizer. To promote more stable convergence, we

[3] https://github.com/Trusted-AI/adversarial-robustness-toolbox/

employed StepLR as the learning rate scheduler with a step size of 20 epochs and a decay factor of 0.1. The number of training epochs was set to 100, and Optuna was utilized to search for suitable hyperparameters, with the objective of minimizing the detection rate of adversarial samples by the target base classifier on the validation set. Specifically, the initial learning rate was selected from the set [0.00001, 0.0001, 0.001, 0.01], the batch size ranged from 50 to 300 with a step size of 50, and the n_critic, representing the number of discriminator training iterations per AdvGAN training epoch (while the generator is trained once per epoch), was selected from the set [1, 5, 10]. The generator trained under the optimal parameters was subsequently used in ADHS-EL to provide adversarial samples for malicious traffic.

## 4.2 Adversarial samples generation analysis

As the number of epochs increases, we observe the changes in the losses during the training process of the improved AdvGAN. Then, we demonstrate the adversarial effect of trained generators generating samples.

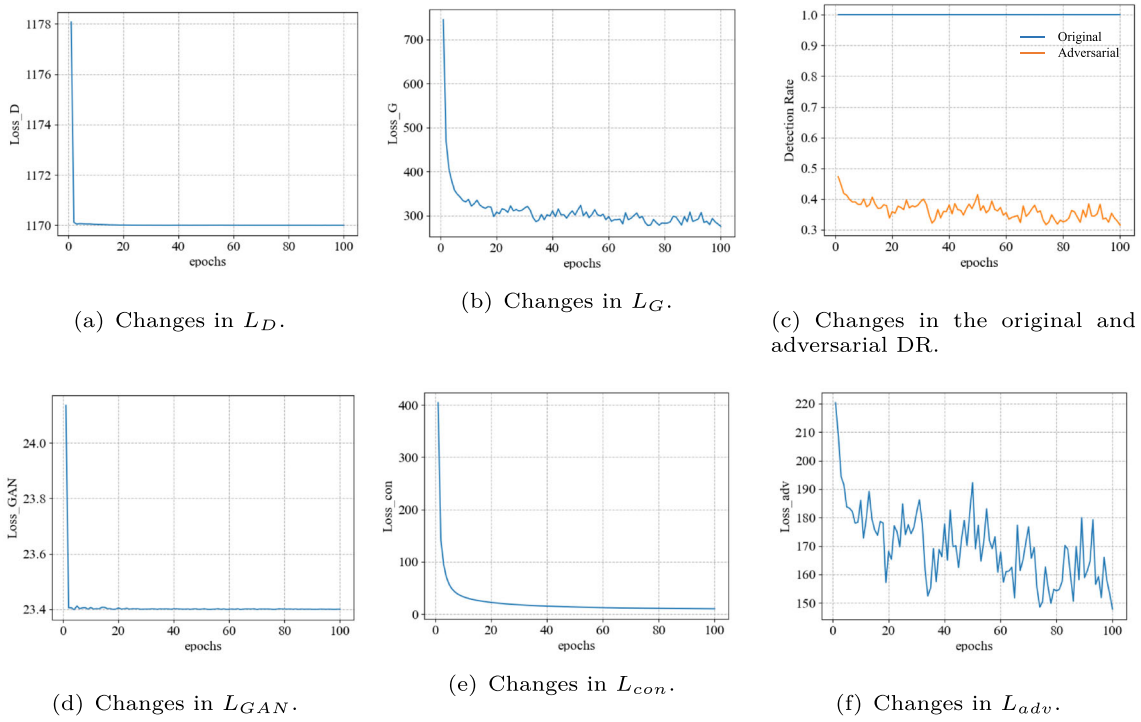
### 4.2.1 Changes in the losses during training

Figure 4 demonstrates the change in losses during AdvGAN training with DT as the target binary classification intrusion detection model on the NSL-KDD dataset. Figure 4(a) and (d) show the losses of the discriminator and generator, $L_D$ and $L_{GAN}$, respectively, which both stabilize after 20 epochs, indicating that the generative and discriminative abilities have reached a balanced state. Figure 4(e) presents the constructive loss of the generator, $L_{con}$, which varies smoothly after 20 epochs. Figure 4(f) shows the adversarial loss $L_{adv}$, fluctuating relatively steadily between 150 and 190 after 20 epochs. Figure 4(b) is the overall change of the three losses, $L_{GAN}$, $L_{con}$, and $L_{adv}$, during the training of the generator, which only fluctuates slightly after 20 epochs, and its fluctuation is mainly due to the adversarial loss. In Fig. 4(c), after 20 epochs, the original malicious sample detection rate tends to 1, while the adversarial samples detection rate ranges from 0.3 to 0.4, indicating that the generator is capable of generating samples with adversarial characteristics.

### 4.2.2 Adversarial performance of generated samples

The trained generator was utilized to generate adversarial attacks against malicious traffic samples from the validation sets of NSL-KDD and CIC-IDS2017. Figure 5 demonstrates the detection rate before and after the attacks on the boundary malicious samples and 1000 randomly sampled malicious samples.

For both datasets, the original detection rate of the boundary malicious samples is around 0.5, while that of the randomly sampled samples is higher. After the adversarial attacks, the detection rate of the boundary malicious samples




(a) Changes in $L_D$.

(b) Changes in $L_G$.

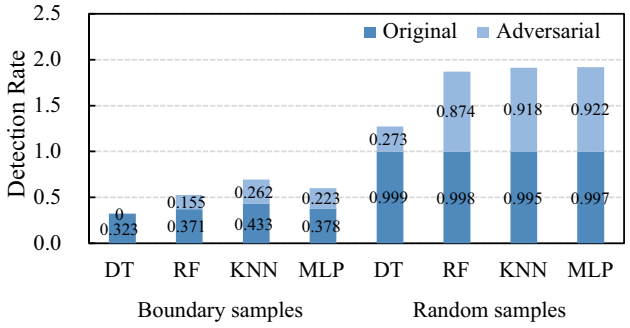
(c) Changes in the original and adversarial DR.

(d) Changes in $L_{GAN}$.

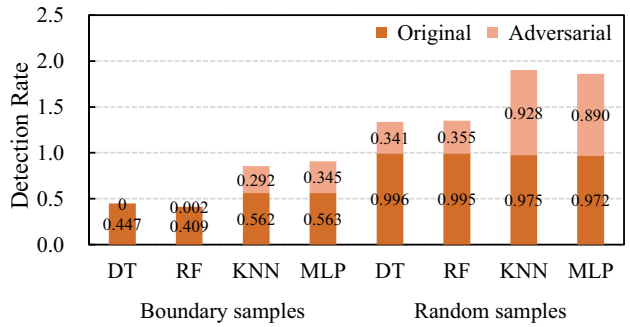
(e) Changes in $L_{con}$.

(f) Changes in $L_{adv}$.

**Fig. 4** The change in losses during improved AdvGAN training




(a) Results on NSL-KDD.




(b) Results on CIC-IDS2017.

**Fig. 5** The effects of generated adversarial samples on both datasets

decreases significantly, indicating that the trained generator is more effective against the boundary malicious samples and serves the purpose of generating adversarial samples.

## 4.3 Performance comparison between the original model and the enhanced model

We compare the accuracy and robustness of the original model and the enhanced model obtained from the proposed ADHS-EL, where the original model was trained on the training set that was only preprocessed. The accuracy evaluation experiments were conducted in a non-adversarial setting, i.e., on a test set where no adversarial attacks are applied and the distribution is consistent with that of the training set. When testing the adversarial robustness, 1,000 malicious samples were selected from the test set according to the percentage of the original sample classes.

In particular, for the NSL-KDD training dataset, the ratio of normal samples to malicious samples is 1.14. When training the binary intrusion detection model, 20,000 malicious samples in the training set were sampled to participate in the training for matching the imbalanced data distribution in actual network traffic.

