ConFlow: Contrast Network Flow Improving Class-Imbalanced Learning in Network Intrusion Detection

Lan Liu[1], Pengcheng Wang[1], Jianliang Ruan[1(✉)], Jun Lin[2,3], and Junhan Hu[1]

[1] Guangdong Polytechnic Normal University, Guangzhou 510655, China
ruanjianliang@gpnu.edu.cn
[2] Sun Yat-Sen University, Guangzhou 510006, China
[3] China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou 510610, China

Abstract. With the increasing complexity and volume of network traffic, accurate detection of malicious network attacks by machine learning-based network intrusion detection systems (NIDSs) remains a challenging task due to imbalanced network traffic. Conventional machine learning algorithms prioritize high overall accuracy without considering class imbalances. To address this issue, we propose ConFlow, a contrastive learning method for network intrusion detection. ConFlow leverages the Dropout layer to obtain two different vector representations of the same traffic, applying supervised contrast loss and cross-entropy loss during training. Experimental results on the ISCX-IDS2012 and CSE-CIC-IDS2017 datasets show that ConFlow outperforms other methods, especially in few-shot learning scenarios, and exhibits high generalization and robustness in real network environments. Our proposed method has significant practical implications for building an intrusion detection system with high accuracy and low false positive rates.

Keywords: Cyber security · Network intrusion detection system · Deep learning · Class-imbalanced · Contrastive learning

1 Introduction

In recent years, the expansion of cyber threats has led to the widespread use of security tools such as firewalls, antivirus software, malware detection, and spam filters to protect networks. However, among these tools, network intrusion detection systems (NIDS) play a critical role in network security. By monitoring network traffic, NIDS can swiftly identify unusual traffic and attack behaviour, thereby improving network security and reliability [1]. It protects network resources such as computers, servers, and data from attacks, destruction,

This research is supported by Special focus areas for general universities in Guangdong Province(2022ZDZX1015).

J. Cai et al. (Eds.): SPNCE 2023, LNICST 525, pp. 125–146, 2025.
https://doi.org/10.1007/978-3-031-73699-5_9

unauthorized access, and modification. In doing so, it ensures the availability, confidentiality, and integrity of these resources. NIDS can operate as an expert system based on signature matching, matching suspicious traffic with a predefined signature library of known attacks, or rely on anomaly detection, which measures the difference between observed traffic and trusted baseline traffic.

Currently, traditional machine learning methods are used to detect abnormal traffic through classification, such as support vector machine (SVM), decision tree (DT), random forest (RF), and naïve Bayes. However, the performance of these methods depends on the quality of feature extraction algorithms, and this performance can degrade due to increasingly complex network traffic.

Deep learning, which is an essential branch of machine learning, can mine the latent features of high-dimensional data, transform the problem of network traffic anomaly detection into a classification problem, and enhance the accuracy of intrusion processing through adaptive learning of the difference between normal and abnormal behaviour. Deep learning algorithms such as multi-layer perceptron (MLP), recurrent neural network (RNN), and convolutional neural network (CNN) have been introduced to improve the performance of intrusion detection systems. However, in the real network world, the majority of traffic is normal, and a small number of malicious attacks can be hidden in this large amount of normal traffic, resulting in a highly imbalanced distribution of traffic and a biased classification model towards normal traffic. Despite extensive research on intrusion detection, class imbalance remains an important factor limiting its performance [2].

Recently, supervised contrastive learning has demonstrated good results in representation learning for computer vision (CV) and natural language processing (NLP) [3]. The core idea of supervised contrastive loss is to cluster point clusters of the same class together in the embedding space by effectively utilizing label information while separating samples from different classes [4]. In the context of intrusion detection, the problem can be viewed as separating normal traffic from abnormal traffic.

To build an effective intrusion detection model, the availability of large and well-distributed data is necessary, and the performance may drop significantly when learning on extremely imbalanced or data-starved datasets. However, due to data sparsity, limitations, privacy, and sensitivity, some traffic classes may not have enough data to train a learning model. Therefore, it is essential to establish a network flow feature extraction model and improve network traffic class imbalance learning to develop a network intrusion detection model that can address the real-world traffic environment.

Taking inspiration from the success of supervised contrastive learning in CV and NLP, we introduce it to compare network flows for network intrusion detection. Our goal is to develop a robust and generalizable intrusion detection model that can address the class-imbalanced problem in intrusion detection data. We use supervised contrastive learning to compare network flows by weighting supervised contrastive loss and cross-entropy loss to train the model. Our approach outperforms state-of-the-art baselines on the ISCX-IDS2012 and CIC-IDS2017 datasets. Our contributions are as follows:

(1) Network flow encoder: We design a bidirectional network flow feature extraction encoder that enhances the neural network's representation ability using Gaussian Error Linear Units (GELU), Layer Normalization (LayerNorm), and Skip-connection units in the Multi-Layer Perceptron (MLP). This enables the network to learn to represent network flow data automatically and reduce reliance on feature engineering.
(2) Training framework and loss function: We use the encoder's dropout layer to obtain augmentation data, weigh supervised contrastive loss and cross-entropy loss in the training phase, and directly further mine malicious attacks hidden in normal traffic without requiring additional data augmentation techniques or two-stage training of contrastive learning pre-training and fine-tuning.
(3) Generalization and robustness testing: We explore the performance of our proposed method on few-shot network traffic data with a few labeled samples (10, 100, and 1000) and perform cross-testing on the two datasets to verify ConFlow's generalization and robustness in real network environments.

The remainder of the paper is organized as follows: Sect. 2 introduces the state-of-the-art in advanced network intrusion detection and summarizes related methods for dealing with imbalanced data. Section 3 details our proposed method, and Sect. 4 validates our approach and analyzes the results. Finally, Sect. 5 concludes the paper by summarizing our proposed research method and discussing possible future research directions.

2 Related Work

In this section, we summarize the algorithms and research work related to this study.

2.1 Intrusion Detection System

With the development of computers and the rapid rise of malware, anomaly-based intrusion detection systems have been proposed to detect unknown malware attacks. The use of machine learning and deep learning has become the mainstream research direction for developing detection models, since it is able to identify variant and unknown threats, compensating for the deficiency of traditional feature and behavior detection methods that detect known attacks only. The technology of threat detection brings forth higher requirements.

Machine learning-based network intrusion detection can be classified into supervised and unsupervised learning. In supervised learning, it can be further classified into generative and discriminative methods according to the modeling algorithm. The generation algorithm aims to learn the distribution of network traffic data from a statistical perspective by reflecting the similarity of the same type of traffic data, such as the Bayesian algorithm [5], Hidden Markov

Model (HMM) [6], and Restricted Boltzmann Machine (RBM) [7]. Meanwhile, the discriminant method can differentiate different traffic data types by learning decision functions or conditional probability distributions as prediction models, such as the K nearest neighbor algorithm (KNN) [8], decision tree (DT) and Support Vector Machine (SVM) [9], logistic regression (LR) [10], and other methods. Unsupervised learning can develop models via dimensionality reduction, clustering, and other methods without relying on network traffic labels. The k-means algorithm is simple and easy to implement, and has been widely adopted in intrusion detection [11]. Hierarchical clustering can discover the hierarchical relationships of classes by calculating the distance between samples and is frequently used for anomaly mining in redundant network traffic [12]. Gaussian Mixture Model (GMM) [13] can learn the probability density function and can identify different types of network traffic with similar distributions; Principal Component Analysis (PCA) [14] is the most commonly used dimensionality reduction method, which can reduce the feature dimension of network traffic and is therefore used in a large number of feature dimension reduction papers.

