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

Lan Liu
Guangdong Polytechnic Normal University
Pengcheng Wang
Guangdong Polytechnic Normal University
Jianliang Ruan (ruanjianliang@gpnu.edu.cn)
Guangdong Polytechnic Normal University
Jun Lin
China Electronic Product Reliability and Environmental Testing Research Institute

Article

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

Posted Date: April 28th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1572776/v1

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

Lan Liu[1†], Pengcheng Wang[1†], Jianliang Ruan[1*] and Jun Lin[2]

[1*]Guangdong Polytechnic Normal University, Guangzhou 510655, China.

[2*]China Electronic Product Reliability and Environmental Testing Research Institute, Guangzhou 510610, China.

*Corresponding author(s). E-mail(s): ruanjianliang@gpnu.edu.cn;
Contributing authors: liulan@gpnu.edu.cn;
ashinwang008@gmail.com; linjun@ceprei.com;
†These authors contributed equally to this work.

Abstract

In today's cyberspace, network traffic is more massive, complex, and multi-dimensional than ever before. In order to capture malicious network attacks, a machine learning-based network intrusion detection system (NIDS) has become the mainstream method. However, there are still high false-positive and false-negative rates, which cannot guarantee detection accuracy. On the one hand, normal behaviour dominates the Internet, and network traffic presents uneven distribution. On the other hand, the goal of machine learning algorithms is usually to obtain the highest overall accuracy without considering class-imbalanced. It is difficult for the model to learn good performance from a few attack examples. Training the model with an imbalanced data distribution often leads to severe overfitting and severely damages the model's generalization ability. To improve class-imbalanced learning in network intrusion detection, it is necessary to capture the similarities between samples in different classes and compare them with samples in other classes. Based on this, we propose ConFlow, a supervised contrastive learning method for network intrusion detection. First, we design a feature extraction encoder for bidirectional network flow, and add GELU, LayerNorm, and Skip-connection units to the MLP framework, which can enhance the representation ability of the neural network. Then, we use the dropout layer's randomness in the encoder for data augmentation, and different vector representations can be obtained by feeding the same network flow into the encoder twice. Lastly, through the weighted supervised contrastive loss and cross-entropy loss in the training phase. The ConFlow

method can improve class-imbalanced learning and does not need the two stages of pre-training and fine-tuning, which can further mine maliciously attacks hidden under normal traffic. The experimental results on the ISCX-IDS2012 and CSE-CIC-IDS2017 datasets show that the ConFlow outperforms other works, and the performance improvement on few-shot learning and robustness test is more significant. The reference PyTorch code is released at https://github.com/AshinWang/ConFlow.

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

1 Introduction

In recent years, the threats that exist in cyberspace are constantly expanding, and many mechanisms such as firewalls, antivirus, antimalware and spam filters are used as security tools to protect the network. Network intrusion detection system is a critical security wall. By monitoring the status of network traffic, can find abnormal traffic and attack behavior in time, and improve the security and reliability of the network [39]. It protects network resources such as computers, servers, data and other resources from network attacks, destruction and unauthorized access or modification. Therefore, it ensures the availability, confidentiality and integrity of resources. Network intrusion detection can be an expert system based on signature matching, matching suspicious traffic with a predefined signature library of known attacks. It can also be based on anomaly detection, which measures the difference between observed traffic and trusted baseline traffic.

The existing NIDS methods are based on traditional machine learning methods that detect abnormal traffic through classification, such as support vector machine (SVM), decision tree (DT), random forest (RF), and naive Bayes. However, the performance of these machine learning methods depends to some extent on the quality of feature extraction algorithms. In the face of increasingly complex network traffic, the performance of traditional machine learning methods is degraded.

Deep learning is an essential branch of machine learning. It mines the latent features of high-dimensional data by training models and transforms the problem of network traffic anomaly detection into a classification problem. Through a large number of data sample training, the difference between normal behavior and abnormal behavior is adaptively learned, which effectively enhances the accuracy of intrusion processing. Some scholars also began to introduce deep learning methods into the field of intrusion detection. Intrusion detection methods based on deep learning include multiLayer perceptron (MLP), recurrent neural network (RNN), convolutional neural network (CNN). These deep learning algorithms can effectively improve the performance of intrusion

detection system. However, in the real network world, normal activities dominate, so a small number of malicious attacks can be hidden in a large amount of normal netflow, resulting in a highly imbalanced distribution of netflow and a more complex classification model, the model are biased towards the majority class. However, in the real network world, normal activities dominate, so a small number of malicious attacks can be hidden in a large amount of normal traffic, resulting in a highly imbalanced distribution of traffic and a more complex classification model, and biased towards normal traffic. Although scholars have conducted extensive research on intrusion detection and achieved good progress, the class-imbalanced is still an important factor limiting the performance of intrusion detection [34].