### 4.3.1 Accuracy comparison

For the NSL-KDD and CIC-IDS2017 datasets, the performance metrics of the original and enhanced models on binary and multiclass classification tasks are shown in Tables 5 and 6.

The results show that the enhanced model has achieved some improvements over the original model in terms of ACC, Macro.F1, and Macro.AUC metrics. On the binary classification task, ADHS-EL improves the Macro.F1 of the original DT, RF, KNN, and MLP by 0.050%, 0.040%, 0.040%, and 0.080%, respectively, for the NSL-KDD dataset, while for the CIC-IDS2017 dataset, the improvements are 0.060%, 0.020%, 0.294%, and 2.13%, respectively. On the multiclass classification task, the performance improvement is relatively more significant. For NSL-KDD, ADHS-EL improves the Macro.F1 of the original DT, RF, KNN, and MLP by 1.62%, 2.51%, 0.806%, and 1.80%, respectively. While for CIC-IDS2017, the improvements in Macro.F1 are 2.40%, 3.45%, 3.63%, and 2.71%, respectively. This indicates that ADHS-EL enhances the average classification ability of the detectors for both benign and malicious traffic samples. In short, ADHS-EL contributes to improving the detection accuracy of the original models for both binary and multiclass classification tasks.

**Table 5** Model performance comparison for NSL-KDD

| Classification | Base classifier | Original model | | | Enhanced model | | |
|---|---|---|---|---|---|---|---|
| | | ACC | Macro.F1 | Macro.AUC | ACC | Macro.F1 | Macro.AUC |
| Binary | DT | 0.9979 | 0.9979 | 0.9978 | 0.9984 | 0.9984 | 0.9984 |
| | RF | 0.9978 | 0.9978 | 0.9977 | 0.9982 | 0.9982 | 0.9981 |
| | KNN | 0.9955 | 0.9954 | 0.9952 | 0.9958 | 0.9958 | 0.9957 |
| | MLP | 0.9960 | 0.9960 | 0.9959 | 0.9968 | 0.9968 | 0.9967 |
| Multiclass | DT | 0.9980 | 0.9240 | 0.9616 | 0.9985 | 0.9255 | 0.9980 |
| | RF | 0.9988 | 0.9248 | 0.9934 | 0.9989 | 0.9480 | 0.9997 |
| | KNN | 0.9945 | 0.8689 | 0.9752 | 0.9947 | 0.8759 | 0.9519 |
| | MLP | 0.9966 | 0.8734 | 0.9972 | 0.9964 | 0.8891 | 0.9976 |

**Table 6** Model performance comparison for CIC-IDS2017

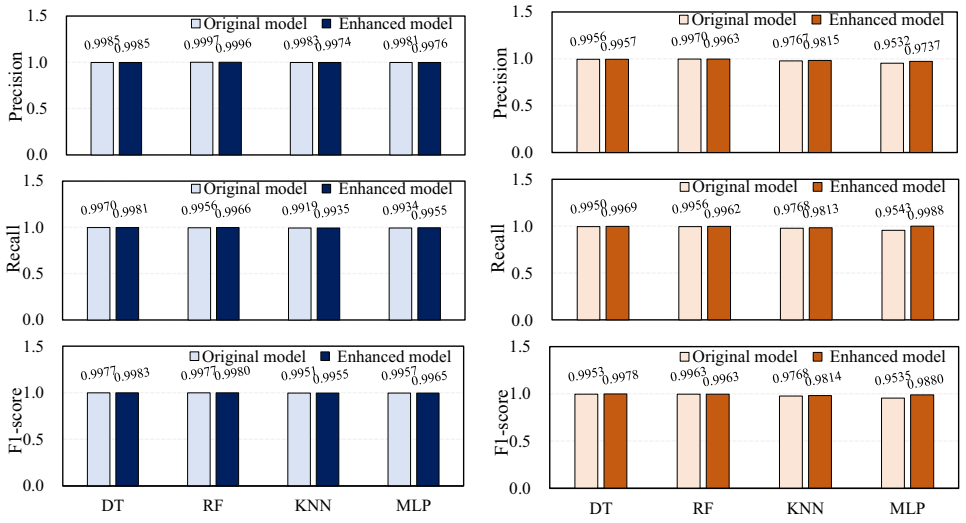
| Classification | Base classifier | Original model | | | Enhanced model | | |
|---|---|---|---|---|---|---|---|
| | | ACC | Macro.F1 | Macro.AUC | ACC | Macro.F1 | Macro.AUC |
| Binary | DT | 0.9984 | 0.9972 | 0.9970 | 0.9988 | 0.9978 | 0.9980 |
| | RF | 0.9983 | 0.9976 | 0.9975 | 0.9987 | 0.9978 | 0.9977 |
| | KNN | 0.9921 | 0.9859 | 0.9858 | 0.9937 | 0.9888 | 0.9885 |
| | MLP | 0.9843 | 0.9720 | 0.9723 | 0.9959 | 0.9927 | 0.9971 |
| Multiclass | DT | 0.9980 | 0.8709 | 0.9385 | 0.9984 | 0.8918 | 0.9913 |
| | RF | 0.9984 | 0.8473 | 0.9989 | 0.9974 | 0.8766 | 0.9994 |
| | KNN | 0.9917 | 0.7552 | 0.9362 | 0.9919 | 0.7826 | 0.9564 |
| | MLP | 0.9710 | 0.6613 | 0.9994 | 0.9848 | 0.6792 | 0.9990 |

The detection performance of malicious traffic in intrusion detection tasks is crucial. Further, we present the Precision, Recall, and F1-score for malicious traffic on the binary classification task, and the results for both datasets are shown in Fig. 6. For NSL-KDD, the Recall values for malicious traffic are improved by 0.110%, 0.101%, 0.161%, and 2.11% for the original DT, RF, KNN, and MLP models, respectively, with ADHS-EL. For CIC-IDS2017, the Recall values of the original DT, RF, KNN, and MLP are enhanced by 0.196%, 0.061%, 0.461%, and 4.67%, while the Precision values of the original RF were slightly decreased due to the fact that increasing Recall may come at the expense of Precision. Overall, the F1-score values of the enhanced models are slightly higher than those of the original models for both datasets, demonstrating that the proposed method facilitates the identification of malicious traffic.

#### 4.3.2 Adversarial robustness comparison

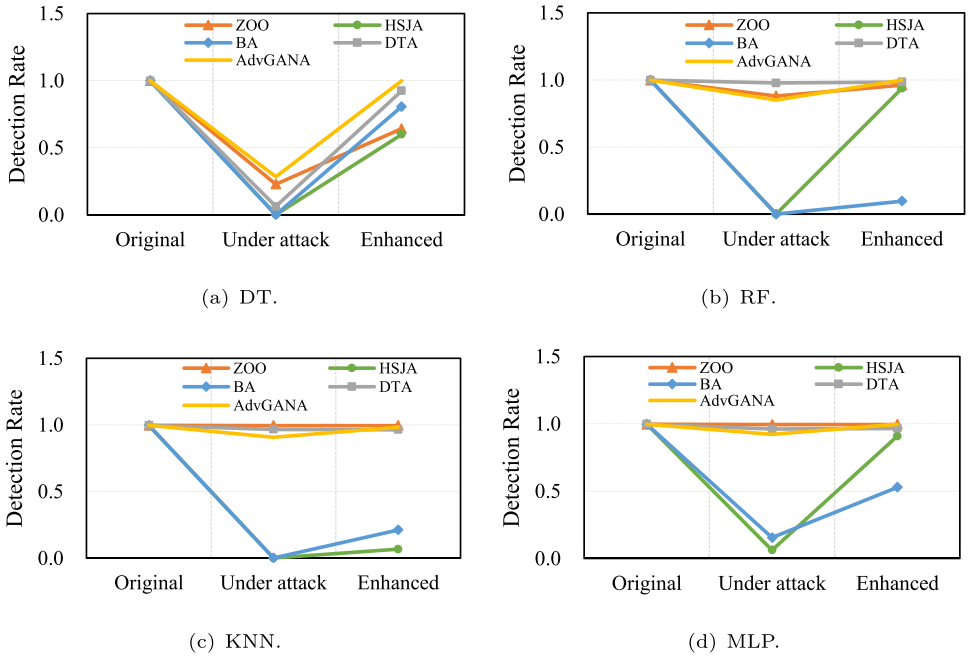
The robustness of the original and enhanced models was evaluated with ZOO, HSJA, BA, and DTA. Additionally, the robustness of the models against adversarial samples generated by the improved AdvGAN presented in this paper was tested. The detection results of the binary classification models on the NSL-KDD and CIC-IDS2017 datasets for adversarial samples are shown in Figs. 7 and 8, where AdvGANA denotes the adversarial samples generated by the improved AdvGAN.