Over the past decade, deep learning has been extensively employed in intrusion detection models. These methods can generate excellent prediction results by learning the practical features of the data. Network intrusion detection systems based on deep learning mostly exploit supervised and unsupervised learning models to learn the latent features of high-dimensional data. Supervised learning includes autoencoder (AE), deep belief network (DBN), recurrent neural network (RNN), convolutional neural networks (CNN), whereby the network traffic feature data is converted into 1-D sequences, text, or images, and neural networks are utilized for classification [15]. Other researchers employ reinforcement learning for sample selection in the training dataset and to motivate detection results [16], use generative adversarial networks (GAN) for few-class attack generation [17], and utilize Transformers to develop more complex classification models [18]. Unsupervised learning, that is, self-supervised Learning (SSL), creates supervised information through data augmentation and pre-trains large-scale unlabeled network flow, obtaining the features of different network traffic [19].

2.2 Imbalanced Learning

Class-imbalanced data can cause the model to learn better on majority class samples but generalize poorly on minor class samples. To address the problem of network imbalance learning, many solutions have been proposed by scholars, which are summarized as follows:

(1) Resampling: Liu et al. proposed oversampling the minority samples (usually malicious) in network traffic [20], while Zuech et al. suggested undersampling the majority samples [21]. However, over-sampling can easily lead to over-fitting of few-class samples, resulting in worse performance on highly imbalanced data. Meanwhile, under-sampling can result in significant information loss of many types, leading to under-fitting. Thus, Bagui et al. combined these two methods to balance the distribution of data types [22].

(2) Data synthesis: Generating "new" data similar to minority samples. The classical method SMOTE and its improvements [13,23,24] arbitrarily select few-class samples, use K nearest neighbors to select similar samples, and generate new samples by linear interpolation of the samples. Other scholars have exploited generative adversarial network framework to create adversarial few-class malicious traffic, improving the model's robustness [25,26].
(3) Reweighting: Assigning different weights to categories of different network traffic. However, the simplest weighting method based on the inverse of the number of categories can be challenging for improving the learning of imbalanced networks since the network traffic distribution is highly skewed. Therefore, some scholars dynamically weight according to the number of "effective" samples to optimize the loss weighting of the classification spacing based on the number of samples [27].

2.3 Contrastive Learning

Contrastive learning (CL) is a technique for extracting information from data by constructing positive and negative sample pairs. The core idea behind contrastive learning is to use a contrastive loss function in which the model pulls similar samples closer together, drives dissimilar samples further apart, and compares the representations of similar and dissimilar data.

In recent years, contrastive learning has become popular in computer vision (CV). For example, Chen et al. proposed SimCLR [28], a framework that has been widely adopted in the CV community. SimCLR enhances pairs of images using data augmentation, encodes image representations using an image encoder, maps the image representations to one dimension using a projection head, and calculates the distance between the sample representations using InfoNCE. This approach has achieved a 7% improvement on ImageNet. However, the training process can be time-consuming and computationally intensive due to the need for large batch sizes to ensure contrastive effects. To address this issue, He et al. [29] introduced MoCo (Momentum Contrast), which uses a memory bank to store the representations of previously processed samples. Caron et al. [30] introduced SwAV, a method that clusters various types of samples and distinguishes the clusters of each class. Khosla et al. [4] proposed a supervised contrastive loss that outperforms cross-entropy loss by clustering point clusters belonging to the same class while separating points from different classes in the embedding space.

Recently, contrastive learning has also been applied to natural language processing (NLP). For example, Gunel et al. [31] proposed a new approach that combines supervised contrastive loss and cross-entropy loss with weighting during the fine-tuning stage of the BERT model. Yan et al. [32] proposed ConSERT, a contrastive framework for self-supervised sentence representation transfer that uses contrastive learning to fine-tune BERT in an unsupervised and efficient manner. Gao et al. [33] proposed SimCSE, a simple contrastive learning approach for generating high-quality sentence vectors.

Contrastive learning has also been applied in intrusion detection. For example, Wang et al. [19] proposed a data augmentation strategy for intrusion detec-

tion data and an intrusion detection model based on unlabeled self-supervised learning. The proposed model was trained on the unlabeled UNSW-NB15 intrusion detection dataset and achieved excellent performance on all metrics. Similarly, Lopez et al. [34] proposed a scheme that uses a shared embedding space to include labels in the same space as features and performs distance comparisons between features and labels. This approach dramatically reduces the number of pairwise comparisons, improving the model's performance.

3 Method

In this section, we propose a network flow encoder and training method for network traffic to improve class-imbalanced learning in NIDS.

3.1 Data Preprocessing

Due to data sparsity, data limitations, data privacy, and data sensitivity, we extract features at the network layer (L3). These features only perform statistics on behavioral features and do not require investigation into the data packets in-depth. Network flow division is determined by the seven-tuple network session (source IP address, destination IP address, protocol, source port, destination port, VLAN identifier, tunnel identifier).

We utilized the NFStream [35] tool to parse network packets (PCAP). This tool provides state-of-the-art flow-based statistical feature extraction. It includes both post-mortem statistical features (e.g., min, mean, stddev, and max of packet size and inter-arrival time) and early flow features (e.g., sequence of first n packets sizes, inter-arrival times, and directions). We set time thresholds, such as active timeout and inactive timeout, and performed behavioral statistics and feature extraction for source to destination flows, destination to source flows, and bidirectional flows. The features extracted by NFStream include continuous features and categorical features, with a total of 57 dimensions in Table 1.

Table 1. Features of network flow by NFStream(including IP source address to destination flow features (src2dst features), IP destination to source address flow features (dst2src features), and bidirectional flow features (bidirectional features)).

src2dst features	dst2src features	bidirectional features
src2dst_duration_ms	dst2src_duration_ms	bidirectional_duration_ms
src2dst_packets	dst2src_packets	bidirectional_packets
src2dst_bytes	dst2src_bytes	bidirectional_bytes
src2dst_syn_packets	dst2src_syn_packets	bidirectional_syn_packets
src2dst_cwr_packets	dst2src_cwr_packets	bidirectional_cwr_packets
src2dst_ece_packets	dst2src_ece_packets	bidirectional_ece_packets

continued

Table 1. continued

src2dst features	dst2src features	bidirectional features
src2dst_urg_packets	dst2src_urg_packets	bidirectional_urg_packets
src2dst_ack_packets	dst2src_ack_packets	bidirectional_ack_packets
src2dst_psh_packets	dst2src_psh_packets	bidirectional_psh_packets
src2dst_rst_packets	dst2src_rst_packets	bidirectional_rst_packets
src2dst_fin_packets	dst2src_fin_packets	bidirectional_fin_packets
src2dst_min_piat_ms	dst2src_min_piat_ms	bidirectional_min_piat_ms
src2dst_mean_piat_ms	dst2src_mean_piat_ms	bidirectional_mean_piat_ms
src2dst_stddev_piat_ms	dst2src_stddev_piat_ms	bidirectional_stddev_piat_ms
src2dst_max_piat_ms	dst2src_max_piat_ms	bidirectional_max_piat_ms
src2dst_min_ps	dst2src_min_ps	bidirectional_min_ps
src2dst_mean_ps	dst2src_mean_ps	bidirectional_mean_ps
src2dst_stddev_ps	dst2src_stddev_ps	bidirectional_stddev_ps
src2dst_max_ps	dst2src_max_ps	bidirectional_max_ps