In the past few years, supervised contrastive learning has achieved good results in computer vision (CV), natural language processing (NLP) representation learning [29]. Khosla et al. proposed a supervised contrastive loss that outperforms cross-entropy loss and can effectively utilize label information to cluster point clusters belonging to the same class together in the embedding space while separating samples from different classes [22]. The core idea of classification is to divide samples of different categories, usually using cross-entropy as the loss function. Gunel et al. propose a new method for the fine-tuning stage of natural language processing. In the fine-tuning stage of the BERT model, weighting combines supervised contrastive loss and cross-entropy loss [15].

Inspired by the success of supervised contrastive learning in CV and NLP, we use it to compare network flows. It uses sample labels to create new sample pairs to draw samples of the same class closer and away from samples of the same class. We hope that the network intrusion detection model has strong robustness and generalization, which can deal with the class-imbalanced problem in the intrusion detection data. We adopts supervised contrastive learning to compare network flows. By directly weighting supervised contrastive loss and cross-entropy loss to train the model. In ISCX-IDS2012 and CIC-IDS2017 datasets outperform state-of-the-art baselines. Our contributions to this work are listed in the following:

(1) We design a feature extraction encoder for bidirectional network flow and add GELU, LayerNorm, and Skip-connection units to the MLP framework, which can enhance the representation ability of the neural network. The model can learn automatically represent learning network flow data and reduce reliance on feature engineering.

(2) We demonstrate that our method is more generalization ability than other methods. The ConFlow method can improve class-imbalanced learning and does not need the two stages of pre-training and fine-tuning. Use the dropout layer's randomness in the encoder for data augmentation and the weighted supervised contrastive loss and cross-entropy loss in the training phase, which can further mine maliciously attacks hidden under normal traffic.

(3) We explore learning performance on few-shot network traffic data, which achieves excellent improvements even on few labeled data (10, 100,

1000 labelled samples) performance. We also perform cross-testing on the two datasets to verify the generalization and robustness of the ConFlow.

The rest of the paper is organized as follows: Section 2 introduces the state-of-the-art advanced network intrusion detection and summarizes related methods for dealing with imbalanced data. Section 3 details our proposed method. Section 4 compares and validates the related algorithms and analyzes and discusses the results. Section 5 is the conclusion part, which summarizes the research method proposed in this paper and discusses 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 development of malware, anomaly-based intrusion detection systems have been introduced to detect unknown malware attacks. The use of machine learning and deep learning to create detection models has become the mainstream of research. It can identify variant and unknown threats, which makes up for the deficiency that traditional feature detection and behaviour detection can only detect known attacks. Threat detection technology puts forward higher requirements.

Machine learning-based network intrusion detection can be divided into supervised and unsupervised learning. In supervised learning, it can be divided into generative and discriminative according to the modeling method. The core idea of the generation algorithm is to learn the distribution of network traffic data from a statistical point of view, reflecting the similarity of the same type of traffic data, such as the Bayesian algorithm [37] and Hidden Markov Model (HMM) [4], Restricted Boltzmann Machine (RBM) [40]. The discriminant method can reflect the difference between different types of traffic data by learning decision function or conditional probability distribution as a prediction model, such as the K nearest neighbour algorithm (KNN) [41], decision tree (DT) and Support vector machine (SVM) [35], logistic regression (LR) [38] and other methods. Unsupervised learning can train models through 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 studied in intrusion detection [20]. Hierarchical clustering can discover the hierarchical relationship of classes by calculating the distance between samples and is often used for anomaly mining in redundant network traffic [33]. Gaussian Mixture Model (GMM) [51] can learn the probability density function and can identify different types of network traffic with similar distributions; Principal Component Analysis (PCA) [16] is the most commonly used dimensionality reduction method, which can reduce the feature dimension

of network traffic, so it is used in a large number of feature dimension reduction works.

Deep learning has been widely used in intrusion detection models in the past decade. These methods can obtain good prediction results by learning the practical features in the data. NIDS based on deep learning mainly learns the latent features of high-dimensional data by training models and is also divided into supervised and unsupervised learning. Supervised learning includes auto-encoder (AE), deep belief network (DBN), recurrent neural network (RNN), convolutional neural networks (CNN), mainly by converting network traffic feature data into one-dimensional sequences, text or images, using neural networks for classification [2]. Other scholars use reinforcement learning to sample the training data set and motivate the detection results [30], use generative adversarial networks (GAN) to generate few-class attacks [24], and use Transformer to build a more complex classification model [23]. Unsupervised learning is mainly self-supervised Learning (SSL), which creates supervised information through data augmentation, pre-trains large-scale unlabeled network flow, which obtains the features of different network traffic [46].