The original model has a low detection rate for adversarial samples. For NSL-KDD and CIC-IDS2017, the detection rates of DT, RF, KNN, and MLP on adversarial samples generated by HSJA and BA are nearly all 0, illustrating that the original models' detection capabilities for malicious




(a) NSL-KDD. (b) CIC-IDS2017.

**Fig. 6** Malicious traffic identification results on the binary classification task for both datasets



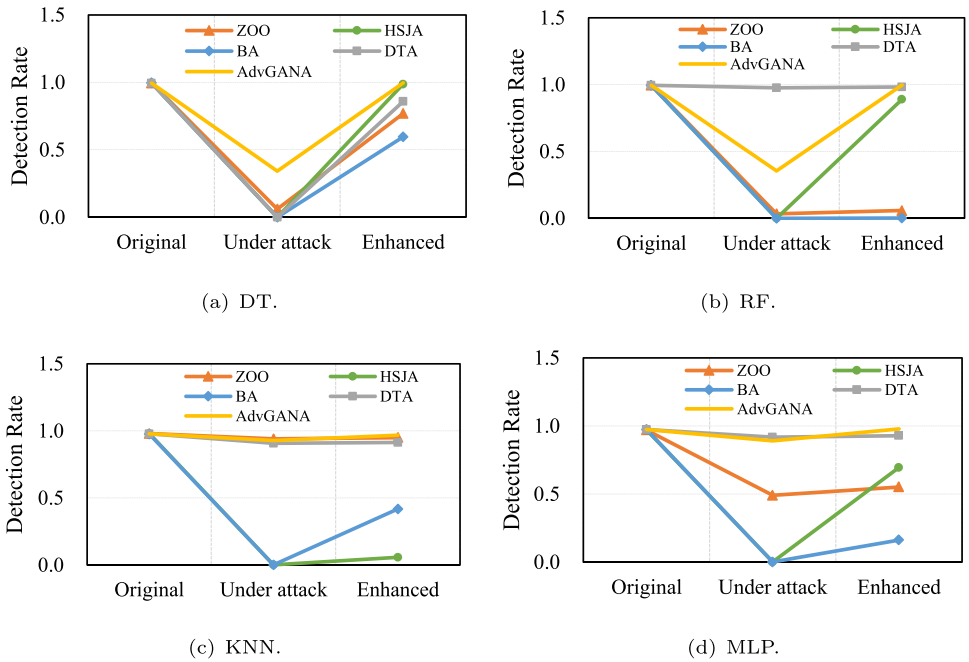

(a) DT. (b) RF.

(c) KNN. (d) MLP.

**Fig. 7** Detection performance of models under adversarial attacks on the NSL-KDD dataset

samples is significantly compromised after being attacked. From Fig. 7(a), ZOO degrades the detection performance of the original DT on the NSL-KDD dataset. From Fig. 8, ZOO significantly reduces the detection rates of DT and RF. DTA only has a strong attack effect on the original DT, with minimal impact on the performance of other models.

On both datasets, the detection rate of the enhanced model for adversarial samples is improved. From Figs. 7(a) and 8(a), the detection rate of the enhanced DT for adversarial samples generated by five types of adversarial attacks reaches over 60%. On the NSL-KDD and CIC-IDS2017 datasets, the enhanced RF restores the detection rate of




(a) DT. (b) RF.

(c) KNN. (d) MLP.

**Fig. 8** Detection performance of models under adversarial attacks on the CIC-IDS2017 dataset

adversarial samples generated by HSJA to 93.7% and 89.1%, respectively, as shown in Figs. 7(b) and 8(b). For KNN, with the application of the proposed method, the detection rate of adversarial samples generated by BA on CIC-IDS2017 reached 41.6%, as illustrated in Fig. 8(c). Figures 7(d) and 8(d) show that the enhanced MLP achieves a detection rate of 70% for adversarial samples generated by HSJA on both datasets. The enhanced models show good recovery effects on the decreased detection rates caused by DTA and Adv-GAN attacks. These improvements in the detection rate of adversarial samples demonstrate that ADHS-EL can enhance the adversarial robustness of the original intrusion detection models to some extent.

In addition, for the multiclass classification task, ADHS-EL can also improve the adversarial robustness of the model to a certain extent. We present the confusion matrices of the original and enhanced DT models under HSJA on the NSL-KDD dataset, as shown in Fig. 9.

From Fig. 9(a), the original DT has a high detection rate for malicious traffic. Under this adversarial attack, the original model misidentifies the adversarial version of malicious traffic as benign traffic, as seen in Fig. 9(b). As depicted in Fig. 9(c), the enhanced model is able to detect most adversarial samples, thus improving the adversarial robustness of the original model.

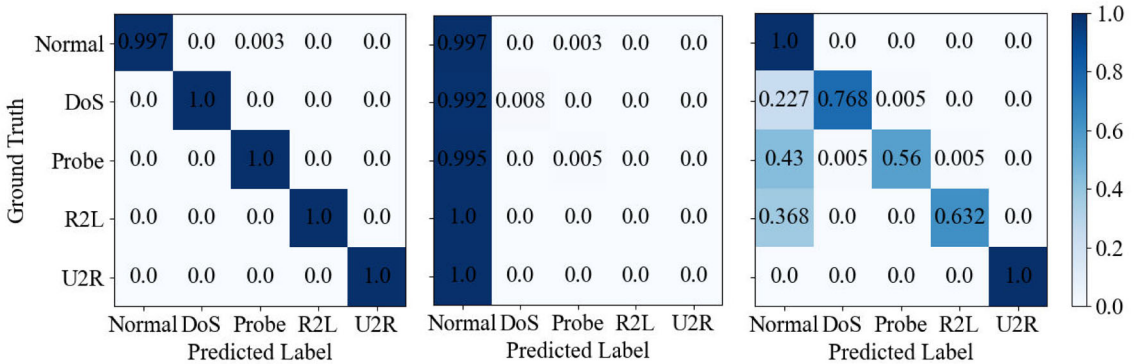
## 4.4 Comparison with related advanced methods

We compare ADHS-EL with class imbalance handling and adversarial defense methods in network intrusion detection and analyze the model performance from both accuracy and robustness on the binary classification task.

### 4.4.1 Comparison with class imbalance handling methods

We selected undersampling, oversampling, hybrid sampling, and ensemble methods as comparison baselines, including RUS, ROS, SMOTE, ADASYN, SMOTETomek, CTGAN, WGAN and DUEN, as mentioned in related work Sect. 2.2. The first five methods were implemented using the Python third-party library imblearn. CTGAN and WGAN were implemented based on the network architecture provided in Zhao et al. (2024). The implementation of DUEN was reproduced according to the algorithm description provided in the original paper. All tested methods employed DT as the base classifier, and the comparison results are shown in Table 7, where the robustness metric refers to the detection rate of the model under adversarial attacks, and AADR represents the average detection rate for adversarial samples. In the table, the best results are highlighted in bold.

