DOI:10.1587/transinf.2024EDP7001

Publicized:2024/04/25

PAPER

Unsupervised Intrusion Detection Based on Asymmetric Auto-Encoder Feature Extraction

Chunbo Liu†, Liyin Wang††, Zhikai Zhang†††, Chunmiao Xiang†††, Zhaojun Gu†, Zhi Wang††††, and Shuang Wang$^{\dagger a)}$, ***Member***

SUMMARY Aiming at the problem that large-scale traffic data lack labels and take too long for feature extraction in network intrusion detection, an unsupervised intrusion detection method ACOPOD based on Adam asymmetric autoencoder and COPOD (Copula-Based Outlier Detection) algorithm is proposed. This method uses the Adam asymmetric autoencoder with a reduced structure to extract features from the network data and reduce the data dimension. Then, based on the Copula function, the joint probability distribution of all features is represented by the edge probability of each feature, and then the outliers are detected. Experiments on the published NSL-KDD dataset with six other traditional unsupervised anomaly detection methods show that ACOPOD achieves higher precision and has obvious advantages in running speed. Experiments on the real civil aviation air traffic management network dataset further prove that the method can effectively detect intrusion behavior in the real network environment, and the results are interpretable and helpful for attack source tracing.

key words: *intrusion detection, feature extraction, network traffic, asymmetric auto-encoder, Copula function.*

1. Introduction

A growing variety of network devices and applications are being developed in the quickly evolving information society to satisfy people's demands in both their personal and professional lives. People's social activities are getting increasingly intertwined with the online world, and internet connection has become an essential component of modern life. Although the internet is convenient, it has also given bad people opportunities to make money from cybercrimes. Attacks on company databases and the dark web selling of stolen personal information are commonplace incidents. The 2010 discovery of the Stuxnet virus, which was designed to target vital infrastructure, serves as more evidence that the hazards associated with cyberattacks are already widespread.

As a preventive defensive method, intrusion detection has steadily grown in importance as a tool for maintaining network security [1]. Intrusion Detection System (IDS) is a real-time monitoring system installed in a network with the purpose of identifying intrusion activity and taking appropriate action by examining data produced by the network. The amount of network data that IDS must examine is expanding quickly due to factors including fast network transmission, the expansion of the Internet of Things (IoT), and the use of technologies like cloud computing. Many distinct protocols are used in network traffic transmission, and these protocols' field values are frequently categorical variables. The high-dimensionality and non-linearity of network traffic characteristics result from this. The detection efficiency is low and the time and computational expenses are quite expensive when directly detecting this high-dimensional data.

Numerous research studies have blended deep learning feature extraction approaches with traditional machine learning to overcome the aforementioned concerns. To reduce the dimensionality of the data, they employ deep neural networks to extract important characteristics from data distributions. The objective of this strategy is to improve the Intrusion Detection System (IDS) operating speed and the quality of input features. The accuracy of IDS directly depends on the quality of its input features. The more effectively the input features represent the overall distribution of the data, the more accurately IDS can differentiate normal behavior from intrusion behavior using these features. In contrast to traditional feature selection, feature extraction produces new features that are more condensed than the originals. Deep learning's strong hierarchical feature learning ability can better match traditional machine learning approaches, particularly when it comes to capturing nonlinear information [2]. The original data are more significantly represented by the learned characteristics, which facilitate data display and categorization.

Traditional machine learning techniques are still in high demand in the intrusion detection space. Based on clustering, the K-Means method [3] separates the data into k clusters, with the distance between each cluster's centroid and each data point within the cluster determining its score. Outliers are defined as data points that are distant from the centroid. The k closest neighbors of a data point are the focus of the K-Nearest Neighbor (KNN) method [4]. The data point is categorized as an outlier if the bulk of these neighbors have already been labeled as outliers. The density-based Local Outlier Fac-

$\dagger$The authors are with Information Security Evaluation Center, Civil Aviation University of China, China.
$\dagger\dagger$The author is with Aeronautical Information Service Center, Air Traffic Management Bureau, Civil Aviation Administration of China, China.
$\dagger\dagger\dagger$The authors are with College of Computer Science and Technology, Civil Aviation University of China, China.
$\dagger\dagger\dagger\dagger$The author is with College of Cyber Science, Nankai University, China.
a) E-mail: s-wang@cauc.edu.cn (Corresponding author)

tor (LOF) algorithm [5] aims to discover local outliers. It calculates the LOF value by comparing the local reachability density (LRD) of the present data point and its neighbors. Points with higher LOF values are considered as anomalies. However, in contrast to LOF, the density-based Connectivity-based Outlier Factor (COF) method [6] computes the density of data points differently. This is because COF computes connection distances using minimal spanning trees while accounting for the relationships between data points. A statistical method called the Histogram-based Outlier Score (HBOS) algorithm [7] creates histograms for each independent attribute in a given dataset. The product of the reverse height in each feature's column yields the anomalous score for each data point. The data are mapped to a high-dimensional space using the Angle-Based Outlier Detection (ABOD) method [8], which then calculates the anomaly score based on the angle discrepancies between the data points and other points. To categorize data points, the One-Class Support Vector Machine (OCSVM) [9] uses the data to learn a decision boundary. Data points are created as nodes in isolation trees using the Isolation Forest (IF) method [10], which assumes that anomalies are uncommon occurrences with feature values that deviate noticeably from anticipated data points. Furthermore, intrusion detection has made use of the Single-Objective Generative Adversarial Active Learning (SO_GAAL) method [11], which is based on generative adversarial networks.