2.2 Imbalanced Learning

The model learns better on samples of majority classes but generalizes poorly on samples of minor classes in class-imbalanced data. For the network imbalance learning problem, many scholars have proposed some solutions. We summarize as follows:

(1) Resampling: Liu et al. proposed oversampling for minority(usually malicious) samples in network traffic [28], and Zuech et al. proposed undersampling for many samples [52]. Over-sampling can easily lead to over-fitting of few-class samples. It is impossible to learn more robust and generalizable features and often perform worse on very imbalanced data. At the same time, under-sampling will cause serious information loss of many types, resulting in under-sampling. Fitting occurs, so Bagui et al. combined the two to balance the data type distribution [5].

(2) Data synthesis: generating "new" data similar to minority samples. The classical method SMOTE and its improvement [32, 50, 51], select the few-class samples arbitrarily, use K nearest neighbors to select similar samples, and obtain new samples by linear interpolation of the samples. Other scholars have used a generative adversarial network framework to generate adversarial few-class malicious traffic to improve the robustness of the model [11, 27].

(3) Reweighting: assign different weights to categories of different network traffic. Because the network traffic distribution is very skewed, the simplest weighting according to the inverse of the number of categories is usually challenging to improve the model's learning of imbalanced networks. Therefore, some scholars dynamically weight according to the number of "effective" samples which optimizes the loss weighting of the classification spacing according to the number of samples [36].

2.3 Contrastive Learning

Contrastive learning (CL) constructs positive and negative samples pairs to extract information from the data itself. The core idea of contrastive learning is to construct a contrastive loss function, and the model can pull in on similar samples, pull out dissimilar samples, and compare similar and dissimilar data.

Contrastive learning has been popular in CV in the last years. Chen et al proposed SimCLR [9], and the model trained with this framework set off a craze for contrastive learning with a 7% improvement on ImageNet. It obtains two enhanced images through data enhancement, inputs them to the image encoder to obtain the representation of the image, then maps the image representation to one dimension through the projection head, and finally calculates the distance between the sample representations through InfoNCE, which is closer to the positive sample, and pull away from negative samples. Since contrastive learning requires a large batch size to ensure the contrast effect, the training is very time-consuming and computationally resource-intensive. In order to break through the limitation of GPU memory, He et al. [18] introduced MoCo (Momentum Contrast), which proposed using a memory bank to store the representation of previously processed samples. Caron et al. [7] introduced a new method SwAV, and the main idea is to cluster various types of samples and then distinguish the clusters of each class. Khosla et al. [22] proposed a supervised contrastive loss that outperforms cross-entropy loss and can effectively utilize label information to cluster together point clusters belonging to the same class in the embedding space while separating points from different classes sample.

There has also been some follow-up work recently in NLP. Gunel et al [15] proposed a new approach to the fine-tuning stage of natural language processing. They combine supervised contrastive loss and cross-entropy loss with weighting in the fine-tuning stage of the BERT model. Yan et al. [48] proposed ConSERT, a contrastive framework for self-supervised sentence representation transfer, which employs contrastive learning to fine-tune BERT in an unsupervised and efficient manner. By introducing various data enhancement methods, ConSERT solves the collapse problem of BERT sentence representation, which is 8% higher than the previous state-of-the-art on the STS dataset, which is comparable to the supervised learning SBERT-NLI method. Gao et al. [14] proposed SimCSE, which uses simple contrastive learning to do sentence embedding, and uses dropout layer's ramdomness for data augmentation to generate better-quality sentence vectors without supervising the data.

Some scholars have also introduced contrastive learning to improve model performance in intrusion detection. Wang et al. [46] proposed a new data augmentation strategy for intrusion detection data and an intrusion detection model based on unlabeled self-supervised learning, using the new data augmentation strategy to introduce a perturbation augmentation model to learn invariant feature representation capabilities and improve BYOL self-supervision. The learning method is trained on the unlabeled UNSW-NB15 intrusion detection dataset to extract network traffic feature representations.

Linear evaluation on UNSW-NB15 and transfer learning on NSL-KDD, KDD-CUP99 and CIDDS-001 achieve excellent performance on all metrics. Lopez et al. [31] propose to include labels in the same embedding space as features and perform distance comparisons between features and labels in this shared embedding space, separating label prototypes while minimizing the distance between each prototype and its peers. The proposed scheme dramatically reduces the number of pairwise comparisons, thereby improving model performance.