In terms of accuracy, ADHS-EL achieves six of the highest scores on two datasets across five accuracy metrics. Regarding robustness, under four adversarial attacks, ADHS-EL obtains the highest average adversarial sample detection rates of 0.7435 and 0.8030 on the two datasets, which are 10.63% and 13.26% higher than the sub-optimal results. Notably, the class imbalance handling approaches improve the detection rate of adversarial samples, indicating that these techniques serve as a defense against the adversarial samples to some extent. This is because imbalance processing corrects the decision boundary and reduces the sensitivity of the model to adversarial samples. Despite achieving relatively high scores on the AADR metric, ADASYN, CTGAN, and WGAN exhibit poor performance in terms of accuracy. This is primarily due to the addition of a large number of malicious samples to the training set, which leads to overfitting during the training phase and thereby reduces the models' generalization ability with respect to original samples during the testing phase. In contrast, ADHS-EL employs dynamic hybrid sampling strategies guided by boundary augmentation. By preserving important benign samples and incorporating critical malicious samples, ADHS-EL not only maintains and improves model accuracy but also enhances its detection capability against adversarial samples.




(a) The original model. (b) The attacked model. (c) The enhanced model.

**Fig. 9** Confusion matrix for DT model under HopSkipJump Attack on NSL-KDD

**Table 7** Comparison with class imbalance handling methods on two datasets

| Dataset | Method | Accuracy metrics | | | | | Robustness metrics | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | Macro.F1 | Macro.AUC | F1 | Recall | Zoo | HSJA | BA | DTA | AADR |
| NSL-KDD | RUS | 0.9974 | 0.9974 | 0.9974 | 0.9972 | 0.9976 | 0.6203 | 0.3823 | 0.6688 | 0.4186 | 0.5225 |
| | ROS | 0.9978 | 0.9978 | 0.9977 | 0.9976 | 0.9966 | 0.5542 | 0.2957 | 0.3451 | 0.2245 | 0.3549 |
| | SMOTE | 0.9980 | 0.9980 | 0.9980 | 0.9979 | 0.9972 | 0.5890 | 0.5182 | 0.3452 | 0.5690 | 0.5054 |
| | ADASYN | 0.9967 | 0.9967 | 0.9966 | 0.9965 | 0.9955 | 0.7245 | **0.9412** | 0.2558 | 0.7414 | 0.6657 |
| | SMOTETomek | 0.9978 | 0.9978 | 0.9977 | 0.9976 | 0.9969 | 0.5973 | 0.5064 | 0.3799 | 0.6170 | 0.5252 |
| | CTGAN | 0.9976 | 0.9976 | 0.9975 | 0.9974 | 0.9966 | 0.6510 | 0.5832 | 0.8151 | 0.6199 | 0.6673 |
| | WGAN | 0.9978 | 0.9978 | 0.9978 | 0.9977 | 0.9968 | **0.7578** | 0.5741 | 0.7188 | 0.6375 | 0.6721 |
| | DUEN | **0.9985** | **0.9985** | **0.9984** | **0.9984** | 0.9978 | 0.5142 | 0.4039 | 0.4399 | 0.2706 | 0.4072 |
| | ADHS-EL(DT) | 0.9984 | 0.9984 | **0.9984** | 0.9983 | **0.9981** | 0.6425 | 0.5996 | **0.8070** | **0.9248** | **0.7435** |
| CIC-IDS2017 | RUS | 0.9985 | 0.9973 | 0.9985 | 0.9957 | **0.9984** | 0.4521 | 0.7146 | 0.2283 | 0.5636 | 0.4897 |
| | ROS | 0.9986 | 0.9974 | 0.9975 | 0.9958 | 0.9958 | 0.4475 | 0.6274 | 0.0554 | 0.3093 | 0.3599 |
| | SMOTE | 0.9985 | 0.9974 | 0.9975 | 0.9957 | 0.9960 | 0.5598 | 0.8292 | 0.0791 | 0.5622 | 0.5076 |
| | ADASYN | 0.9985 | 0.9972 | 0.9976 | 0.9962 | 0.9955 | 0.5184 | 0.8798 | 0.1392 | 0.5316 | 0.5173 |
| | SMOTETomek | 0.9986 | 0.9975 | 0.9978 | 0.9959 | 0.9965 | 0.4056 | 0.7336 | 0.0735 | 0.4590 | 0.4179 |
| | CTGAN | 0.9984 | 0.9972 | 0.9970 | 0.9953 | 0.9950 | 0.7493 | 0.9080 | 0.4091 | 0.7696 | 0.7090 |
| | WGAN | 0.9984 | 0.9971 | 0.9970 | 0.9952 | 0.9949 | 0.7570 | 0.8087 | 0.5420 | 0.6819 | 0.6974 |
| | DUEN | 0.9986 | 0.9976 | 0.9977 | 0.9959 | 0.9961 | 0.5228 | 0.7456 | 0.0356 | 0.2448 | 0.3872 |
| | ADHS-EL(DT) | **0.9988** | **0.9978** | **0.9980** | **0.9963** | 0.9969 | **0.7689** | **0.9875** | **0.5965** | **0.8592** | **0.8030** |

#### 4.4.2 Comparison with adversarial defense methods

Adversarial training and model integration were chosen as comparative baselines, both of which, similar to ADHS-EL, enhance model robustness to resist adversarial attacks. Adversarial training primarily involves generating adversarial attack examples using FGSM, PGD, and WGAN, which are then mixed with the original data to retrain the model. FGSM and PGD are capable of performing adversarial attacks only on neural network-structured models, and in our experimental tests, MLP was used as the target model to generate adversarial samples. WGAN can perform adversarial attacks on any machine learning model. In this experiment, WGAN generates adversarial samples targeting DT and MLP. Note that while WGAN has been used for addressing class imbalance problems, the training procedure in the context of adversarial defense differs from that setup (refer to Usama et al. 2019 for details). We refer to this adversarial training method as WGAN-AT. Additionally, recent research on improving model robustness includes the ensemble model Apollon (Paya et al. 2024). We compared ADHS-EL, using DT and MLP as base classifiers, with adversarial training methods FGSM, PGD, and WGAN-AT, as well as the ensemble method Apollon. The results on the NSL-KDD and CIC-IDS2017 datasets are presented in Table 8.

For accuracy evaluation, ADHS-EL with DT and MLP as base classifiers achieves the optimal values in accuracy metrics, outperforming the FGSM, PGD, and WGAN-AT methods trained with the same base classifiers. These three adversarial training methods using MLP as the base classifier exhibit poor performance in terms of accuracy, even falling below that of the original models (see Tables 5 and 6). In terms of robustness, among the models using DT as the base classifier, ADHS-EL achieves the highest average adversarial detection rate across both datasets. In models using MLP as the base classifier, WGAN-AT and PGD attain the highest AADR scores on the NSL-KDD and CIC-IDS2017 datasets, respectively. FGSM and PGD with DT as the base classifier perform average in robustness metrics, probably because the adversarial samples generated with MLP as the target model are less effective for DT. WGAN-AT (using MLP) achieves a higher AADR on the NSL-KDD dataset but performs averagely in other cases, such as with different datasets or when using DT as the base classifier. WGAN-AT focuses on causing the generated samples to be misclassified as normal during the training process, while neglecting constraints on perturbations of the original samples. As a result, the generated samples fail to effectively enhance the models' robustness. Apollon, which integrates multiple classification models, may have its accuracy pulled down by meta-classifiers with lower performance. Additionally, Apollon is inferior to the adversarial training methods in robustness metrics, indicating that adversarial training is an effective approach to improving model robustness. ADHS-EL constrains the perturbations between generated samples and original malicious samples when