There are various methods for feature extraction using deep learning, including Auto-Encoder(AE) approaches [12]-[15], enhanced Auto-Encoder approaches[16]-[19], Long Short-Term Memory (LSTM) networks [20], Stacked Non-symmetric Deep Autoencoder (SNDAE) [21]-[22], and more. Wang et al. [12] proposed Auto-Encoder (AE) to do dimensionality reduction and feature extraction on the original data. They used an improved K-means technique to further categorize the produced data. Xiao et al. [13] decreased the dimensionality of the data by using Principal Component Analysis (PCA) and AE. The reduced data were then format-ted into pictures. They trained a Convolutional Neural Network (CNN) to deliver the optimal attributes using the transformed pictures. Liu et al. [14] first employed AE with two consecutive hidden layers to extract features. After that, they selected features using Random Forest (RF) and Support Vector Machines (SVM). Kunang et al. [15] used AE for feature extraction and SVM as the classifier.

Furthermore, the use of improved Auto-Encoder (AE) for feature extraction has been extensively studied. Song Yong et al. [16] employed an enhanced Sparse Auto-Encoder to extract features in an intelligent and adaptable manner. Yan et al. [17] used Stacked Sparse Auto-Encoder (SSAE) to extract high-level feature representations of invasion activities. These low-dimensional sparse characteristics were used to construct several foundation classifiers. Yao et al. [18] used algorithms like KNN for anomaly detection and the Variational Autoencoder (VAE) to extract useful features for unsupervised anomaly detection applications. Meghan et al. [19] used support vector machines (SVM) for classification and sparse auto-encoder (SAE) to extract high-level feature representations. Furthermore, Wang et al. [20] proposed two deep feature extraction strategies based on Long Short-Term Memory (LSTM) networks to extract significant features from the data.

Shone et al.'s [21] Stacked Non-symmetric Deep Auto-encoder (SNDAE) is a notable approach in current research on feature extraction using deep learning. SNDAE combines the efficiency of Auto-Encoder (AE) with the benefits of layered learning seen in Stacked Auto-Encoder (SAE). SNDAE has higher unsupervised layered feature learning capabilities as compared to simple AE. SNDAE, unlike SAE, delivers significant data representations without the need for layer wise greedy training approaches. It also provides faster training times and more efficient feature extraction. Wang et al. [22] initially employed Generative Adversarial Networks (GAN) to oversample the dataset. Subsequently, they established a RF for intrusion detection using features extracted by SNDAE.

Nonetheless, there are still gaps in the existing study. Firstly, the current techniques that use deep neural networks for feature extraction still need a lengthy training period, even after several optimizations. Secondly, the curse of dimensionality refers to the fast rise in the runtime of traditional unsupervised anomaly detection algorithms when working with high-dimensional data.

To address the aforementioned issues, we presents the Adam Non-symmetric Auto-Encoder (ANAE) for feature extraction and suggests a modification to the Stacked Non-symmetric Deep Auto-Encoder (SNDAE) based on the Adam optimization approach [23]. During training, this algorithm facilitates a faster convergence of the loss score to the ideal value and boasts a more streamlined network structure. This reduces the training time of the algorithm, and the extracted ideal features contribute to enhancing the model's detection accuracy. COPOD obtained the highest ROC-AUC score among mainstream unsupervised anomaly detection algorithms [24]. Therefore, this paper decided to integrate ANAE feature extraction technique with the COPOD algorithm based on the probability copula function. Through this integration, an effective fusion of feature extraction technique and unsupervised intrusion detection technique was achieved, thus enabling efficient unsupervised intrusion detection. Its effectiveness is validated on the publicly available NSL-KDD dataset and then applied to a real civil aviation air traffic management network environment.

2. Preliminaries

2.1 Auto-encoder

An unsupervised feature learning technique based on neural networks called Auto-Encoder (AE) [25] aims to produce output data x that closely resemble the input data $\hat{x}$. It is frequently applied to feature extraction and data dimensionality reduction. Three layers make up a basic AE: an input layer, an output layer, and a hidden layer. Fig. 1 shows an illustration of AE.

Data from high-dimensional to low-dimensional hidden layers are first encoded by AE using encoder $f(x)$. The low-dimensional data are then rebuilt by decoder $d(x)$ from the hidden layer, yielding the reconstructed data $\hat{x}$. These data are then compared to the input x to determine the reconstruction loss $L\left(x, d\left(f\left(x\right)\right)\right)$. Ultimately, backpropagation is used to update the network. In order to enable the decoder to more precisely rebuild using these newly acquired features, AE aims to learn features that more substantially represent the data distribution during the encoding phase, when data are decreased in dimensionality.

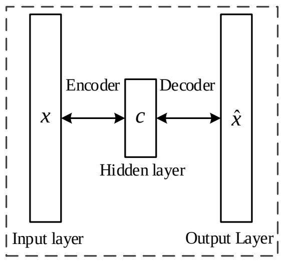

Fig. 1 Auto-encoder

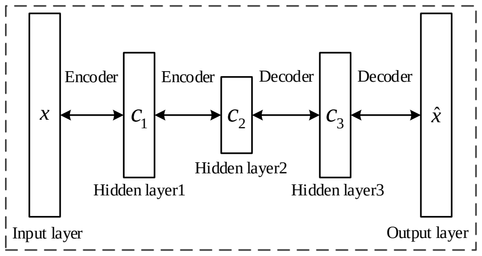

Fig. 2 Stacked Auto-encoder

A Deep Auto-Encoder (DAE) is composed of many layers of encoding and decoding. Hinton and colleagues' work [26] has shown that employing DAE for feature extraction produces lower-dimensional data with enhanced discriminative powers, making it possible to differentiate previously indivisible data. The Stacked Auto-Encoder (SAE) is a common kind of DAE [27]. The layer-wise greedy training technique is used by SAE, in which all previous layers' parameters are while a particular layer is being trained. This technique is performed iteratively to the next layer until the entire network is trained, once each layer's training is finished. Fig. 2 is an illustration of a basic SAE.