3 Method

In this section, we propose an encoder architecture and training method for network flow, which improves the 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 make statistics on the behavioral feature and do not need to investigate the information of the data packets in depth. The division of network flow is determined by the network session seven-tuple (source IP address, destination IP address, protocol, source port, destination port, VLAN identifier, tunnel identifier).

We use the NFStream [3] tool to parse network packets (PCAP), set maximum time thresholds such as active timeout and inactive timeout, and perform 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, which retain a total of 58 features, as shown in Table 1.

3.2 Encoder network

Popularly big and complex network structures such as CNN and Transformer have achieved good performance in computer vision and natural language processing tasks. However, the neural network don't need use convolution and attention structures for relatively simple network traffic to representation learning. We use MLP and skip-connection as the main architecture, the representation encoder network for network traffic is shown in Figure 1.

In the proposed encoder for network flow feature extraction, categorical features in the input are coded by one-hot encoding and dropout layers, while LayerNorm layers normalize continuous features. When inputting to the next neural unit, they are stitched together to get better information interaction between categorical and continuous features. The main structure of the network includes two DRB (Dense Resnet Block) modules, which can effectively reduce the gradient disappearance and network degradation problems mainly through the Skip-connection mechanism, making training easier. After the DRB module it also contains an MLP unit consisting of linear layers, GELU,

Table 1 Extract features of network flow by NFStream tool.

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
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

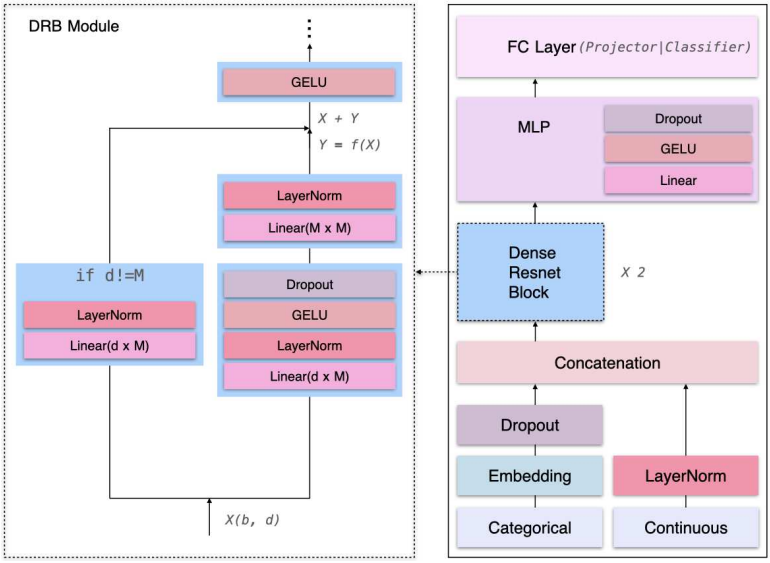

Fig. 1 Encoder architectures for network flow.

and Dropout components, whose role is to output the representation information extracted by the encoder to a fixed dimension. Finally, through a fully connected layer (Fully Connected, FC), the network flow vector obtained

by the encoder is mapped to the classification category dimension or the projection dimension on the unit sphere.

The DRB module also includes components such as linear layer, LayerNorm, GELU, and Dropout, which are used to improve the expressiveness of the network flow embeddings. Using LayerNorm in the encoder can obtain different network features without relying on other data, thus avoiding the problem of mini-batch data distribution in BatchNorm. In addition, some studies found that BatchNorm can have some negative effects on network performance, especially leading to information leakage in contrastive learning[17]. GELU (Gaussian Error Linear Unit) is smoother than ReLU and is used by NLP models such as BERT, GPT-3 and ViT. This chapter also uses the GELU activation function to replace the commonly used ReLU.

3.3 Training framework and loss function

It is difficult to manually construct data augmentation for network traffic that guarantees semantic invariance. Dropout is usually used to prevent model overfitting in deep learning. SimCSE [14] and R-Drop [47] take advantage of the randomness of Dropout to rapidly expand data and achieve good results, and the effectiveness of this method is demonstrated on several datasets in different domains. Therefore, we use "Dropout twice" to construct positive samples for supervised contrastive learning, that is, a network traffic sample passes through the model twice to obtain different feature vectors of the same input.

Due to using the Dropout for data augmentation, the obtained positive samples are similar, resulting in feature suppression in the network flow. The deep learning model cannot distinguish between sample similarity and class similarity and more Bias to have a large number of normal traffic samples without considering the actual class differences between them.