3.2 Network Flow Encoder

Popular complex network structures, such as CNN and Transformer, have widely demonstrated good performance in computer vision and natural language processing tasks. However, these structures may not be necessary for relatively simple network traffic representation learning. In this study, we propose an architecture that uses MLP and skip-connections as the main components, as shown in Fig. 1, to serve as the representation encoder network for network traffic.

In the proposed network flow feature extraction encoder, we encode categorical features in the input using one-hot encoding and dropout layers, while continuous features are normalized through LayerNorm layers. To enable better information interaction between categorical and continuous features, they are stitched together when inputted to the next neural unit. The main structure of the network includes two Dense Resnet Block (DRB) modules which employ the Skip-connection mechanism to efficiently reduce gradient disappearance and network degradation problems, thus making training easier. After the DRB module, an MLP unit consisting of linear layers, GELU, and Dropout components is utilized to output the representation information extracted by the encoder to a fixed dimension. Finally, the network flow vector obtained by the encoder is mapped to either the classification category dimension or the projection dimension on the unit sphere through a fully connected layer (FC).

The DRB module includes various components such as linear layer, LayerNorm, GELU, and Dropout, which enhance the expressiveness of the network flow embeddings. LayerNorm in the encoder can provide different network features without requiring other data, thus avoiding the issue of mini-batch data distribution in BatchNorm. Additionally, some studies found that BatchNorm can negatively impact network performance, particularly causing information

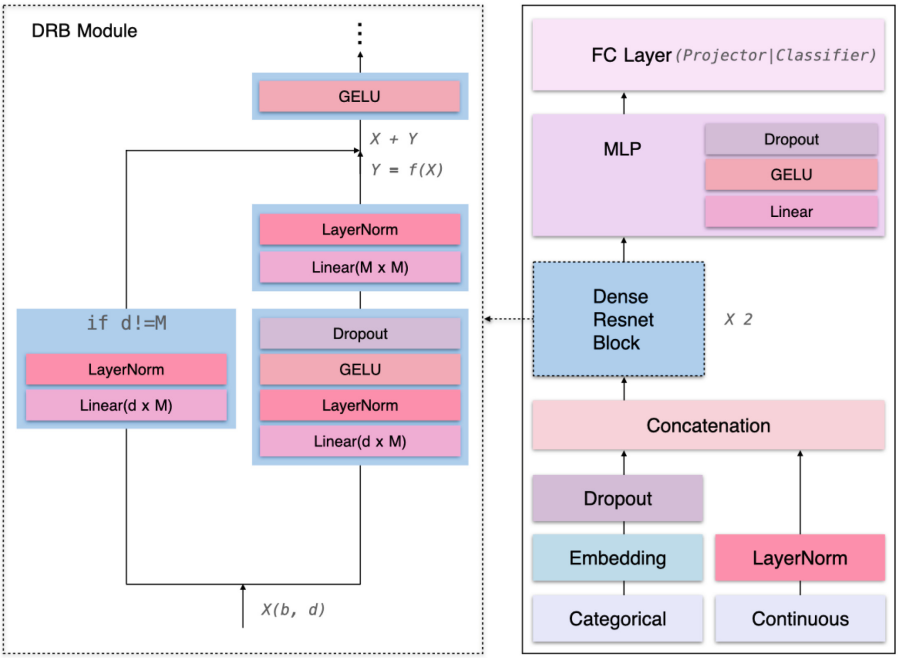

Fig. 1. Encoder structure for network flows.

leakage in contrastive learning [36]. Furthermore, GELU (Gaussian Error Linear Unit), which is smoother than ReLU and is utilized by NLP models such as BERT, GPT-3, and ViT, is also adopted in this study to replace the commonly used ReLU activation function.

3.3 Training Framework and Loss Function

It is challenging to manually construct data augmentation techniques for network traffic that guarantee semantic invariance. Dropout is commonly utilized to prevent model overfitting in deep learning. SimCSE [33] and R-Drop [37] leverage the randomness of Dropout to rapidly expand data and achieve satisfactory results, as demonstrated on several datasets from different domains. Hence, we employ the "Dropout twice" approach to generate positive samples for supervised contrastive learning. Specifically, a network traffic sample passes through the model twice to obtain different feature vectors of the same input.

However, using Dropout for data augmentation leads to similar positive samples, resulting in feature suppression in the network flow. The deep learning model can not differentiate between sample similarities and class similarities, leading to a bias towards having a large number of normal traffic samples without considering the actual class differences.

Motivated by the success of supervised contrastive learning in CV and NLP, we adopt it to compare network flows. It utilizes sample labels to create new sample pairs to draw samples of the same class closer and away from different classes. By inputting the model twice to obtain different vector representations of the same input and calculating their similarity, we can obtain a similarity matrix with the input's size. We design a supervised contrast loss on this similarity matrix, where the larger the sum of similarity distances of the same type of traffic, the better; and the smaller the sum of similarity distances of different classes, the better.

The architecture of ConFlow is shown in Fig. 2. The first input and output pipeline of network flow are represented as the solid line part, and the second time is represented by the dotted line part. The encoder network maps the traffic to the representation space, where each flow is input to the same encoder twice, yielding a pair of representation vectors. Finally, a 512-dimensional representation vector is obtained using a multilayer perceptron with only one hidden layer. The projection network maps the representation vector to a final vector for loss calculation via a fully connected layer with a dimension of 64. To maximize the similarity between augmentation flow projections and minimize the similarity between different flows, a unit hypersphere is utilized for regularization. The classification network maps the dimensions to the label category dimension using a linear layer.