Fig. 2 depicts a 5-layer neural network consisting of two symmetrical encoding and decoding stages. The central layer, referred to as the hidden layer 2 or the encoding layer, contains data known as encodings (Code). These encodings represent data that have undergone dimension reduction and is more discriminative, making them suitable for classification and visualization. The features to be extracted are the encodings located in the central hidden layer.

2.2 Copula and COPOD

The Copula function is a probabilistic statistical function used to effectively model the dependencies between multiple random variables. For any d-dimensional random variables with a joint distribution $F\left(x_1,\ldots,x_d\right)$ and marginal distributions $F_1,\ldots,F_d$, there exists a Copula function $F\left(x\right)=C\left(F_1\left(x_1\right),\ldots,F_d\left(x_d\right)\right)$. Using the Copula function, the joint distribution of a multivariate random variable can be represented as a function of each of its marginal distributions.

COPOD (Copula-Based Outlier Detection) algorithm is a probability-based anomaly detection method. It generates an empirical Copula function by calculating the Empirical Cumulative Distribution Functions (ECDF) of the data and estimates the approximate tail probability for each point using this empirical Copula function. For each sample point x_i, the goal of COPOD is to calculate the probability of observing a point as extreme as x_i.

Assuming sample point x_i follows a certain d-dimensional distribution function F_X. COPOD calculates the sample probability $F_X\left(x_i\right)=P\left(X \le x_i\right)$ and $1-F_X\left(x_i\right)=P\left(X \ge x_i\right)$. If x_i is an outlier, it won't appear frequently, and the probability of observing a point as extreme as x_i will be very low. Therefore, if $F_X\left(x_i\right)$ or $1-F_X\left(x_i\right)$ is particularly small, it indicates that the point rarely appears, meaning it is an outlier. COPOD refers to $F_X\left(x_i\right)$ as the left tail probability of x_i and $1-F_X\left(x_i\right)$ as the right tail probability of x_i. If either of these quantities is very small, it means that the point has a very small tail probability.

3. ACOPOD anomaly detection algorithm

We provide a novel model framework that combines traditional machine learning anomaly detection methods with the effective unsupervised feature extraction methodology, ANAE. The core idea is that neural networks capture the most significant distribution of data during the dimensionality reduction process, which improves the difference between various sample types and makes the data in their hidden layers more distinctive. Therefore, conducting intrusion detection on the data from the hidden layers becomes more effective.

COPOD achieved the highest score among existing unsupervised anomaly detection algorithms [24]. Additionally, we compared six unsupervised detection algorithms, including COPOD, on an intrusion detection dataset, and found COPOD achieving the most ideal score. Therefore, we determine to combine the ANAE feature extraction technique with the COPOD algorithm based on the probability copula function. We name this model ACOPOD (ANAE and Copula-Based Outlier Detection), and its overall framework is depicted in Fig. 3.

We start by pre-processing the network traffic data, digitizing it using one-hot encoding, and then standardizing the data to conform to a standard normal distribution. The pre-processed data are divided into a training dataset and a test dataset. The training dataset is used to train ANAE, employing Minibatch batch training and back-propagation to obtain the optimal weights for ANAE. Subsequently, the trained ANAE is used to extract features from the test set's data, extracting it in one go for the entire test dataset, and this extraction is performed twice. Finally, COPOD is employed to classify the data after feature extraction, thereby detecting intrusion behavior.

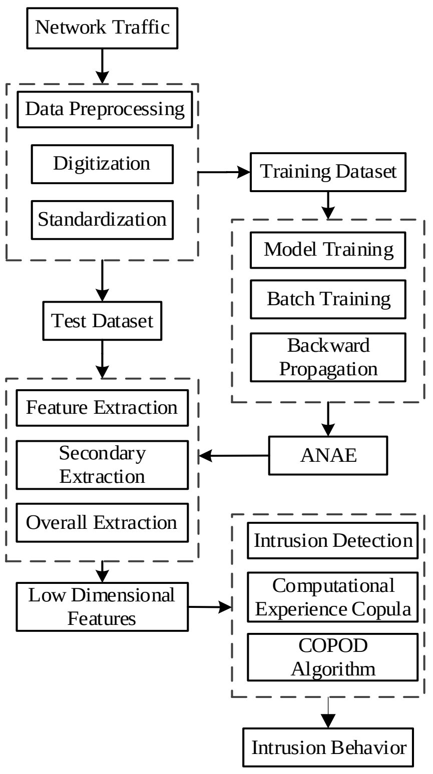

Fig. 3 ACOPOD Model Framework

3.1 Adam-Based Nonsymmetric Autoencoder

When using the Deep Auto-encoder (DAE) [27] for feature extraction, the focus is mainly on the encoder, while the decoder is primarily utilized during training to reconstruct the hidden layer data for computing the reconstruction loss. Once the network is trained, only the encoding operation is executed for feature extraction. In this context, the performance of the encoder is more critical for feature extraction than the decoder. Furthermore, the objective of training a neural network is to learn the maximum knowledge with the fewest neurons, and a streamlined neural network structure can save training time. Therefore, this paper proposes an Adam-based Nonsymmetric Autoencoder (ANAE) that emphasizes the encoder.