**Table 8** Comparison with adversarial defense methods on two datasets

| Dataset | Method | Accuracy metrics | | | | | Robustness metrics | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ACC | Macro.F1 | Macro.AUC | F1 | Recall | Zoo | HSJA | BA | DTA | AADR |
| NSL-KDD | FGSM (DT) | 0.9980 | 0.9980 | 0.9979 | 0.9978 | 0.9969 | 0.4868 | 0.4445 | 0.4534 | 0.5002 | 0.4712 |
| | FGSM (MLP) | 0.9961 | 0.9961 | 0.9960 | 0.9958 | 0.9935 | 0.9935 | 0.6681 | 0.2271 | 0.9451 | 0.7085 |
| | PGD (DT) | 0.9980 | 0.9980 | 0.9979 | 0.9979 | 0.9969 | 0.4739 | 0.4051 | 0.5118 | 0.4232 | 0.4535 |
| | PGD (MLP) | 0.9963 | 0.9962 | 0.9961 | 0.9960 | 0.9937 | 0.9943 | 0.7294 | 0.2629 | 0.9501 | 0.7342 |
| | WGAN-AT(DT) | 0.9975 | 0.9975 | 0.9974 | 0.9973 | 0.9962 | 0.6211 | 0.4486 | 0.6428 | 0.4399 | 0.5381 |
| | WGAN-AT(MLP) | 0.9950 | 0.9950 | 0.9949 | 0.9947 | 0.9923 | 0.9911 | 0.9403 | 0.9225 | 0.9542 | 0.9520 |
| | Apollon | 0.9893 | 0.9892 | 0.9886 | 0.9884 | 0.9781 | 0.7306 | 0.6294 | 0.4348 | 0.5981 | 0.5982 |
| | ADHS-EL (DT) | 0.9984 | 0.9984 | 0.9984 | 0.9983 | 0.9981 | 0.6425 | 0.5996 | 0.8070 | 0.9248 | 0.7435 |
| | ADHS-EL (MLP) | 0.9968 | 0.9968 | 0.9967 | 0.9965 | 0.9955 | 0.9953 | 0.9080 | 0.5282 | 0.9655 | 0.8493 |
| CIC-IDS2017 | FGSM (DT) | 0.9984 | 0.9971 | 0.9970 | 0.9953 | 0.9949 | 0.6576 | 0.7206 | 0.3886 | 0.6066 | 0.5934 |
| | FGSM (MLP) | 0.9836 | 0.9706 | 0.9682 | 0.9510 | 0.9451 | 0.5850 | 0.6106 | 0.4076 | 0.9186 | 0.6305 |
| | PGD (DT) | 0.9984 | 0.9972 | 0.9970 | 0.9953 | 0.9948 | 0.6164 | 0.7897 | 0.4164 | 0.3394 | 0.5405 |
| | PGD (MLP) | 0.9837 | 0.9710 | 0.9717 | 0.9519 | 0.9535 | 0.6219 | 0.5857 | 0.4918 | 0.9349 | 0.6586 |
| | WGAN-AT(DT) | 0.9984 | 0.9972 | 0.9971 | 0.9954 | 0.9950 | 0.5022 | 0.5556 | 0.1852 | 0.4896 | 0.4332 |
| | WGAN-AT(MLP) | 0.9834 | 0.9705 | 0.9706 | 0.9510 | 0.9513 | 0.5521 | 0.5463 | 0.0791 | 0.9143 | 0.5230 |
| | Apollon | 0.9623 | 0.9378 | 0.9752 | 0.8988 | 0.9947 | 0.6496 | 0.7564 | 0.0392 | 0.6157 | 0.5152 |
| | ADHS-EL (DT) | 0.9988 | 0.9978 | 0.9980 | 0.9963 | 0.9969 | 0.7689 | 0.9875 | 0.5965 | 0.8592 | 0.8030 |
| | ADHS-EL (MLP) | 0.9959 | 0.9927 | 0.9971 | 0.9880 | 0.9988 | 0.5509 | 0.6938 | 0.1608 | 0.9276 | 0.5833 |

using an improved AdvGAN to generate adversarial samples. Furthermore, ADHS-EL only incorporates adversarial versions of boundary malicious traffic. This strategy not only enhances the models' resistance to adversarial attacks but also improves the detection accuracy of original samples.

#### 4.4.3 Statistical significance test

To evaluate the statistical significance of performance differences, we employed the Wilcoxon Signed-Rank Test to analyze the results from 10 experimental runs for each dataset, calculating the corresponding p-values. We set the significance level at 0.05 to evaluate whether the performance differences between the proposed method and the baseline methods are statistically significant. The test results (p-values) for the proposed method versus each baseline method regarding the Macro.F1, Recall, and AADR metrics on the NSL-KDD and CIC-IDS2017 datasets are shown in Table 9.

For ADHS-EL(DT), on the NSL-KDD dataset, it significantly outperforms most baseline methods, such as RUS, ROS, SMOTE, SMOTETomek, WGAN-AT(DT), and Apollon, in terms of the Macro.F1, Recall, and AADR metrics. On the CIC-IDS2017 dataset, ADHS-EL(DT) achieves significantly better results than the baseline methods across these three metrics. ADHS-EL(MLP) demonstrates significantly superior performance compared to other adversarial defense methods on the NSL-KDD dataset. Regarding the CIC-IDS2017 dataset, ADHS-EL(MLP) demonstrates comparable performance in the AADR metric but achieves significantly better results in both the Macro.F1 and Recall metrics.

Overall, compared to most baseline methods, the proposed approach exhibits a significant advantage in at least one category of metrics: either the performance metrics (Macro.F1 and Recall) or the AADR metric. It also shows comparable performance on the remaining indicators. This highlights that ADHS-EL effectively achieves both accuracy and robustness in network intrusion detection.

### 4.5 Ablation study

Three main modules are included in ADHS-EL, namely, dy namic undersampling (DUS), dynamic oversampling (DOS), and boundary malicious traffic adversarial augmentation (BMTAA). Taking the example of performing a binary classification task using DT as the base detection model on the CIC-IDS2017 dataset, we explored the effectiveness of each module, and the experimental results are shown in Table 10. Among them, the measurement unit for training time and testing time is seconds.

Compared to the original model (1), models (2) and (3) that employed dynamic undersampling and dynamic oversampling, respectively, show improvement in both accuracy and adversarial robustness. Specifically, the imbalance handling module significantly enhanced the model's defense capability against HopSkipJumpAttack. Model (4), which combines dynamic undersampling and dynamic oversampling, shortens the training time compared to model (3) and has an advantage in adversarial robustness over models (2) and (3).

**Table 9** Wilcoxon Signed-Rank Test results (p-values) for the proposed method compared with baseline methods

| Proposed method | Baseline method | NSL-KDD | | | CIC-IDS2017 | | |
|---|---|---|---|---|---|---|---|
| | | Macro.F1 | Recall | AADR | Macro.F1 | Recall | AADR |
| ADHS-EL(DT) | RUS | 0.0020 | 0.0039 | 0.0039 | 0.0020 | 0.0020 | 0.0020 |
| | ROS | 0.0020 | 0.0020 | 0.0020 | 0.0020 | 0.0039 | 0.0020 |
| | SMOTE | 0.0020 | 0.0039 | 0.0020 | 0.0020 | 0.0020 | 0.0020 |
| | ADASYN | 0.0020 | 0.0020 | 0.0645 | 0.0020 | 0.0039 | 0.0020 |
| | SMOTETomek | 0.0020 | 0.0039 | 0.0020 | 0.0020 | 0.0371 | 0.0020 |
| | DUEN | 0.5566 | 0.0371 | 0.0020 | 0.0020 | 0.0020 | 0.0020 |
| | CTGAN | 0.0020 | 0.0020 | 0.2754 | 0.0020 | 0.0020 | 0.0273 |
| | WGAN | 0.0039 | 0.0020 | 0.1055 | 0.0020 | 0.0020 | 0.0039 |
| | FGSM(DT) | 0.0020 | 0.0020 | 0.0039 | 0.0020 | 0.0020 | 0.0020 |
| | PGD(DT) | 0.0039 | 0.0020 | 0.0039 | 0.0020 | 0.0020 | 0.0020 |
| | WGAN-AT(DT) | 0.0020 | 0.0020 | 0.0039 | 0.0020 | 0.0020 | 0.0020 |
| | Apollon | 0.0020 | 0.0020 | 0.0020 | 0.0020 | 0.0020 | 0.0020 |
| ADHS-EL(MLP) | FGSM(MLP) | 0.0195 | 0.0098 | 0.0273 | 0.0020 | 0.0020 | 0.3223 |
| | PGD(MLP) | 0.0273 | 0.0059 | 0.0137 | 0.0020 | 0.0020 | 0.1934 |
| | WGAN-AT(MLP) | 0.0020 | 0.0020 | 0.0098 | 0.0020 | 0.0020 | 0.0371 |
| | Apollon | 0.0020 | 0.0020 | 0.0020 | 0.0020 | 0.0020 | 0.0098 |