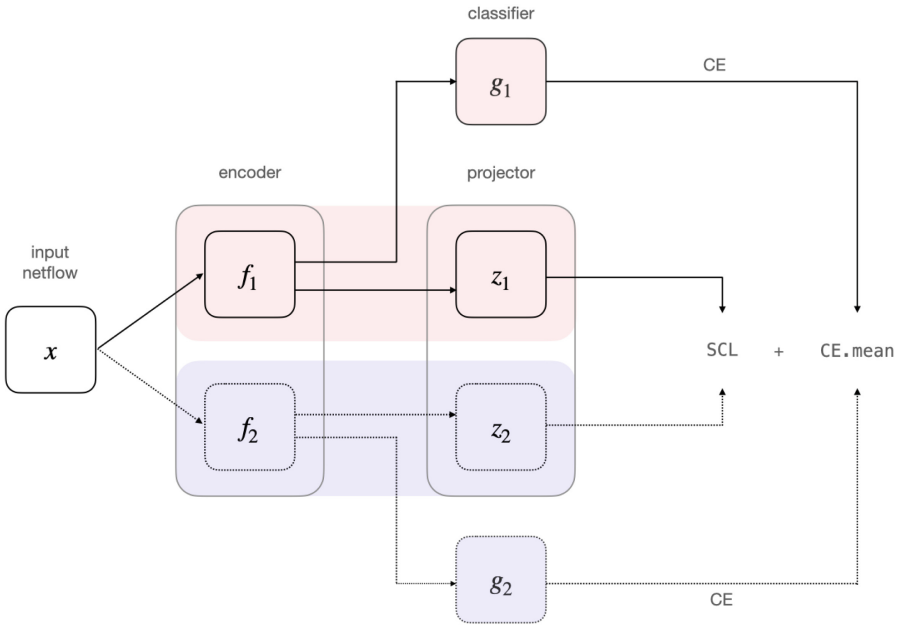

Fig. 2. ConFlow's framework.

ConFlow introduces supervised contrastive loss into the standard training method to detect malicious attacks obscured by large volumes of normal traffic. By inputting the model twice, the final loss is calculated based on a weighting of the supervised contrastive loss and the average loss of the two cross-entropies. Additional data augmentation methods are employed, and a two-stage training process is used that includes traditional contrastive learning pre-training followed by fine-tuning.

The loss function used is a weighted cross-entropy (CE) loss with the inclusion of supervised contrastive loss (SCL), which is formulated as follows:

$$\mathscr{L}_{CE} = -\frac{1}{2N}\left(\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \cdot \log \hat{y}_{i,c'} + \sum_{i=1}^{N}\sum_{c'=1}^{C'} y_{i',c'} \cdot \log \hat{y}_{i',c'}\right) \tag{1}$$

$$\mathscr{L}_{SCL} = \sum_{i=1}^{2N} -\frac{1}{2N_{y_i} - 1} \sum_{j=1}^{2N} \mathbf{1}_{i \neq j} \mathbf{1}_{y_i = y_j} \log \frac{\exp\left(z\left(x_i\right) \cdot z\left(x_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbf{1}_{i \neq k} \exp\left(z\left(x_i\right) \cdot z\left(x_k\right)/\tau\right)} \tag{2}$$

$$\mathscr{L}_{OSS} = (1 - \lambda)\mathscr{L}_{CE} + \lambda \mathscr{L}_{SCL} \tag{3}$$

The canonical definition of the CE loss that we use is given in equation (1). The SCL loss is given in the equation (2). The overall loss is a weighted average of CE and SCL loss, as given in the equation (3).

For the intrusion detection case of class C, we use a batch of size N, $\{xi, yi\}_{i=1,\ldots N}$. $z(\cdot)$ denotes the encoder that outputs the l_2 normalized final encoder hidden layer before projection. The batch size is $2N$ after passing through the encoder twice; $2N_{y_i}$ is the total number of examples in the batch with the same label as y_i; $\tau > 0$ is an adjustable scalar temperature parameter that controls the separation of classes; yi, c represent labels and $\hat{y}_i, c$ represents the model output for the probability of the ith example belonging to class c; λ is a scalar weighted hyperparameter tuned by the setting.

4 Experiments

4.1 Benchmark Dataset

Obtaining datasets that accurately reflect real-world cyberspace is challenging due to the rarity and secrecy of network intrusion attacks. Many studies still rely on outdated network traffic data, such as the KDD Cup 1999 dataset, which is incompatible with real-world attacks and does not reflect the current cyber environment.

To address this issue, we use the ISCX-IDS2012 [38] and CIC-IDS2017 [39] benchmark datasets. These datasets were collected over nearly a decade and include traffic composition and intrusions. They are modifiable, scalable, and reproducible, making them suitable for our purposes. The ISCX-IDS2012 dataset consists of over 100GiB of labeled network traces, including full packet payloads

in PCAP format, and contains DoS, Botnet, Brute force, Infiltration, and normal activities. The CIC-IDS2017 dataset, released by the Canadian Institute for Cybersecurity in late 2017, resembles true real-world data, and includes the most common attacks based on the 2016 McAfee report (DoS, DDoS, Web attack, Brute force, Infiltration, Heart-bleed, Botnet, and Portscan). Table 2 provides a summary of the ISCX-IDS2012 and CIC-IDS2017 intrusion detection evaluation datasets, which cover 7-day and 5-day network activities, respectively.

We extracted features from the ISCX-IDS2012 and CIC-IDS2017 datasets using the NFStream tool. Flows that were idle (no packets received) for more than 15 s were removed, as were activities that lasted longer than 1800 s. We labeled each flow according to its network session seven-tuple and timestamp. The extracted flow data distribution is shown in Table 3, along with the distribution of traffic, which exhibited highly imbalanced bias in the amount of each class. For instance, in the CIC-IDS2017 dataset, the Infiltration attack class was 56032 times more prevalent than benign traffic, posing a challenge for training a model with good generalization.

Table 2. Summary of the ISCX-IDS2012 and CIC-IDS2017 datasets(PCAPs).

Dataset	Date	Description	Size(GiB)
ISCX -IDS2012	2010-06-11	Normal, hence no malicious activity	16.1
	2010-06-12	Infiltrating the network from inside and normal activity	4.22
	2010-06-13	Infiltrating the network from inside and normal activity	3.95
	2010-06-14	HTTP denial of service and normal activity	6.85
	2010-06-15	Distributed denial of service using an IRC Botnet	23.4
	2010-06-16	Normal Activity, no malicious activity	17.6
	2010-06-17	Brute force SSH and normal activity	12.3
CIC-IDS2017	2017-07-03	Normal Activity	11
	2017-07-04	Normal Activity, BForce, SFTP and SSH	11
	2017-07-05	Normal Activity, DoS and Hearbleed Attacks slowloris, Slowhttptest, Hulk and GoldenEye	13
	2017-07-06	Normal Activity, Web and Infiltration Attacks Web BForce, XSS and SQL Inject, Infiltration Dropbox Download and Cool disk	7.8
	2017-07-07	Normal Activity, DDoS LOIT, Botnet ARES, PortScan	8.3

4.2 Evaluation Metrics

We employed several evaluation metrics to assess the performance of our experimental model, including Accuracy, Precision, Recall, and F1-measure. These evaluation criteria can effectively evaluate traffic identification, detection rate,

and the accuracy of an intrusion detection system.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$F1measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

Furthermore, in order to evaluate the performance of the classification model, we included a ROC curve (receiver operating characteristic curve). This graph displays the model's TPR (True Positive Rate) versus its FPR (False Positive Rate) at different classification thresholds. Lowering the classification threshold will increase the number of items classified as positive, subsequently increasing both False Positives and True Positives. The AUC (Area Under the Curve) provides a comprehensive performance assessment across all possible classification thresholds.