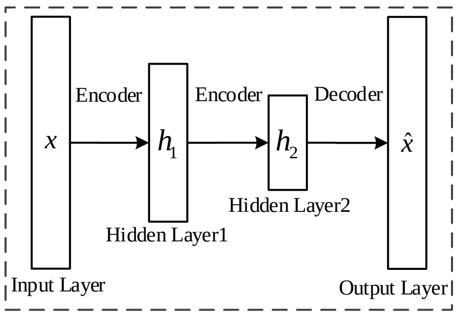

Fig. 4 Adam-based nonsymmetric Autoencoder(ANAE)

Fig. 4 shows an illustration of ANAE. It uses two hidden layers to perform encoding tasks. In contrast to symmetric autoencoders, which rebuild data layer by layer through decoding, ANAE computes the reconstruction loss by performing decoding operations on the hidden layer data just once. This is because the asymmetry in the number of encoding and decoding layers allows the neural network to extract optimum features as long as the loss function converges to an ideal value during training. After comparison, it was found that two hidden layers are sufficient to meet the requirement for extracting significant features, and increasing the number of layers would lead to an increase in feature extraction time.

The Adam optimization algorithm was used during network training. The Adam optimization algorithm [23] is currently a popular neural network optimization method that combines the advantages of the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Compared to other optimization algorithms, it converges faster, requires relatively lower memory, and can adapt to large-scale datasets.

ANAE uses equation (1) to gradually map the input vector $x \in R^d$ to the hidden layer $h_i \in R^{d_i}$, where i refers to the i-th layer of the network, and d represents the vector's dimension.

$$h_i = \sigma\left(W_i \cdot h_{i-1} + b_i\right) \tag{1}$$

In this context, where W and b represent weight and bias, the Sigmoid function $\sigma(t)=\frac{1}{(1+e^{-t})}$ is used as the activation function. During training, it is sufficient to use equation (2) to reconstruct the hidden layer data and generate the output $\hat{x}$:

$$\hat{x}=\sigma(W'\cdot h_{last}+b') \tag{2}$$

Where W' and b' are the parameters of the decoding layer, and h_{last} is the encoding generated by the last encoding layer. Equation (3) represents the loss function, and the purpose of training is to minimize the loss scores for m samples.

$$L(\theta)=\sum_{i=1}^{m}(x_i-\hat{x}_i)^2 \tag{3}$$

The parameter $\theta=(W_i,b_i)$.

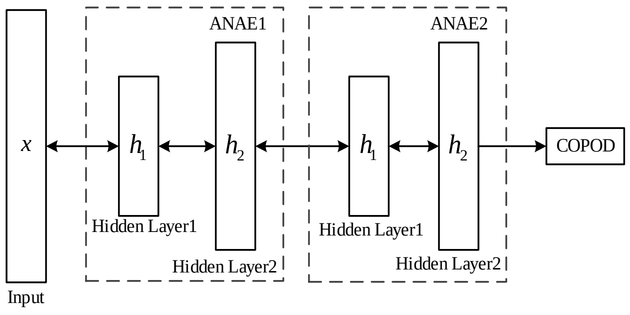

Fig. 5 The network architecture of the ANAE model

As mentioned earlier, we employ a two-stage feature extraction to enhance the effectiveness of feature learning. Two-stage feature extraction involves stacking two ANAEs. After using the first ANAE for feature extraction, the extracted data are passed on to the next ANAE for further feature extraction. This is done to create a deep learning hierarchical structure for hierarchical unsupervised feature learning, aiming to capture the nonlinear and complex relationships between different features. The second round of feature extraction optimizes the data extracted in the first round, making the extracted data more prominent. The ANAE network structure is shown in Fig. 5.

The following provides a brief overview of the execution process of the Adam optimizer. Firstly, the parameter θ is updated using equation (4):

$$\theta_t=\theta_{t-1}-\alpha\cdot\frac{\hat{m}_t}{(\sqrt{\hat{v}_t}+\varepsilon)} \tag{4}$$

Where α represents the learning rate, t keeps track of the steps in parameter updates, and ε is a very small number to prevent division by zero. By computing the exponentially weighted average of gradients, denoted as m_t, and the exponentially weighted average of squared gradients, denoted as v_t, it is possible to estimate the local mean of the parameters. Parameter updates are influenced by their past values over a certain time period.

$$m_t=\beta_1\cdot m_{t-1}+(1-\beta_1)\cdot g_t \tag{5}$$

$$v_t=\beta_2\cdot v_{t-1}+(1-\beta_2)\cdot g_t^2 \tag{6}$$

In equations (5) and (6), $\beta_1,\beta_2\in[0,1)$ are predefined hyperparameters that control the weighting of historical information. g_t represents the gradient of parameter θ at time step t, calculated as $\Delta_\theta L_t(\theta_{t-1})$. g_t^2 is the element-wise squared gradient. In equation (4), and are bias-corrected versions of m_t and v_t, obtained from equations (7) and (8).

$$\hat{m}_t=\frac{m_t}{(1-\beta_1^t)} \tag{7}$$

$$\hat{v}_t=\frac{v_t}{(1-\beta_2^t)} \tag{8}$$

Where β_1^t and β_2^t represent the t-th power of β_1 and β_2, respectively.