Inspired by the success of supervised contrastive learning in CV and NLP, we use it to compare network flows. It uses sample labels to create new sample pairs to draw samples of the same class closer and away from samples of the same class. A similarity matrix with the size of the input can be obtained by inputting the model twice to obtain different vector representations of the same input and calculating the similarity between them. In this similarity matrix, designing a supervised contrast loss, the larger the sum of the similarity distances of the same type of traffic, the better, and the smaller the sum of the similarity distances of different classes, the better.

The ConFlow's architecture is shown in Figure 2. The solid line part represents the first input and output pipeline of network flow, and the solid line part represents the second time. The role of the encoder network is to map the traffic to the representation space. Each flow is input to the same encoder twice. A pair of representation vectors is obtained, and finally, a 512-dimensional representation vector is obtained by using a multilayer perceptron with only one hidden layer. The role of the projection network is to map the representation vector into a final vector for loss calculation. The projection uses a fully

connected layer, and the dimension is 64. the unit hypersphere is used for regularization to maximize the similarity between augmentation flows projections and minimize the similarity between different flows. The classification network, which maps the dimensions to the label category dimension through a linear layer.

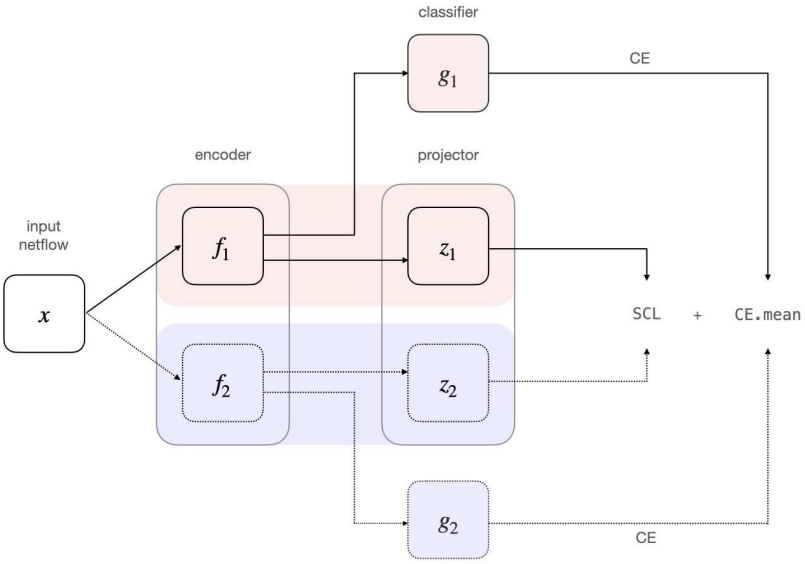

Fig. 2 ConFlow's architecture.

ConFlow introduces supervised contrast loss into the standard training method to mine malicious attacks hidden under a large amount of normal traffic. By inputting the model "twice", the weighting of the supervised contrast loss and the average loss of the two cross entropies is used as the final loss. Additional data augmentation methods and two-stage training of traditional contrastive learning pre-training plus fine-tuning are required.

We weight cross-entory (CE) loss with the supervised contrastive loss (SCL). The formula is as follows:

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

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

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

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

Due to the rarity and secrecy of network intrusion attacks, it is challenging to obtain datasets that reflect current real-world cyberspace. Many studies are still using network traffic data collected initially, even on the KDD Cup1999. However, those attacks are outdated, incompatible with real-world attacks, and do not reflect the current cyber environment.

We use ISCX-IDS2012 [43] and CIC-IDS2017 [42] as benchmark datasets, they were collected over nearly a decade, including traffic composition and intrusions, and are modifiable, scalable, and reproducible. ISCX-IDS2012 dataset consists of labeled network traces, including full packet payloads in PCAP over 100GiB, including DoS, Botnet, Brute force, Infiltration and normal activity. CIC-IDS2017 was released in late 2017 via Canadian Institute for Cybersecurity(CIC), which resembles the true real-world data, which contains the most common attack based on the 2016 McAfee report(DoS, DDoS, Web attack, Brute force, Infiltration, Heart-bleed, Botnet and Portscan). As shown in Table 2, the ISCX-IDS2012 and CIC-IDS2017 intrusion detection evaluation datasets include the following 7-day and 5-day network activities.

We performed feature extraction on ISCX-IDS2012 and CIC-IDS2017 using the NFStream tool. Flows idle (no packets received) for more than 15 seconds will expire, and those activities for more than this 1800 seconds will expire. We label each flow according to the network session seven-tuple and timestamp. The extracted flow data distribution is shown in Table 3. We also show the distribution of traffic, with a highly imbalanced bias in the amount each class. For example, in CIC-IDS2017 dataset, the imbalance of Infiltration attacks is as high as 56032 times compared with benign traffic, which is challenging to train a model with good generalization.