Table 3. Distribution of ISCX-IDS2012 and CIC-IDS2017 datasets(Imbalanced weight are compared with Benign traffic as a benchmark).

Dataset	Type	Total	Imbalanced weight
ISCX-IDS2012	Benign	2216455	1
	Bot	38288	58
	Infiltration	12467	178
	DoS	5290	419
	Brute force	5011	442
CIC-IDS2017	Benign	1680963	1
	DoS	179346	9
	PortScan	158946	11
	DDoS	83681	20
	Patator	6953	242
	Web Attack	2005	838
	Bot	1228	1369
	Infiltration	30	56032

4.3 Implementation Details

The experiments detailed in this chapter were performed on an Intel i5-12400F CPU and accelerated using an NVIDIA 3060 GPU. We employed the Adam optimizer with a learning rate of 5e-5 and a dropout rate of 0.3. For contrastive

learning, a substantial batch size can ensure the availability of more challenging samples within each batch. This approach can improve the model's capacity to learn and detect a broader range of malicious attacks during training. Thus, for this chapter's experiments, we used a batch size of 16384. Model training was conducted over 500 rounds, with an early stopping threshold of 50 rounds. If the model's performance failed to improve on the validation set after 50 rounds, training would terminate. We randomly sampled 20% of the dataset by traffic category to evaluate the final model's performance on the test set.

4.4 Analysis and Results

We conducted a hyperparameter sweep over the range of $\tau \in \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$ and $\lambda \in \{0.03, 0.05, 0.07, 0.1, 0.3, 0.5\}$. When lambda=0, only CE is used to train the model. On the other hand, when λ is set to 1, only SCL is used to train the model in two stages. The first stage employs the SCL pre-training model, while the second stage freezes the model's encoder and trains the final classification layer.

Figure 3 displays the results, wherein we observed that the models that displayed the highest F1-measure in the experimental setting predominantly utilized the hyperparameter values of $\tau = 0.05$ and $\lambda = 0.9$.

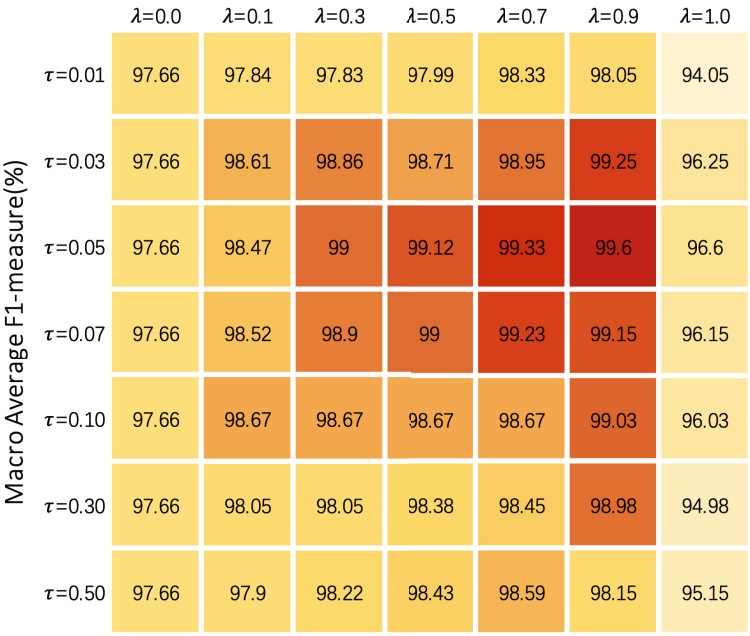

Fig. 3. The performance visualization with τ and λ combinations of strategies.

The effect of different dropout probabilities p={0.1,0.2,0.3,0.4,0.5} on the performance of the model is explored. As can be seen from Table 4, the highest F1-measure is achieved when the dropout rate is p=0.3, so the probability of the dropout layer of the fixed model in the experiment is p=0.3.

As demonstrated in Table 5, we present the outcomes obtained from the ISCX-IDS2012 dataset. Our proposed flow encoder that has undergone standard CE loss training performs comparably to several baselines. However, post-conducting training with ConFlow, we observed consistent enhancements. The bisection accuracy showed an improvement of 0.73%, and the F1-measure showed an improvement of 0.96%. Notably, in the multiclass task, the F1-measure showed a substantial improvement of 1.4%.

Table 4. The performance visualization with τ and λ combinations of strategies.

p	0.1	0.2	0.3	0.4	0.5
F1	99.42	99.54	**99.6**	99.57	99.53

Table 5. Independent test performance metrics of different method on the ISCX-IDS2012 dataset(Accuracy, Precision, and F1-measure are the macro average).

Metric	Method	Accuracy	Precision	Recall	F1-measure
Binary	Conv-LSTM [40]	95.29	97.25	97.50	97.29
	Session-Based [41]	99.48	99.39	99.39	99.39
	Metric learning [42]	99.71	99.71	99.71	99.71
	ITSN [43]	99.88	99.83	99.85	99.83
	Ours(CE)	99.78	99.11	98.81	98.96
	Ours(ConFlow)	**99.91**	**99.91**	**99.91**	**99.91**
Multiclass	DNN-FS [44]	95.20	92.86	93.17	93.17
	MFVT [?]	99.88	99.86	99.75	99.80
	Ours(CE)	99.78	98.16	98.23	98.20
	Ours(ConFlow)	**99.91**	**99.36**	**98.97**	**99.16**

As depicted in Table 6, we present the outcomes achieved through the CIC-IDS2017 dataset. Our proposed flow encoder surpasses other techniques, exhibiting consistent enhancement after training with ConFlow. We observed an improvement of 0.16% in the bisection accuracy and 0.04% in the F1-measure. Notably, in the multiclass task, the accuracy improved by 0.22%, with a substantial improvement of 1.94% in the F1-measure.

Table 7 displays the few-shot learning outcomes attained on the ISCX-IDS2012 and CIC-IDS2017 datasets, employing 10, 100, 1000, and 10000 labeled

Table 6. Independent test performance metrics of different method on the CIC-IDS2017 dataset(Accuracy, Precision, and F1-measure are the macro average).