**Table 10** Experimental results of modules ablation in the proposed method

| | | (1) | (2) | (3) | (4) | (5) |
|---|---|---|---|---|---|---|
| Module | DUS | – | ✓ | – | ✓ | ✓ |
| | DOS | – | – | ✓ | ✓ | ✓ |
| | BMTAA | – | – | – | – | ✓ |
| Accuracy metrics | ACC | 0.9984 | 0.9986 | 0.9987 | 0.9987 | 0.9988 |
| | Macro.F1 | 0.9972 | 0.9976 | 0.9977 | 0.9977 | 0.9978 |
| | Macro.AUC | 0.9970 | 0.9978 | 0.9979 | 0.9980 | 0.9980 |
| | F1-score | 0.9953 | 0.9959 | 0.9961 | 0.9962 | 0.9963 |
| | Recall | 0.9950 | 0.9963 | 0.9967 | 0.9970 | 0.9969 |
| Robustness metrics | ZOO | 0.0605 | 0.6953 | 0.6362 | 0.6565 | 0.7689 |
| | HSJA | 0 | 0.9911 | 0.9848 | 0.9901 | 0.9875 |
| | BA | 0 | 0.0846 | 0.3888 | 0.6036 | 0.5965 |
| | DTA | 0 | 0.3869 | 0.3733 | 0.3929 | 0.8592 |
| | Average | 0.0151 | 0.5395 | 0.5958 | 0.6608 | 0.8030 |
| Efficiency metrics | Training time | 154.5871 | 83.5418 | 467.9279 | 237.0971 | 316.5186 |
| | Test time | 0.1590 | 1.2483 | 1.2199 | 1.5737 | 1.2474 |

Model (5) achieves the best performance in terms of accuracy and robustness metrics, demonstrating the effectiveness of combining all three modules. The training time of model (5) is higher than that of models (1) and (4), with the main time consumption coming from iterative training and adversarial data augmentation.

In particular, the boundary augmentation strategy is designed for the adversarial data augmentation. The effectiveness of this strategy is analyzed by using the full augmentation as a benchmark, and the experimental results are presented in Fig. 10.

As shown in Fig. 10(a), the accuracy metrics decrease significantly in the full augmentation strategy where all malicious samples are used to generate adversarial samples, which may be due to the addition of a large number of adversarial samples potentially disrupting the original traffic distribution. As depicted in Fig. 10(b), the robustness of the model based on the full augmentation strategy is not high, which is attributed to the designed AdvGAN adversarial attack being more effective against boundary samples and less effective against non-boundary samples. In contrast, the boundary augmentation strategy plays a positive role in enhancing both model accuracy and robustness.

## 4.6 Out-of-distribution generalization analysis

We employed Test Set 2 of NSL-KDD to test the generalizability of ADHS-EL, and the comparison results with the original models are provided in Table 11.

The original models exhibit significant performance differences on Test Set 1 (which consists of 10% of the samples split from KDDTrain) and Test Set 2, with the metrics on Test Set 1 being noticeably higher than those on Test Set 2 (see Table 5), which to some extent reflects a possible inconsistency in the distribution between Test Set 2 and the training set. The proposed method achieves more significant performance improvements on Test Set 2. As shown in Table 11, for the binary classification task, the ACC, Macro.F1, and Macro.AUC metrics of the enhanced DT are improved by 3.34%, 3.41%, and 2.71%, respectively, demonstrating the generalizability of ADHS-EL on out-of-distribution traffic samples.

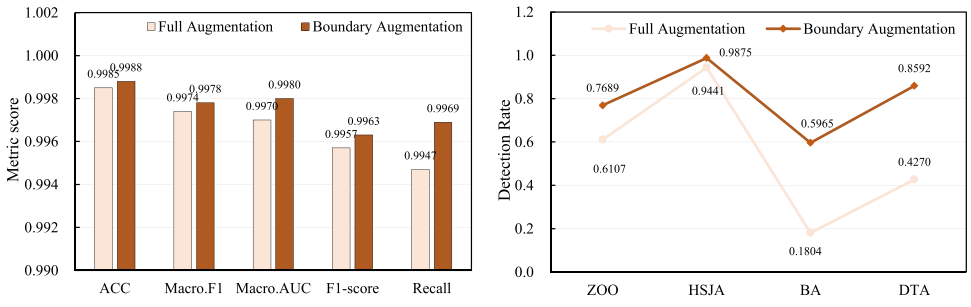



**Fig. 10** Comparison of two adversarial augmentation strategies

(a) Comparison of accuracy indicators.

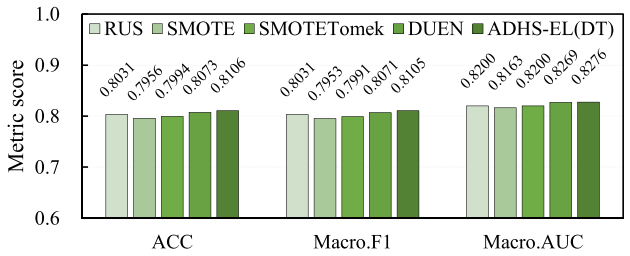
(b) Comparison of robustness indicators.

**Table 11** Performance comparison on NSL-KDD Test Set 2

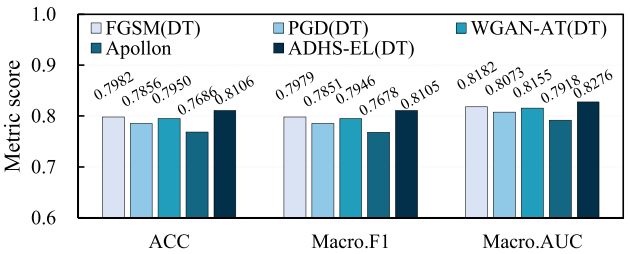
| Classification | Base classifier | Original model | | | Enhanced model | | |
|---|---|---|---|---|---|---|---|
| | | ACC | Macro.F1 | Macro.AUC | ACC | Macro.F1 | Macro.AUC |
| Binary | DT | 0.7844 | 0.7838 | 0.8058 | 0.8106 | 0.8105 | 0.8276 |
| | RF | 0.7522 | 0.7503 | 0.7792 | 0.7548 | 0.7531 | 0.7815 |
| | KNN | 0.7635 | 0.7632 | 0.7837 | 0.7746 | 0.7744 | 0.7933 |
| | MLP | 0.7908 | 0.7906 | 0.8099 | 0.7951 | 0.7950 | 0.8135 |
| Multiclass | DT | 0.7596 | 0.5185 | 0.7200 | 0.7528 | 0.5279 | 0.7824 |
| | RF | 0.7447 | 0.4811 | 0.8764 | 0.7486 | 0.5097 | 0.8989 |
| | KNN | 0.7408 | 0.4953 | 0.7339 | 0.7320 | 0.4972 | 0.7446 |
| | MLP | 0.7626 | 0.5295 | 0.8543 | 0.7527 | 0.5382 | 0.8395 |

Furthermore, on the NSL-KDD Test Set 2, the proposed method was compared with the imbalance handling and adversarial defense methods described in Sect. 4.4. Except for Apollon, the tested methods used DT as the base classifier, and the experimental results on the binary classification task are shown in Fig. 11. Among them, four representative imbalance handling methods with the best performance on the NSL-KDD Test Set 2 were selected for display.