Algorithm 1 ACOPOD Anomaly Detection Algorithm

```
Input: $n$ data samples $X_r$, each with $r$ dimensions, after data preprocessing.
Output: Anomaly score $O(X)$.
1 : Train the encoder $f(x): h_i=\sigma(W_i.h_{i-1}+b_i); i=\overline{1,n}$
2 : Train the decoder $d(x): \hat{x}=\sigma(W_{n+1}.h_i+b_{n+1})$
3 : Update the parameters: $\theta_t=\theta_{t-1}-\alpha\cdot\frac{\hat{m}_t}{(\sqrt{\hat{v}_t}+\varepsilon)}$
4 : Feature extraction: $X_d=f(X_r)$
5 : FOR each dimension $d$ DO
6 :     calculate the left-tail ECDF $\hat{F}_d(x)=\frac{1}{n}\sum_1^n I(X_i\le x)$
7 :     calculate the right-tail ECDF: $\hat{\hat{F}}_d(x)=\frac{1}{n}\sum_1^n I(-X_i\le -x)$
8 :     calculate the sample skewness $b_d$ according to formula (10)
9 : END FOR
10 : FOR 1,…n DO
11 :         calculate the empirical copula observations
12 :         $\hat{U}_{d,i}=\hat{F}_d(x_i)$
13 :         $\hat{V}_{d,i}=\hat{\hat{F}}_d(x_i)$
```

```
14 :     IF $b_d < 0$ : $\hat{B}_{d,i} = \hat{U}_{d,i}$
15 :     else $\hat{B}_{d,i} = \hat{V}_{d,i}$
16 :     calculate the tail probability of
17 :         $p_l = -\sum_{j=1}^{d} \log(\hat{U}_{j,i})$
18 :         $p_r = -\sum_{j=1}^{d} \log(\hat{V}_{j,i})$
19 :         $p_s = -\sum_{j=1}^{d} \log(\hat{B}_{j,i})$
20 :     anomaly score $O(x_i) = \max\{p_l, p_r, p_s\}$
21 : END FOR
22 : RETURN $O(X) = [O(x_1), \ldots O(x_d)]^T$
```

3.2 Anomaly detection combining ANAE and COPOD.

When using traditional unsupervised anomaly detection methods like LOF and ABOD, the runtime of these methods increases rapidly as the data dimensionality grows. This situation is referred to as the "curse of dimensionality". Network data, which include categorical variables such as protocols and operational states, can result in high data dimensionality after one-hot encoding. Using LOF, ABOD, and similar methods for detection on such data can undoubtedly lead to the curse of dimensionality. To address this problem, this paper introduces the ACOPOD (ANAE and Copula-Based Outlier Detection) anomaly detection algorithm. Since the extracted features are low-dimensional and significant, with only dimensionality changing compared to the original data, ANAE could be well compatible with other intrusion detection algorithms. This method incorporates the ANAE feature extraction technique to reduce the dimensionality of the data through two-stage feature extraction. The features extracted by ANAE are more prominent and assist the model in distinguishing between normal and anomalous samples, thus improving detection accuracy. After experimental comparisons, the probability-based COPOD method [24] is chosen for anomaly detection on the feature-extracted data.

The input to the ACOPOD algorithm consists of preprocessed data, represented as n samples of r-dimensional data $X_r = (X_{1,i}, X_{2,i}, \ldots, X_{r,i}), i = 1, \ldots, n$. The output is an anomaly score $O(X) = [O(x_1), \ldots O(x_d)]^T$, where the range of anomaly scores is $(0, \infty)$. Anomaly scores do not represent the probability of X_i being an anomaly but rather a relative measure of its likelihood of being an anomaly compared to other points in the dataset. In other words, the larger the $O(X_i)$, the more likely x_i is an anomaly. ACOPOD first utilizes the input X_r to train the ANAE network and then employs the trained ANAE network to perform feature extraction on these data.

For the extracted d-dimensional data X_d, ACOPOD first uses equation (9) to fit the left-tail ECDF for each dimension, $\hat{F}_1(x), \cdots, \hat{F}_d(x)$. Then, it replaces X with $-X$ and fits the right-tail ECDF for each dimension, $\hat{\hat{F}}_1(x), \cdots, \hat{\hat{F}}_d(x)$.

$$\hat{F}(x) = P((-\infty, x]) = \frac{1}{n}\sum_{i=1}^{n} I(X_i \le x) \tag{9}$$

Using equation (10), calculate the skewness of the sample distribution, denoted as $b = [b_1, \ldots, b_d]$. The purpose is to determine whether the distribution leans to the left or to the right, and the algorithm pays more attention to the tail end towards which the distribution leans.

$$b_i = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}_i)^3}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_i)^2}^{\,3}} \tag{10}$$

Using equation (11), insert each x_j into the corresponding ECDF to calculate the empirical Copula observations for each X_i.

$$(\hat{U}_{1,i}, \ldots, \hat{U}_{d,i}) = (\hat{F}_1(X_{1,i}), \ldots, \hat{F}_d(X_{d,i})) \tag{11}$$

This results in left-tail empirical Copula observations $\hat{U}_{d,i} = \hat{F}_d(x_i)$ and right-tail empirical Copula observations $\hat{V}_{d,i} = \hat{\hat{F}}_d(x_i)$. Then, calculate the skewness-corrected empirical Copula observations. If $b_d < 0$, $\hat{B}_{d,i} = \hat{U}_{d,i}$, otherwise, $\hat{B}_{d,i} = \hat{V}_{d,i}$.

Finally, using the empirical Copula observations, calculate the tail probability of X_i. The negative logarithm of the generated probabilities from the left-tail empirical Copula, right-tail empirical Copula, and skewness-corrected empirical Copula is computed, and the maximum value is used as the anomaly score. The smaller the tail probability, the larger its negative logarithm. A point is considered an anomaly if it has a low left-tail probability, a low right-tail probability, or a low skewness-corrected tail probability.

4. Experiments and Results Analysis

Six traditional anomaly detection algorithms were selected for comparative experiments, and the experiments were conducted using the open-source Python toolbox Pyod [28]. The experimental setup included an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz with a 20-core processor and 128GB of RAM. Precision and the area under the receiver operating characteristic curve (ROC-

AUC) were chosen as evaluation metrics. The experimental datasets included the publicly available NSL-KDD dataset [10] and the RCAN (Real Civil Aviation Network) dataset generated from a real civil aviation network environment.