Metric	Method	Accuracy	Precision	Recall	F1-measure
Binary	QDA [45]	96.00	96.60	93.20	94.40
	RFC [45]	99.80	99.80	99.80	99.80
	CNN-LSTM [46]	99.30	98.90	99.20	99.10
	BO-TPE-R [47]	99.99	99.00	99.00	99.00
	Ours(CE)	99.83	99.89	99.95	99.92
	Ours(ConFlow)	**99.99**	**99.96**	**99.96**	**99.96**
Multiclass	LSTM [48]	97.84	87.42	85.82	86.61
	TSVM [49]	83.25	84.15	91.75	87.78
	DBN+ANN [50]	98.97	98.07	98.42	98.24
	SS-Deep-ID [51]	99.69	92.31	96.29	94.18
	AE-CGAN-RF [17]	NaN	98.46	93.29	95.38
	DNN-FS [44]	99.73	98.05	96.68	99.58
	Ours(CE)	99.76	99.69	99.54	97.66
	Ours(ConFlow)	**99.98**	**99.61**	**99.60**	**99.60**

training examples. As the distribution of network traffic is significantly imbalanced, it is unfeasible to maintain the stratified proportion sampling according to class distribution. Therefore, we randomly selected half of the normal and attack samples from each class. ConFlow exhibited enhanced performance on the test set across all few-shot training sets. In the 10-sample training set, the accuracy rate on the ISCX-IDS2012 dataset improved by 5.47%, with the F1-measure improving by 4.18%. Similarly, in the 100-sample training set, we observed an increase of 5.47% in the accuracy rate and 4.18% in the F1-measure on the ISCX-IDS2012 dataset, whereas on the CIC-IDS2017 dataset, the accuracy rate improved by 1.13%, and the F1-measure improved by 1.26%. However, the improvement was insignificant in the 1000-sample training set, with the accuracy rate on the ISCX-IDS2012 dataset improving by only 0.2%, and the F1-measure improving by 1.19%; on the other hand, the accuracy rate on the CIC-IDS2017 dataset improved by 0.62%, with a 0.95% improvement in the F1-measure. Notably, the performance improvement over CE decreased as the number of labeled examples increased.

We assessed the generalization and robustness of the proposed method through cross-training and testing on two different data generations. Table 8 presents a comparison of the results obtained through the binary classification of CE and ConFlow. The model underwent training on the ISCX-IDS2012 training set, and subsequently, we performed training on the IDS2017 training set, following which we tested the performance on the ISCX-IDS2012 test set. ConFlow imparted improved accuracy and F1-measure values, with a respective increase of 1.31% and 6%. The results indicated that the model trained using

the ConFlow technique exhibited enhanced generalization ability and significantly improved robustness compared to CE. In addition, Fig. 4 illustrates the detection performance of the ConFlow method in real scenarios.

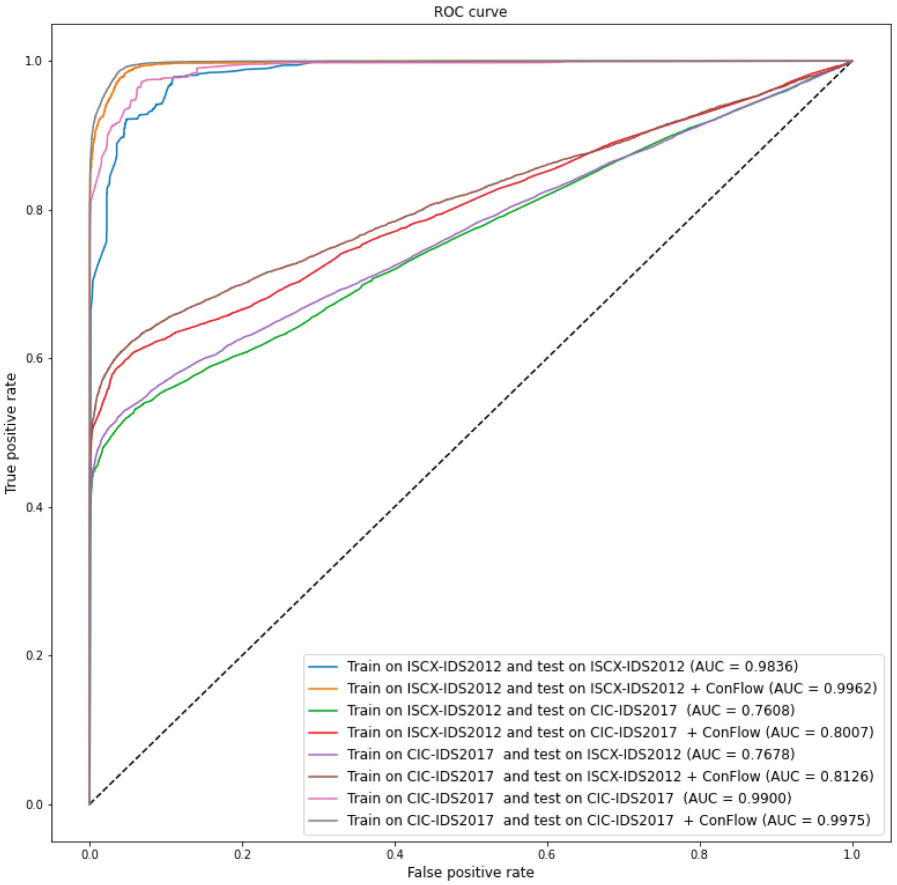

Fig. 4. Comparison of ROC curve metric results for cross-training and testing(AUC stands for Area under the ROC Curve).

Figure 5 portrays the loss variation observed during training with CE loss and ConFlow techniques. The figure indicates that training with CE loss transpires an early stop when the training is close to 200 times, and the loss curve is smoother than CE loss during the ConFlow training of 500 times. There was no major fluctuation observed in the curve, and adequate room for further decline persisted. The figure provides an explanation for how the ConFlow method effectively enhances performance. During training, the employment of contrasting network flow can bring similar samples closer and push different samples further

Table 7. Independent test performance metrics of different method on the CIC-IDS2017 dataset(Accuracy, Precision, and F1-measure are the macro average).

Dataset	N	Accuracy	Precision	Recall	F1-measure
ISCX-IDS2012	10+CE	87.17	53.15	82.27	64.58
	10+ConFlow	**92.64**	**63.13**	**94.83**	**68.76**
	100+CE	93.70	64.48	94.01	70.59
	100+ConFlow	**97.35**	**75.09**	**97.30**	**82.45**
	1000+CE	98.44	81.69	98.55	88.20
	1000+ConFlow	**98.64**	**83.31**	**98.55**	**89.39**
	full trainset+CE	99.78	97.96	97.83	97.90
	full trainset+ConFlow	**99.91**	**99.36**	**98.97**	**99.16**
CIC-IDS2017	10 CE	95.12	92.74	93.41	93.07
	10+ConFlow	**96.36**	**94.03**	**94.91**	**94.46**
	100+CE	97.01	96.30	95.36	95.83
	100+ConFlow	**98.14**	**97.79**	**96.43**	**97.09**
	1000+CE	99.15	98.72	98.68	98.70
	1000+ConFlow	**99.77**	**99.72**	**99.58**	**99.65**
	full trainset+CE	99.83	99.89	99.95	99.92
	full trainset+ConFlow	**99.99**	**99.96**	**99.96**	**99.96**

away. In comparison with the standard CE loss, this methodology can mine the similarity between hard samples hidden in vast amounts of redundant traffic, without the need for additional data augmentation techniques. Employing the two-stage training approach of contrastive learning pre-training and fine-tuning can enhance class-imbalanced learning concerning network intrusion detection.

In addition, we evaluated the real-time detection for network traffic, and the training throughput could reach up to 96,600 pieces/second during the model training process. The traffic throughput could reach 285,181 pieces/second during the detection model testing process. Therefore, the implementation of the ConFlow approach to detect large-scale network traffic in real-time with high accuracy has significant practical significance.