As seen in Fig. 11, ADHS-EL achieves the highest scores in ACC, Macro.F1, and Macro.AUC metrics when compared to the imbalance handling and adversarial defense approaches. The performance of the proposed method remains optimal on the test samples that differ from the training set distribution. Essentially, adversarial and out-of-distribution samples are data patterns not contained in the training set. The combination of imbalance handling and adversarial training in ADHS-EL is beneficial in improving the quality of the training set and obtaining the model with both accuracy and robustness.




(a) Comparison with class imbalance handling methods.




(b) Comparison with adversarial defense methods.

**Fig. 11** Comparison with baseline methods on the NSL-KDD Test Set 2

## 5 Discussion

This section discusses the deployment of ADHS-EL in intrusion detection systems, the factors that affect model performance in real-world network environments, the limitations of the proposed method, and potential solutions.

### 5.1 Practical application

ADHS-EL can be integrated into real-world network intrusion detection systems to enhance their detection accuracy and robustness. Specifically, the first step is to evaluate the architecture and data flow of the current system to ensure compatibility with ADHS-EL. Next, the necessary software libraries and tools are installed in the development environment, and the ADHS-EL model is trained and evaluated using labeled network traffic data collected from the target network. Subsequently, an application programming interface is designed to deploy the ADHS-EL model into the system, ensuring it can process network traffic data and output detection results. Finally, comprehensive functional, performance, and security tests are conducted to ensure that ADHS-EL can seamlessly integrate and enhance the overall detection capabilities. In practical operation, ADHS-EL can work in conjunction with existing signature-based and anomaly detection algorithms, combining the strengths of multiple detection methods to not only improve the accuracy of known attack identification but also effectively detect adversarial attack patterns, thereby significantly enhancing the system's overall defense capabilities and providing more comprehensive and reliable cybersecurity.

Compared to the controlled datasets used in experimental evaluations, the performance of ADHS-EL in real-world scenarios is influenced by several factors. First, the diversity

and complexity of network traffic are Crucial. The attack traffic in real-world networks is more varied and continuously evolving, requiring ADHS-EL to have high adaptability and generalization capabilities. Second, the dynamic nature of network environments, including changes in traffic patterns, user behavior, and system configurations, can affect the model's detection effectiveness. Additionally, computational resource constraints, especially in resource-limited environments such as edge devices, necessitate optimization to ensure efficient operation. These factors collectively determine the performance and reliability of ADHS-EL in practical deployment.

### 5.2 Limitations and potential solutions

Despite the promising results of our research, several limitations still exist. Firstly, ADHS-EL has limited capability to defend against adversarial attacks. As shown in Figs. 7 and 8, although the proposed method effectively reduces the effectiveness of adversarial attacks, it does not completely eliminate them. The enhanced robustness of ADHS-EL is partly attributed to the inclusion of adversarial patterns of malicious traffic in the training set. However, the adversarial samples generated using improved AdvGAN may not fully represent all possible scenarios. Secondly, ADHS-EL demonstrates limited generalization capability when handling traffic with distribution changes. As evident from Tables 10 and 11, while the proposed method improves the detection performance for out-of-distribution traffic to some extent, its performance remains significantly lower than for in-distribution traffic. This is because ADHS-EL relies on patterns learned from historical data, making it challenging to identify emerging traffic patterns that differ from known ones. Additionally, although the proposed method enhances the accuracy and robustness of the intrusion detection model, it also increases the training time. Therefore, a trade-off between training time and model performance must be considered when deploying ADHS-EL in resource-constrained environments.

To address the limitations of ADHS-EL, we propose several potential solutions. The training dataset can be expanded to include a broader range of adversarial attack types to improve the model's ability to defend against adversarial attacks. Additionally, the intrusion detection model can be improved by increasing the time and resources required for adversarial attacks, such as by introducing randomness into the model. For the issue of limited generalization to traffic with distribution changes, the model can be regularly updated through online learning or incremental learning, thereby enhancing its adaptability to evolving network traffic. The implementation of these solutions is expected to significantly improve the practicality and effectiveness of the model and will be a key focus of our future research efforts.

## 6 Conclusion

In this paper, we discuss the challenges faced by network intrusion detection models based on machine learning techniques, including traffic class imbalance and adversarial attacks. To address these issues, we propose an ensemble learning method called ADHS-EL to provide an accurate and robust intrusion detection model. Our proposed method integrates a dynamic hybrid sampling with adversarial augmentation into the Boosting ensemble framework, providing balanced data subsets that incorporate adversarial samples for iterative training. In the proposed sampling mechanism, we utilize classification hardness to identify sample types and perform dynamic hybrid sampling with the guiding principle of enhancing the importance of boundary samples to balance the traffic class distribution. Subsequently, we perform adversarial augmentation on boundary malicious samples to introduce adversarial samples into the training set. Extensive experimental evaluation results demonstrate that ADHS-EL can improve the accuracy of the base model and its robustness against adversarial attacks, outperforming the compared imbalance handling and adversarial defense techniques. Future work will focus on mitigating the limitations in defending against adversarial attacks and generalizing to out-of-distribution traffic, as well as exploring the application of our method in real-world network environments.

**Author Contributions** Huajuan Ren: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft. Yonghe Tang: Validation, Formal analysis, Writing - Review & Editing. Shuai Ren: Software, Data curation, Visualization. Ruimin Wang: Validation, Writing - Review & Editing. Weiyu Dong: Validation, Supervision, Funding acquisition.

**Funding** This work was supported by the Key R&D Project in Henan Province of China (grant number 221111210300) and the Natural Science Foundation of Henan Province of China (grant number 242300420698).

**Data Availability** Data will be made available on reasonable request.

## Declarations

**Conflict of interest** The authors have no relevant financial or non-financial interests to disclose.

## References


Abou Khamis R, Shafiq MO, Matrawy A (2020) Investigating resistance of deep learning-based IDS against adversaries using min-max optimization. In: 2020 IEEE international conference on communications (ICC), pp 1–7. https://doi.org/10.1109/ICC40277.2020.9149117

Aleksander M, Aleksandar M, Ludwig S et al (2019) Towards deep learning models resistant to adversarial attacks. arXiv:1706.06083

Arjovsky M, Chintala S, Bottou L (2017) Wasserstein Generative Adversarial Networks. In: ICML'17: Proceedings of the 34th international conference on machine learning, pp 214–223. https://doi.org/10.5555/3305381.3305404

Brendel W, Rauber J, Bethge M (2017) Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. arXiv:1712.04248

Carlini N, Wagner D (2017) Towards Evaluating the Robustness of Neural Networks. In: 2017 IEEE symposium on security and privacy (SP), pp 39–57. https://doi.org/10.1109/SP.2017.49

Casas P, Marín G, Capdehourat G et al (2019) MLSEC-benchmarking shallow and deep machine learning models for network security. In: 2019 IEEE security and privacy workshops (SPW), pp 230–235. https://doi.org/10.1109/SPW.2019.00050

Chen PY, Zhang H, Sharma Y et al (2017) ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In: AISec'17: Proceedings of the 10th ACM workshop on artificial intelligence and security, pp 15–26. https://doi.org/10.1145/3128572.3140448

Chen J, Jordan MI, Wainwright MJ (2020) HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. In: 2020 IEEE symposium on security and privacy (SP), pp 1277–1294. https://doi.org/10.1109/SP40000.2020.00045