4.2 Evaluation metrics

We use Accuracy, Prediction, Recall, and F1-measure to evaluate the performance of the experimental model. These evaluation criteria can reflect the performance of traffic identification, detection rate and accuracy of intrusion detection system.

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

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

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

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

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

4.3 Implementation Details

The experiments in this chapter are run on CPU i5-12400F and accelerated with GPU NVIDIA 3060. We use the Adam optimizer with a learning rate of 5e-5, and a dropout of 0.3. In contrastive learning, the batchsize can make more difficult samples meet in the same batch under a considerable size, which helps the model learn and mine more malicious attacks during training. Therefore, this chapter sets the batchsize as 16384. Set the number of training rounds to 500 and set the early stop to 50. When the model does not improve the

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

performance of the validation set in 50 pieces of training, the training will be terminated early. 20% of the data set is fixedly sampled according to the traffic category as the test set for the final model performance evaluation.

4.4 Analysis and results

We perform hyperparameter sweep for $\tau \in \{0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$ 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. When lambda=1, only SCL is used to train the model, which is divided into two stages. The first stage uses the SCL pre-training model. The second stage freezes the model's encoder and trains the final classification layer.

The results can be found in Figure 3. We can make the following observations the models with the best test F1-measure in the experimental setting overwhelmingly used the hyperparameter combination $\tau = 0.05$ and $\lambda = 0.9$.

As shown in Table 4, we report results on the ISCX-IDS2012 dataset, our proposed flow encoder is close to some baselines, with consistent improvements after training with ConFlow. The bisection accuracy is improved by 0.73%, and the F1-measure is improved by 0.96%. The improvement is even more significant in the multiclass task, with an F1-measure improvement of 1.4%.

As shown in Table 5, we report results on the CIC-IDS2017 dataset, our proposed flow encoder outperforms other methods, with consistent improvements after training with ConFlow. The bisection accuracy is improved by 0.16%, and the F1-measure is improved by 0.04%. The improvement is even more significant in the multi-class task, the accuracy is improved by 0.22%, with an F1-measure improvement of 1.94%.

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

Table 4 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 [21]	95.29	97.25	97.50	97.29
	Session-Based [49]	99.48	99.39	99.39	99.39
	Metric learning [8]	99.71	99.71	99.71	99.71
	ITSN [26]	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 [25]	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**

In Table 6, we report few-shot learning results on ISCX-IDS2012 and CIC-IDS2017 with 10, 100, 1000, 1000 labeled training examples. Since network traffic distribution is highly imbalanced and cannot satisfy the stratified proportion sampling according to the class distribution, half of the normal samples and half of the attack samples are randomly selected in each class. ConFlow improves performance on the test set in all few-shot training sets. In the training set of 10 samples, the accuracy rate on the ISCX-IDS2012 dataset increased by 5.47%, and the F1-measure increased by 4.18%. In the training set of 100 samples, the accuracy rate on the ISCXIDS2012 dataset is increased by 5.47%, and the F1-measure is increased by 4.18%; on the CIC-IDS2017 dataset, the accuracy rate is increased by 1.13%, and the F1-measure is increased by 1.26%.

Table 5 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 [6]	96.00	96.60	93.20	94.40
	RFC [6]	99.80	99.80	99.80	99.80
	CNN-LSTM [12]	99.30	98.90	99.20	99.10
	BO-TPE-R [19]	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 [10]	97.84	87.42	85.82	86.61
	TSVM [45]	83.25	84.15	91.75	87.78
	DBN+ANN [13]	98.97	98.07	98.42	98.24
	SS-Deep-ID [1]	99.69	92.31	96.29	94.18
	AE-CGAN-RF [24]	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**

Table 6 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**

In the training set of 1000 samples, the improvement compared to the baseline is insignificant. The accuracy rate on the ISCX-IDS2012 dataset is increased by 0.2%, and the F1 measure is increased by 1.19%; the accuracy rate on the CIC-IDS2017 dataset is increased by 0.62%, 0.95% increases the F1-measure that the performance improvement over CE decreases as the number of labeled examples increases.

5 Conclusion

To further improve class-imbalanced learning in network intrusion detection, we proposed a method that contrast network flow. It obtains different feature vectors input by the dropout layer's randomness, the same flow through inputting the model twice, and combines supervised contrastive learning and cross-entropy to train the model. ConFlow significantly improves imbalanced network traffic learning and does not need the two stages of pre-training and fine-tuning, which can further mine maliciously attacks hidden under normal traffic. Outperforming state-of-the-art baselines when evaluated on ISCX-IDS2012 and CIC-IDS2017 datasets, performing well on few-shot learning and robust tests.