4.1 Evaluation metrics

Using precision based on the confusion matrix and ROC-AUC as evaluation metrics. The definition of the confusion matrix is provided in Table 1.

Table 1 Confusion matrix definition

Sample Class		Prediction	
		Normal	Attack
Reference	Normal	TN	FP
	Attack	FN	TP

Precision is the proportion of correctly predicted samples as attacks out of all samples predicted as attacks. A high precision implies that the model produces fewer false alarms.

$$Precision = \frac{TP}{TP+FP} \tag{12}$$

The ROC curve has the false positive rate (FPR) on the horizontal axis and the true positive rate (TPR) on the vertical axis. The closer the ROC curve is to the top-left corner, the more accurate the model is. The ROC-AUC value, which is the area under the ROC curve, effectively reflects the detection accuracy of unsupervised algorithms at different thresholds. It is a strong indicator of the algorithm's stability, with a maximum value of 1. A higher score indicates better algorithm accuracy.

4.2 NSL-KDD dataset

The NSL-KDD dataset consists of network traffic data. Network traffic data comprise data packets organized in time intervals and are one of the most widely used data sources for Intrusion Detection Systems (IDS) [10]. This dataset is a benchmark dataset released by the Canadian Institute for Cybersecurity. Many research studies have validated their findings on this dataset, which is highly authoritative, making experimental results on this dataset more convincing.

The NSL-KDD [27] dataset addresses issues such as data redundancy in the original KDD dataset, making it an optimized version. There are a total of 125,973 training records and 22,543 testing records. The dataset includes four types of attacks: Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). DoS and Probe attacks involve short-duration attacks or scanning multiple hosts, establishing numerous connections. In contrast, R2L and U2R attacks are embedded in the data part of packets and typically only involve a single connection. Data statistics are presented in Table 2.

Table 2 Statistics of the NSL-KDD dataset

Data categories	Training dataset	Test dataset
DoS	45927	7460
Probe	11656	2421
R2L	995	2885
U2R	52	67
Normal	67343	9711
Total	125973	22543

4.3 RCAN dataset

The intrusion detection field currently faces challenges of insufficient high-quality data, and most intrusion detection methods are primarily validated on public simulated datasets, with their detection capabilities in real network environments yet to be verified. To address this, we extracted data from a real civil aviation air traffic management network spanning a month, and after processing, created the RCAN dataset. The original data comprise a total file size of 206GB, with 80GB dedicated to Netflow data, and the remainder recording messages related to POP3, SMTP, and DNS protocols. The Netflow data consist of 450,000 records on the first day, and the daily count reaches millions for the subsequent days, exceeding 3 million records on the second, third, and fourth days.

Netflow data are based on sessions, which represent the interaction process between two terminal applications [29]. Netflow data do not store raw traffic data, but instead records fields from the packet headers of each session, such as packet count and session duration. A session is typically defined by a five-tuple (client IP, client port, server IP, server port, and protocol). There are two advantages to using sessions for detection: (1) sessions are suitable for detecting attacks between specific IP addresses, such as tunneling and Trojan horse attacks; (2) sessions contain detailed communication between attackers and victims, which aids in locating the source of the attack.

The data are stored in JSON format as key-value pairs, with each initial data message having a double-layered nested structure. The outer layer contains fields such as file name, file type, ID, and other information about the data. The inner layer values record the statistical information of a single session, with a total of 68 fields. These fields provide detailed information about the session's status, service type, packet count, etc., which is advantageous for the model to learn the features of sessions and distinguish between different types of sessions.

4.4 Data Preprocessing

Using One-Hot Encoding for the digitization of categorical features allows for a more reasonable calculation of distances between features. StandardScaler is employed to process the data, ensuring it conforms to a standard normal distribution with a mean of 0 and a standard deviation of 1. Compared to normalization, standardization better preserves the distances between samples. The transformation function is given by equation (12), where μ denotes the mean of all sample data, and σ represents the standard deviation of all sample data.

$$x^* = \frac{x-\mu}{\sigma} \qquad (13)$$

The NSL-KDD dataset has 41 features, of which “protocol_type”, “service”, “flag” and “label” are categorization variables. There are 3 categories for “protocol_type”, 70 categories for “service” and 11 categories for “flag”. After performing one-hot encoding on these categorical variables, the data dimensionality increases from the original 41 dimensions to 122 dimensions. The dataset has been pre-divided into training and testing sets. Standardization has been applied separately to ensure the data conform to a normal distribution. Individual datasets have been created for the four types of attacks, meaning each attack dataset contains only instances of that specific attack without the presence of other attacks.

When analyzing the 68 fields in the JSON file with double-layer nesting, four types of fields were identified that cannot be directly analyzed by the model. The first type of field has values that are all zeros, making these features meaningless for array calculations in machine learning. These fields include ‘reliability’, ‘out_conn_type’, ‘l3_protocol’, ‘l7_protocol’, etc. The second type of field is inherited directly from other types of data during development, such as ‘level’, ‘target’, ‘sub_attack_type’, ‘attack_type’, etc. These fields are inherited from security data, which record data detected by probes and data hit by rules. The recorded data are not necessarily problematic; it just might be, and some values are also all zeros. The third type is reserved fields, such as ‘index1’, ‘index2’, ‘index3’, etc. Reserved fields have no practical meaning. The fourth type is fields that record IDs, such as ‘src_ip’, ‘dst_ip’, ‘dev_id’, etc. These fields record IP addresses and device IDs and can be used for subsequent anomaly traceback analysis. After removing the above meaningless fields, 24 fields were retained for the experiment, as shown in Table 3.