Duy PT, Khoa NH, Do Hoang H et al (2023) Investigating on the robustness of flow-based intrusion detection system against adversarial samples using generative adversarial networks. J Inf Secur Appl 74:103472. https://doi.org/10.1016/j.jisa.2023.103472

Engelen G, Rimmer V, Joosen W (2021) Troubleshooting an intrusion detection dataset: the CICIDS2017 Case Study. In: 2021 IEEE security and privacy workshops (SPW), pp 7–12. https://doi.org/10.1109/SPW53761.2021.00009

Goodfellow IJ, Shlens J, Szegedy C (2014a) Explaining and harnessing adversarial examples. arXiv:1412.6572

Goodfellow I, Pouget-Abadie J, Mirza M et al (2014b) Generative adversarial nets. In: NIPS'14: Proceedings of the 27th international conference on neural information processing systems, pp 2672–2680. https://doi.org/10.5555/2969033.2969125

Han D, Wang Z, Zhong Y et al (2021) Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors. IEEE J Sel Areas Commun 39(8):2632–2647. https://doi.org/10.1109/JSAC.2021.3087242

Jiang H, Lin J, Kang H (2022) FGMD: A robust detector against adversarial attacks in the IoT network. Future Gener Comput Syst 132:194–210. https://doi.org/10.1016/j.future.2022.02.019

Khan FA, Gumaei A, Derhab A et al (2019) A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access 7:30373–30385. https://doi.org/10.1109/ACCESS.2019.2899721

Korium MS, Saber M, Beattie A et al (2024) Intrusion detection system for cyberattacks in the internet of vehicles environment. Ad Hoc Network 153:103330. https://doi.org/10.1016/j.adhoc.2023.103330

Li Y, Abdallah S (2023) A multi-stage automated online network data stream analytics framework for iiot systems. IEEE Trans Ind Informatics 19(2):2107–2116. https://doi.org/10.1109/TII.2022.3212003

Lichy A, Bader O, Dubin R et al (2023) When a RF beats a CNN and GRU, together-A comparison of deep learning and classical machine learning approaches for encrypted malware traffic classification. Comput Secur 124:103000. https://doi.org/10.1016/j.cose.2022.103000

Lin Z, Shi Y, Xue Z (2022) IDSGAN: Generative adversarial networks for attack generation against intrusion detection. In: Pacific-Asia conference on knowledge discovery and data mining, pp 79–91. https://doi.org/10.1007/978-3-031-05981-0_7

Liu J, Gao Y, Hu F (2021) A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. Comput Secur 106:102289. https://doi.org/10.1016/j.cose.2021.102289

Liu Z, Cao W, Gao Z et al (2020) Self-paced ensemble for highly imbalanced massive data classification. In: 2020 IEEE 36th international conference on data engineering (ICDE), pp 841–852,. https://doi.org/10.1109/ICDE48307.2020.00078

Li W, Yi P, Wu Y et al (2014) A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network. J Electr Comput Eng 2014. https://doi.org/10.1155/2014/240217

Mohanty H, Roudsari AH, Lashkari AH (2022) Robust stacking ensemble model for darknet traffic classification under adversarial settings. Comput Secur 120:102830. https://doi.org/10.1016/j.cose.2022.102830

Moosavi-Dezfooli SM, Fawzi A, Frossard P (2016) DeepFool: a simple and accurate method to fool deep neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2574–2582. https://doi.org/10.1109/CVPR.2016.282

Papernot N, McDaniel P, Goodfellow I (2016) Transferability in machine learning: from Phenomena to Black-Box Attacks using adversarial samples. arXiv:1605.07277

Pawlicki M, Choraś M, Kozik R (2020) Defending network intrusion detection systems against adversarial evasion attacks. Future Gener Comput Syst 110:148–154. https://doi.org/10.1016/j.future.2020.04.013

Paya A, Arroni S, García-Díaz V et al (2024) Apollon: A robust defense system against adversarial machine learning attacks in intrusion detection systems. Comput Secur 136:103546. https://doi.org/10.1016/j.cose.2023.103546

Ren H, Tang Y, Dong W et al (2023) DUEN: Dynamic ensemble handling class imbalance in network intrusion detection. Expert Syst Appl 229:120420. https://doi.org/10.1016/j.eswa.2023.120420

Roshan K, Zafar A, Haque SBU (2024) Untargeted white-box adversarial attack with heuristic defence methods in real-time deep learning based network intrusion detection system. Comput Commun 218:97–113. https://doi.org/10.1016/j.comcom.2023.09.030

Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP 2018: Proceedings of the 4th international conference on information systems security and privacy, pp 108–116. https://doi.org/10.5220/0006639801080116

Shin Y, Kim M, Kim H et al (2024) Towards unbalanced multiclass intrusion detection with hybrid sampling methods and ensemble classification. Appl Soft Comput 157:111517. https://doi.org/10.1016/j.asoc.2024.111517

Tavallaee M, Bagheri E, Lu W et al (2009) A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on

computational intelligence for security and defense applications, pp 1–6. https://doi.org/10.1109/CISDA.2009.5356528

Tian Q, Han D, Hsieh MY et al (2021) A two-stage intrusion detection approach for software-defined IoT networks. Soft Comput 25:10935–10951. https://doi.org/10.1007/s00500-021-05809-y

Usama M, Asim M, Latif S et al (2019) Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In: 2019 15th International wireless communications & mobile computing conference (IWCMC), pp 78–83. https://doi.org/10.1109/IWCMC.2019.8766353

Vinayakumar R, Alazab M, Soman KP et al (2019) Deep learning approach for intelligent intrusion detection system. IEEE Access 7:41525–41550. https://doi.org/10.1109/ACCESS.2019.2895334

Vinayakumar R, Soman K, Poornachandran P (2017a) Applying convolutional neural network for network intrusion detection. In: 2017 International conference on advances in computing, communications and informatics (ICACCI), pp 1222–1228. https://doi.org/10.1109/ICACCI.2017.8126009

Vinayakumar R, Soman K, Poornachandran P (2017b) Evaluation of recurrent neural network and its variants for intrusion detection system (IDS). International Journal of Information System Modeling and Design 8(3):43–63. https://doi.org/10.4018/IJISMD.2017070103

Wang S, Xu W, Liu Y (2023) Res-TranBiLSTM: An intelligent approach for intrusion detection in the Internet of Things. Comput Networks 235:109982. https://doi.org/10.1016/j.comnet.2023.109982

Xiao C, Li B, Zhu JY et al (2018) Generating adversarial examples with adversarial networks. In: IJCAI'18: Proceedings of the 27th international joint conference on artificial intelligence, pp 3905–3911. https://doi.org/10.24963/ijcai.2018/543

Xu L, Skoularidou M, Cuesta-Infante A et al (2019) Modeling tabular data using conditional GAN. In: NIPS'19: Proceedings of the 33th international conference on neural information processing systems, pp 7333–7343. https://doi.org/10.5555/3454287.3454946

Zhang C, Costa-Pérez X, Patras P (2020a) Tiki-Taka: Attacking and Defending Deep Learning-based Intrusion Detection Systems. In: CCSW'20: Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop, pp 27–39. https://doi.org/10.1145/3411495.3421359

Zhang J, Xu X, Han B et al (2020b) Attacks which do not kill training make adversarial learning stronger. In: ICML'20: Proceedings of the 37th International Conference on Machine Learning, pp 11278–11287. https://doi.org/10.5555/3524938.3525984

Zhang J, Zhu J, Niu G et al (2020c) Geometry-aware instance-reweighted adversarial training. arXiv:2010.01736

Zhao X, Huang G, Jiang J et al (2021) Research on lightweight anomaly detection of multimedia traffic in edge computing. Comput Secur 111:102463. https://doi.org/10.1016/j.cose.2021.102463

Zhao X, Fok KW, Thing VL (2024) Enhancing network intrusion detection performance using generative adversarial networks. Comput Secur 145:104005. https://doi.org/10.1016/j.cose.2024.104005