Comparing the performance of the standard CE loss function trained bidirectional flow encoder designed in this paper evaluation test with some current works, we observed close results, whereas ConFlow demonstrated a significant improvement. Notably, this methodology showed substantial performance improvements during few-sample learning. Finally, we verified the generalization and robustness of ConFlow through cross-training and testing on data from two different generations, thereby effectively reflecting its performance in real network environments. Building an intrusion detection system with high accuracy and low false-positive rates would have far-reaching practical significance.

Table 8. Cross training and test on different datasets.

Test set	Training set	Loss func	Acc	Pre	Recall	F1
ISCX-IDS2012	CIC-IDS2017	CE	82.26	72.93	66.18	68.33
		ConFlow	**85.82**	**81.59**	**68.25**	**73.39**
	ISCX-IDS2012	CE	99.78	97.96	97.83	97.90
		ConFlow	**99.91**	**99.36**	**98.97**	**99.16**
CIC-IDS2017	ISCX-IDS2012	CE	96.93	69.83	60.52	63.50
		ConFlow	**98.24**	**75.83**	**64.27**	**69.50**
	CIC-IDS2017	CE	99.83	99.89	99.95	99.92
		ConFlow	**99.99**	**99.96**	**99.96**	**99.96**

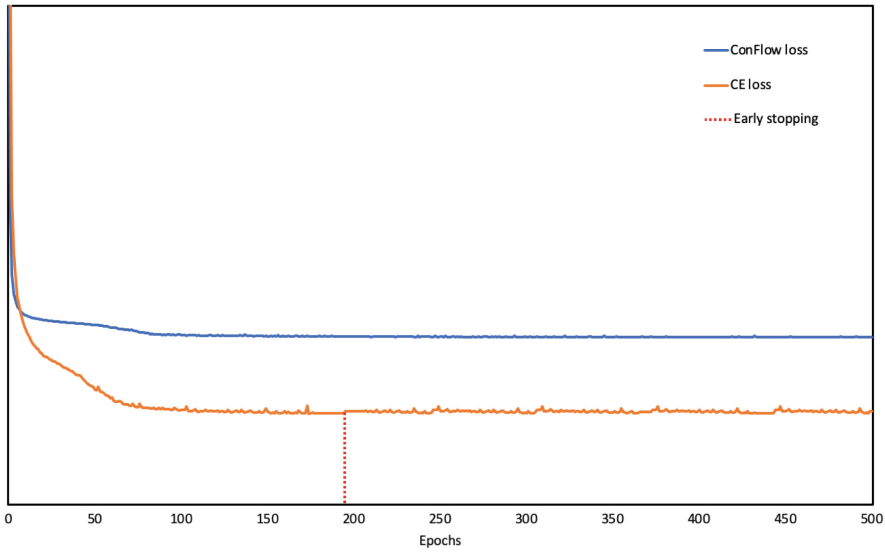

Fig. 5. Loss change curve in different loss function in training.

5 Conclusion

In real network environments, there exists vast amounts of traffic data, with very little annotated attack activity data, making it challenging to improve class-imbalanced learning in network intrusion detection. To overcome this challenge, we proposed the ConFlow method that contrasts network flow, which obtains different feature vectors using the dropout layer's randomness and the same flow's inputting into the model twice, combined with supervised contrastive learning for model training. Unlike conventional approaches, ConFlow doesn't require the two-stage pre-training and fine-tuning processes, allowing it to mine malicious attacks hidden under a large volume of normal traffic during a single train-

ing session. It outperforms state-of-the-art baselines on the ISCX-IDS2012 and CIC-IDS2017 datasets, and its performance in few-shot learning and robust tests reflects its efficacy in real network environments.

Over the past two years, self-supervised learning has emerged as an excellent solution in computer vision and natural language processing, as it doesn't depend on sample labels and can automatically mine feature information from large-scale data. Contrastive learning, which is usually used in data augmentation and other means to construct positive and negative pairs for pre-training, may face challenges in network intrusion detection as positive samples obtained using the dropout layer for data augmentation might be very similar, resulting in feature suppression. Consequently, the model might not be able to distinguish between sample similarity and class similarity, resulting in biases toward having large numbers of normal traffic samples without considering their actual class differences. As a result, we plan to investigate automated feature extraction and data augmentation techniques for network traffic, extending our supervised contrastive learning objective to self-supervised learning, enabling better class-imbalanced learning with fewer labels and label-independent initial feature information. When performing self-supervised learning, an excellent initialization could benefit the model from the pre-training task, eventually learning a more comprehensive representation of network flows.

Data Availability. The ISCX-IDS2012 and CIC-IDS2017 datasets used and analysed during this study are available in the Canadian Institute for Cybersecurity datasets repository at https://www.unb.ca/cic/datasets/index.html. The reference code for this study has been made available at https://github.com/AshinWang/ConFlow.

References

1. Scarfone, K., Mell, P., et al.: Guide to intrusion detection and prevention systems (IDPS). NIST Spec. Publ. **800**, 94 (2007)
2. Molina-Coronado, B., Mori, U., Mendiburu, A., Miguel-Alonso, J.: Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process. IEEE Trans. Netw. Serv. Manage. **17**, 2451–2479 (2020)
3. Liu, X., et al.: Self-supervised learning: generative or contrastive. IEEE Trans. Knowl. Data Eng. **35**(1), 857–876 (2021)
4. Khosla, P., et al.: Supervised contrastive learning. Adv. Neural. Inf. Process. Syst. **33**, 18661–18673 (2020)
5. Panda, M., Patra, M.R.: Network intrusion detection using naive bayes. Int. J. Comput. Sci. Netw. Secur. **7**, 258–263 (2007)
6. Ariu, D., Tronci, R. & Giacinto, G.: HMMPayl: an intrusion detection system based on Hidden Markov Models. Comput. Secur. **30**, 221–241 (2011). Publisher: Elsevier
7. Seo, S., Park, S., Kim, J.: Improvement of network intrusion detection accuracy by using restricted Boltzmann machine. In: 2016 8th International Conference on Computational Intelligence and Communication Networks (CICN), pp. 413–417. IEEE (2016)