Table 3 Reserved fields used for model analysis

Index	Field	Type	Field Description
1	‘record_type’	classification	Access type
2	‘rule_major_type’	classification	The main rule type
3	‘rule_minor_type’	classification	The secondary rule type
4	‘src_type’	classification	The group to which the source belongs
5	‘dst_type’	classification	The group to which the destination belongs
6	‘src_branch_type’	classification	Source branch type
7	‘dst_branch_type’	classification	Destination branch type
8	‘session_state’	classification	Session state
9	‘net_action’	classification	Action
10	‘src_port’	numeric value	Source port
11	‘dst_port’	numeric value	Destination port
12	‘l4_protocol’	classification	The Layer 4 protocol comes from the RF document protocol ID, which is the same as the protocol number in the IP header
13	‘rule_id’	classification	Rule ID
14	‘serv_crc’	classification	Application type CRC
15	‘src_country_crc’	classification	Source country
16	‘dst_country_crc’	classification	Country of destination
17	‘src_province_crc’	classification	The province to which the source belongs
18	‘dst_province_crc’	classification	The province to which the destination belongs
19	‘request_flow’	numeric value	Request traffic size
20	‘response_flow’	numeric value	Response traffic size
21	‘request_pack’	numeric value	Request packet size
22	‘response_pack’	numeric value	Response packet size
23	‘session_time’	numeric value	Session time
24	‘addition_offset’	numeric value	Additional offset

In the retained fields, there are 16 categorical features and 8 numerical features. RCAN is a dataset composed of values extracted from these 24 fields. The fields retaining information about the source and destination countries, as well as the source and destination provinces, have practical significance in network security analysis. Moreover, their fixed categories ensure that encoding them does not lead to the curse of dimensionality. Digitalization is performed using one-hot encoding, which, compared to label encoding, makes distance calculations between samples more reasonable. Among the 24 features in RCAN, 16 are categorical, including “session status”, “protocol type”, “application type”, etc. The “session status” includes 6 different categorical variables, “protocol type” includes 3 different categorical variables, and “application type” includes 15 different categorical variables. After digitizing these categorical features, the dimensionality of the RCAN dataset is expanded to 67. Finally, standard deviation standardization (StandardScaler) is applied to the data to make it conform to a standard normal distribution.

4.5 Analysis of NSL-KDD dataset results

In this paper, the LOF algorithm based on the calculation of sample distance[5], the ABOD algorithm based on the calculation of sample angle [8], the OCSVM algorithm based on the calculation boundary [9], the IF algorithm [10], the SO_GAAL algorithm based on generative adversarial network [11] and the COPOD algorithm [24] are selected as benchmarks, and the ACOPOD algorithm is compared with the public dataset NSL-KDD. These six benchmark algorithms are all classic unsupervised algorithms, which are often used to implement intrusion detection. In particular, the OCSVM algorithm is a very popular unsupervised intrusion detection solution. The effectiveness of the ACOPOD algorithm is validated by comparing it with these benchmarks on the public NSL-KDD dataset. Precision, ROC-AUC values, and runtime are computed for each algorithm on the dataset to assess the effectiveness of the ACOPOD algorithm.

The precision of the seven algorithms on the NSL-KDD dataset is shown in Table 4. The numbers in parentheses represent the ranking of the algorithm's detection accuracy for that attack, with bold font indicating the highest ranking. "Total" represents the simultaneous detection of all four attacks in the NSL-KDD dataset. Algorithms such as LOF, ABOD, and OCSVM are implemented using the Pyod tool, which automatically selects the optimal parameters based on input data, eliminating the need for manually setting algorithm parameters. ANAE is used to reduce the data from 122 dimensions to 50 dimensions, with a learning rate of 0.01, a batch size of 64, and 1000 batch training iterations.

From Table 5, it can be observed that for DoS attacks, the ACOPOD algorithm ranks first in precision, achieving a 2.65% improvement over the second-ranked ABOD algorithm. For Probe attacks, the ACOPOD algorithm ranks fourth, achieving a precision of 75.18%. For R2L and U2R attacks, the ACOPOD algorithm ranks second, indicating that the detection results for these two types of attacks are suboptimal for all four algorithms, likely due to the scarcity of anomalous samples. In the detection of the complete NSL-KDD dataset, the ACOPOD algorithm achieves a precision of 83.19%, ranking first and outperforming the second-ranked COPOD algorithm by 4.25%. Overall, the ACOPOD algorithm demonstrates favorable precision on the NSL-KDD dataset. Precision reflects only the detection results achieved by the algorithm at a specific threshold. To comprehensively showcase the algorithm's performance at different thresholds, ROC curves for ACOPOD on the four types of attacks in the NSL-KDD dataset are plotted. From Fig. 6, it can be observed that ACOPOD maintains high precision across different thresholds for the detection of all four attacks, demonstrating strong overall performance.