However, using the dropout for data augmentation, the obtained positive samples are similar, resulting in feature suppression in the model. The model cannot distinguish between sample similarity and class similarity and more bias to have a large number of normal traffic samples without considering the actual class differences between them. Therefore, we plan to investigate automated feature extraction and data augmentation techniques for network traffic, extending our supervised contrastive learning objective to self-supervised learning, which can learn in a few-label, class-imbalanced better label-independent initial feature information. When self-supervised perform an excellent initialization, the model can benefit from the pre-training task and eventually learn a more general representation of the network flow.

Data availability

The ISCX-IDS2012 and CIC-IDS2017 datasets generated and analysed during the current study are available in the Canadian Institute for Cybersecurity datasets repository, https://www.unb.ca/cic/datasets/index.html.

References

[1] Mohamed Abdel-Basset, Hossam Hawash, Ripon K Chakrabortty, and Michael J Ryan. Semi-supervised spatiotemporal deep learning for intrusions detection in iot networks. *IEEE Internet of Things Journal*, 8(15):12251–12265, 2021.

[2] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z Emam. Deep learning approaches for anomaly-based intrusion detection systems: A survey,

taxonomy, and open issues. *Knowledge-Based Systems*, 189:105124, 2020. Publisher: Elsevier.

[3] Zied Aouini and Adrian Pekar. Nfstream: A flexible network data analysis framework. *Computer Networks*, 204:108719, 2022.

[4] Davide Ariu, Roberto Tronci, and Giorgio Giacinto. HMMPayl: An intrusion detection system based on Hidden Markov Models. *computers & security*, 30(4):221–241, 2011. Publisher: Elsevier.

[5] Sikha Bagui and Kunqi Li. Resampling imbalanced data for network intrusion detection datasets. *Journal of Big Data*, 8(1):1–41, 2021.

[6] Viktoras Bulavas, Virginijus Marcinkevičius, and Jacek Rumiński. Study of multi-class classification algorithms' performance on highly imbalanced network intrusion datasets. *Informatica*, 32(3):441–475, 2021.

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[8] Mo Chen, Xiaojuan Wang, Mingshu He, Lei Jin, Khalid Javeed, and Xiaojun Wang. A network traffic classification model based on metric learning. *CMC-computers Materials & Continua*, 64(2):941–959, 2020.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv preprint arXiv:2002.05709*, 2020.

[10] Mario Di Mauro, Giovanni Galatro, and Antonio Liotta. Experimental review of neural-based approaches for network intrusion management. *IEEE Transactions on Network and Service Management*, 17(4):2480–2495, 2020.

[11] Hongwei Ding, Leiyang Chen, Liang Dong, Zhongwang Fu, and Xiaohui Cui. Imbalanced data classification: A knn and generative adversarial networks-based hybrid approach for intrusion detection. *Future Generation Computer Systems*, 2022.

[12] Nebrase Elmrabit, Feixiang Zhou, Fengyin Li, and Huiyu Zhou. Evaluation of machine learning algorithms for anomaly detection. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–8. IEEE, 2020.

[13] Sunanda Gamage and Jagath Samarabandu. Deep learning methods in network intrusion detection: A survey and an objective comparison.

Journal of Network and Computer Applications, 169:102767, 2020.

[14] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[15] Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*, 2020.

[16] Amal Hadri, Khalid Chougdali, and Raja Touahni. A network intrusion detection based on improved nonlinear fuzzy robust PCA. In *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, pages 636–641. IEEE, 2018.

[17] Fengxiang He, Tongliang Liu, and Dacheng Tao. Why resnet works? residuals generalize. *IEEE transactions on neural networks and learning systems*, 31(12):5349–5362, 2020.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[19] MohammadNoor Injadat, Abdallah Moubayed, Ali Bou Nassif, and Abdallah Shami. Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Transactions on Network and Service Management*, 18(2):1803–1816, 2020.

[20] Meng Jianliang, Shang Haikun, and Bian Ling. The application on intrusion detection based on k-means cluster algorithm. In *2009 International Forum on Information Technology and Applications*, volume 1, pages 150–152. IEEE, 2009.

[21] Muhammad Ashfaq Khan, Md Karim, Yangwoo Kim, et al. A scalable and hybrid intrusion detection system based on the convolutional-lstm network. *Symmetry*, 11(4):583, 2019.

[22] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[23] Rafa\l Kozik, Marek Pawlicki, and Micha\l Choraś. A new method of hybrid time window embedding with transformer-based traffic data classification in IoT-networked environment. *Pattern Analysis and Applications*, 24(4):1441–1449, 2021. Publisher: Springer.

[24] JooHwa Lee and KeeHyun Park. Ae-cgan model based high performance network intrusion detection system. *Applied Sciences*, 9(20):4221, 2019.