8. Shapoorifard, H., Shamsinejad, P.: Intrusion detection using a novel hybrid method incorporating an improved KNN. Int. J. Comput. Appl. **173**, 5–9 (2017)
9. Mulay, S.A., Devale, P., Garje, G.: Intrusion detection system using support vector machine and decision tree. Int. J. Comput. Appl. **3**, 40–43 (2010)
10. Sapre, S., Ahmadi, P., Islam, K.: A robust comparison of the KDDCup99 and NSL-KDD IoT network intrusion detection datasets through various machine learning algorithms. arXiv preprint arXiv:1912.13204 (2019)
11. Jianliang, M., Haikun, S., Ling, B.: The application on intrusion detection based on k-means cluster algorithm. In: 2009 International Forum on Information Technology and Applications, vol. 1, pp. 150–152. IEEE (2009)
12. Mazarbhuiya, F.A., AlZahrani, M.Y., Georgieva, L.: Anomaly detection using agglomerative hierarchical clustering algorithm. In: International Conference on Information Science and Applications, pp. 475–484. Springer (2018)
13. Zhang, H., Huang, L., Wu, C.Q., Li, Z.: An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset. Comput. Netw. **177**, 107315 (2020)
14. Hadri, A., Chougdali, K., Touahni, R.: A network intrusion detection based on improved nonlinear fuzzy robust PCA. In: 2018 IEEE 5th International Congress on Information Science and Technology (CiSt), pp. 636–641. IEEE (2018)
15. Aldweesh, A., Derhab, A., Emam, A.Z.: Deep learning approaches for anomaly-based intrusion detection systems: a survey, taxonomy, and open issues. Knowl. -Based Syst. **189**, 105124 (2020). Publisher: Elsevier
16. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A.: Application of deep reinforcement learning to intrusion detection for supervised problems. Expert Syst. Appl. **141**, 112963 (2020). Publisher: Elsevier
17. Lee, J., Park, K.: AE-CGAN model based high performance network intrusion detection system. Appl. Sci. **9**, 4221 (2019)
18. Kozik, R., Pawlicki, M., Choraś, M.: A new method of hybrid time window embedding with transformer-based traffic data classification in IoT-networked environment. Pattern Anal. Appl. **24**, 1441–1449 (2021). Publisher: Springer
19. Wang, Z., Li, Z., Wang, J., Li, D.: Network intrusion detection model based on improved BYOL self-supervised learning. Secur. Commun. Netw. **2021** (2021)
20. Liu, J., Gao, Y., Hu, F.: A fast network intrusion detection system using adaptive synthetic oversampling and lightGBM. Comput. Secur. **106**, 102289 (2021)
21. Zuech, R., Hancock, J., Khoshgoftaar, T.M.: Detecting web attacks using random undersampling and ensemble learners. J. Big Data **8**(1), 1–20 (2021). https://doi.org/10.1186/s40537-021-00460-8
22. Bagui, S., Li, K.: Resampling imbalanced data for network intrusion detection datasets. J. Big Data **8**(1), 1–41 (2021). https://doi.org/10.1186/s40537-020-00390-x
23. Yulianto, A., Sukarno, P., Suwastika, N.A.: Improving AdaBoost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset. J. Phys. : Conf. Ser. **1192**, 012018 (2019). IOP Publishing (2019)
24. Ma, X., Shi, W.: AESMOTE: adversarial reinforcement learning with smote for anomaly detection. IEEE Trans. Netw. Sci. Eng. **8**, 943–956 (2021). https://doi.org/10.1109/TNSE.2020.3004312
25. Lin, Z., Shi, Y., Xue, Z.: IDSGAN: generative adversarial networks for attack generation against intrusion detection. arXiv preprint arXiv:1809.02077 (2018)
26. Ding, H., Chen, L., Dong, L., Fu, Z., Cui, X.: Imbalanced data classification: a KNN and generative adversarial networks-based hybrid approach for intrusion detection. Future Gener. Comput. Syst. **131**, 240–254 (2022)

27. Mulyanto, M., Faisal, M., Prakosa, S.W., Leu, J.-S.: Effectiveness of focal loss for minority classification in network intrusion detection systems. Symmetry **13**, 4 (2021)
28. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A Simple Framework for Contrastive Learning of Visual Representations. arXiv preprint arXiv:2002.05709 (2020)
29. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9729–9738 (2020)
30. Caron, M., et al.: Unsupervised learning of visual features by contrasting cluster assignments. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS) (2020)
31. Gunel, B., Du, J., Conneau, A., Stoyanov, V.: Supervised contrastive learning for pre-trained language model fine-tuning. arXiv preprint arXiv:2011.01403 (2020)
32. Yan, Y., et al.: ConSERT: a contrastive framework for self-supervised sentence representation transfer. arXiv preprint arXiv:2105.11741 (2021)
33. Gao, T., Yao, X., Chen, D.: SimCSE: simple contrastive learning of sentence embeddings. arXiv preprint arXiv:2104.08821 (2021)
34. Lopez-Martin, M., Sanchez-Esguevillas, A., Arribas, J.I., Carro, B.: Supervised contrastive learning over prototype-label embeddings for network intrusion detection. Inf. Fusion **79**, 200–228 (2022)
35. Aouini, Z., Pekar, A.: NFStream: a flexible network data analysis framework. Comput. Netw. **204**, 108719 (2022). https://doi.org/10.1016/j.comnet.2021.108719
36. He, F., Liu, T., Tao, D.: Why resNet works? Residuals generalize. IEEE Trans. Neural Netw. Learn. Syst. **31**, 5349–5362 (2020)
37. Wu, L., et al.: R-drop: regularized dropout for neural networks. Adv. Neural Inf. Process. Syst.**34** (2021)
38. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. **31**, 357–374 (2012)
39. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp **1**, 108–116 (2018)
40. Khan, M.A., Karim, M., Kim, Y., et al.: A scalable and hybrid intrusion detection system based on the convolutional-LSTM network. Symmetry **11**, 583 (2019)
41. Yu, Y., Long, J., Cai, Z.: Session-based network intrusion detection using a deep learning architecture. In: International Conference on Modeling Decisions for Artificial Intelligence, pp. 144–155. Springer (2017)
42. Chen, M., et al.: A network traffic classification model based on metric learning. CMC-Comput. Mater. Continua **64**, 941–959 (2020)
43. Li, M., Han, D., Yin, X., Liu, H., Li, D.: Design and implementation of an anomaly network traffic detection model integrating temporal and spatial features. Secur. Commun. Netw. **2021**, 7045823 (2021)
44. Siddiqi, M.A., Pak, W.: Optimizing filter-based feature selection method flow for intrusion detection system. Electronics **9**, 2114 (2020)
45. Bulavas, V., Marcinkevičius, V., Rumiński, J.: Study of multi-class classification algorithms' performance on highly imbalanced network intrusion datasets. Informatica **32**, 441–475 (2021)
46. Elmrabit, N., Zhou, F., Li, F., Zhou, H.: Evaluation of machine learning algorithms for anomaly detection. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), pp. 1–8. IEEE (2020)

47. Injadat, M., Moubayed, A., Nassif, A.B., Shami, A.: Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans. Netw. Serv. Manage. **18**, 1803–1816 (2020)
48. Di Mauro, M., Galatro, G., Liotta, A.: Experimental review of neural-based approaches for network intrusion management. IEEE Trans. Netw. Serv. Manage. **17**, 2480–2495 (2020)
49. Wang, X., Wen, J., Alam, S., Jiang, Z., Wu, Y.: Semi-supervised learning combining transductive support vector machine with active learning. Neurocomputing **173**, 1288–1298 (2016)
50. Gamage, S., Samarabandu, J.: Deep learning methods in network intrusion detection: a survey and an objective comparison. J. Netw. Comput. Appl. **169**, 102767 (2020)
51. Abdel-Basset, M., Hawash, H., Chakrabortty, R.K., Ryan, M.J.: Semi-supervised spatiotemporal deep learning for intrusions detection in IoT networks. IEEE Internet Things J. **8**, 12251–12265 (2021)