[25] Ming Li, Dezhi Han, Dun Li, Han Liu, and Chin-Chen Chang. Mfvt: an anomaly traffic detection method merging feature fusion network and vision transformer architecture. 2021.

[26] Ming Li, Dezhi Han, Xinming Yin, Han Liu, and Dun Li. Design and implementation of an anomaly network traffic detection model integrating temporal and spatial features. *Security and Communication Networks*, 2021, 2021.

[27] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: Generative adversarial networks for attack generation against intrusion detection. *arXiv preprint arXiv:1809.02077*, 2018.

[28] Jingmei Liu, Yuanbo Gao, and Fengjie Hu. A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm. *Computers & Security*, 106:102289, 2021.

[29] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[30] Manuel Lopez-Martin, Belen Carro, and Antonio Sanchez-Esguevillas. Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Systems with Applications*, 141:112963, 2020. Publisher: Elsevier.

[31] Manuel Lopez-Martin, Antonio Sanchez-Esguevillas, Juan Ignacio Arribas, and Belen Carro. Supervised contrastive learning over prototype-label embeddings for network intrusion detection. *Information Fusion*, 79:200–228, 2022.

[32] Xiangyu Ma and Wei Shi. Aesmote: Adversarial reinforcement learning with smote for anomaly detection. *IEEE Transactions on Network Science and Engineering*, 8(2):943–956, 2021.

[33] Fokrul Alom Mazarbhuiya, Mohammed Y AlZahrani, and Lilia Georgieva. Anomaly detection using agglomerative hierarchical clustering algorithm. In *International Conference on Information Science and Applications*, pages 475–484. Springer, 2018.

[34] Borja Molina-Coronado, Usue Mori, Alexander Mendiburu, and Jose Miguel-Alonso. Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process. *IEEE Transactions on Network and Service Management*, 17(4):2451–2479,

2020.

[35] Snehal A Mulay, PR Devale, and GV Garje. Intrusion detection system using support vector machine and decision tree. *International journal of computer applications*, 3(3):40–43, 2010.

[36] Mulyanto Mulyanto, Muhamad Faisal, Setya Widyawan Prakosa, and Jenq-Shiou Leu. Effectiveness of focal loss for minority classification in network intrusion detection systems. *Symmetry*, 13(1):4, 2021.

[37] Mrutyunjaya Panda and Manas Ranjan Patra. Network intrusion detection using naive bayes. *International journal of computer science and network security*, 7(12):258–263, 2007.

[38] Suchet Sapre, Pouyan Ahmadi, and Khondkar Islam. 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.

[39] Karen Scarfone, Peter Mell, et al. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007.

[40] Sanghyun Seo, Seongchul Park, and Juntae Kim. Improvement of network intrusion detection accuracy by using restricted Boltzmann machine. In *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 413–417. IEEE, 2016.

[41] Hossein Shapoorifard and Pirooz Shamsinejad. Intrusion detection using a novel hybrid method incorporating an improved KNN. *Int. J. Comput. Appl*, 173(1):5–9, 2017.

[42] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

[43] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.

[44] Murtaza Ahmed Siddiqi and Wooguil Pak. Optimizing filter-based feature selection method flow for intrusion detection system. *Electronics*, 9(12):2114, 2020.

[45] Xibin Wang, Junhao Wen, Shafiq Alam, Zhuo Jiang, and Yingbo Wu. Semi-supervised learning combining transductive support vector machine with active learning. *Neurocomputing*, 173:1288–1298, 2016.

[46] Zhendong Wang, Zeyu Li, Junling Wang, and Dahai Li. Network intrusion detection model based on improved byol self-supervised learning. *Security and Communication Networks*, 2021, 2021.

[47] Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. R-drop: regularized dropout for neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[48] Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. Consert: A contrastive framework for self-supervised sentence representation transfer. *arXiv preprint arXiv:2105.11741*, 2021.

[49] Yang Yu, Jun Long, and Zhiping Cai. Session-based network intrusion detection using a deep learning architecture. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 144–155. Springer, 2017.

[50] Arif Yulianto, Parman Sukarno, and Novian Anggis Suwastika. Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset. In *Journal of Physics: Conference Series*, volume 1192, page 012018. IOP Publishing, 2019.

[51] Hongpo Zhang, Lulu Huang, Chase Q Wu, and Zhanbo Li. An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset. *Computer Networks*, 177:107315, 2020.

[52] Richard Zuech, John Hancock, and Taghi M Khoshgoftaar. Detecting web attacks using random undersampling and ensemble learners. *Journal of Big Data*, 8(1):1–20, 2021.