Table 4 Precision of seven algorithms on NSL-KDD dataset

Attack	LOF	ABOD	OCSVM	IForest	SO_GAAL	COPOD	**ACOPOD**
DoS	0.658(5)	0.790(2)	0.584(6)	0.731(4)	0.357(7)	0.777(3)	**0.815(1)**
Probe	0.212(5)	0.178(7)	0.797(3)	**0.866(1)**	0.323(5)	0.865(2)	0.752(4)
R2L	0.308(6)	0.444(3)	0.415(4)	0.388(5)	0.181(7)	**0.506(1)**	0.464(2)
U2R	0.015(2)	0.015(2)	0	0.119(3)	0	**0.493(1)**	0.015(2)
Total	0.596(7)	0.760(4)	0.779(3)	0.724(5)	0.640(6)	0.789(2)	**0.832(1)**

Table 5 ROC-AUC values of seven algorithms on NSL-KDD dataset

Attack	LOF	ABOD	OCSVM	IForest	SO_GAAL	COPOD	**ACOPOD**
DoS	0.669(5)	0.834(3)	0.623(6)	0.825(4)	0.432(7)	0.882(2)	0.897(1)
Probe	0.345(7)	0.470(5)	0.967(3)	0.980(2)	0.428(6)	0.983(1)	0.949(4)
R2L	0.483(6)	0.729(4)	0.615(5)	0.731(3)	0.357(7)	0.771(2)	0.839(1)
U2R	0.552(6)	0.745(5)	0.78(4)	0.867(2)	0.858(3)	0.958(1)	0.882(2)
Total	0.529(7)	0.740(5)	0.868(2)	0.744(4)	0.648(6)	0.823(3)	0.894(1)

Table 6 The running time of the seven algorithms on the NSL-KDD dataset

Attack	LOF	ABOD	OCSVM	IForest	SO_GAAL	COPOD	**ACOPOD**
DoS	81	1384	465	37	1594	14	13
Probe	43	517	231	26	1085	10	10
R2L	38	469	206	22	928	8	10
U2R	27	352	158	22	900	7	10
Total	428	1898	5452	37	1644	26	14

To numerically represent the ROC performance of ACOPOD and facilitate comparison with benchmark algorithms, the ROC-AUC values for the seven algorithms on the NSL-KDD dataset are calculated and presented in Table 5.

From Table 4, it can be observed that for DoS attacks, ACOPOD algorithm ranks first in terms of ROC-AUC value, with an increase of 0.0147 compared to the second-ranking algorithm. For Probe attacks, ACOPOD algorithm ranks fourth, reaching 0.9486. In the case of R2L attacks, ACOPOD algorithm ranks first, with an increase of 0.067 compared to the second-ranking algorithm. For U2R attacks, ACOPOD algorithm ranks second, reaching 0.8821. In the detection of the entire dataset, ACOPOD algorithm ranks first, achieving a ROC-AUC value of 0.8936, with an increase of 0.0257 compared to the second-ranking OCSVM algorithm. From the ROC-AUC values, it is evident that ACOPOD exhibits superior performance, particularly in the detection of U2R attacks, providing higher precision with different threshold settings. Overall, ACOPOD outperforms other algorithms in terms of ROC-AUC values.

Algorithm runtime is an important evaluation metric for intrusion detection algorithms, representing the efficiency of the algorithm. An ideal algorithm should exhibit both high precision and low runtime to cope with the rapidly growing volume of network data. In this regard, this paper provides a summary of the runtime for each algorithm on the NSL-KDD dataset, as shown in Table 6. The numerical values in the table are presented in seconds.

From Table 6, it can be observed that, for the average runtime across the four types of attacks, ACOPOD algorithm takes 10.75 seconds, COPOD algorithm takes 9.75 seconds, IForest algorithm takes 26.75 seconds, LOF algorithm takes 206.25 seconds, ABOD algorithm takes 798.25 seconds, SO_GAAL algorithm takes 1126.75 seconds, and OCSVM algorithm takes 2544.75 seconds, being the most time-consuming. For the detection of the complete dataset, ACOPOD algorithm has the least runtime. Unlike other algorithms that compute the initial dimensional sample distances, ACOPOD algorithm first reduces the sample dimensionality and then calculates the probability of a sample being an anomaly.

Reducing the dimensionality of the data can significantly decrease the computation time of the algorithm. After feature extraction, it can be observed that the total computation time of the ACOPOD algorithm (including feature extraction time) is shorter than that of not using the COPOD algorithm. On this dataset, the ACOPOD algorithm reduces the data from 122 dimensions to 55 dimensions, with 55 dimensions being the optimal value determined through comparison.

The NSL-KDD dataset is a publicly available simulated dataset. Compared to real network environments, it has a relatively small data size, only 2GB. Therefore, the runtime of other unsupervised algorithms is relatively fast. However, the runtime of the ACOPOD algorithm differs significantly from other algorithms, and the ACOPOD algorithm has higher precision and ROC-AUC scores.

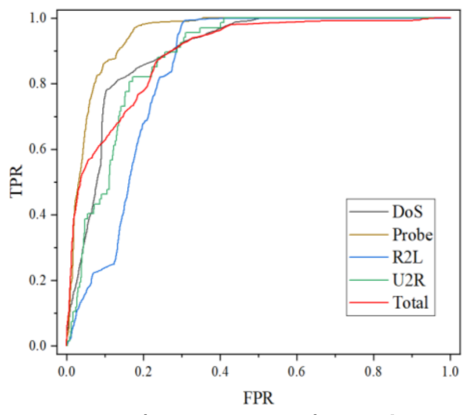

Fig. 6 ROC curves of ACOPOD against four attacks on NSL-KDD dataset

4.6 Analysis of RCAN dataset results

To further validate the practicality and effectiveness of the ACOPOD algorithm, it was applied to the unlabeled RCAN dataset, and the detection results were analyzed through manual tracing. Approximately 2.3 million records from the RCAN dataset were selected for detection, and ANAE was used to reduce the data from 67 dimensions to 30 dimensions. The learning rate was set to 0.01, batch size to 128, and the number of batch training iterations to 10,000. The ACOPOD algorithm detected about 230,000 abnormal records. Using Principal Component Analysis (PCA), the data were further reduced to three dimensions, as shown in Fig. 7, where blue dots represent normal samples, and red dots represent detected abnormal samples.

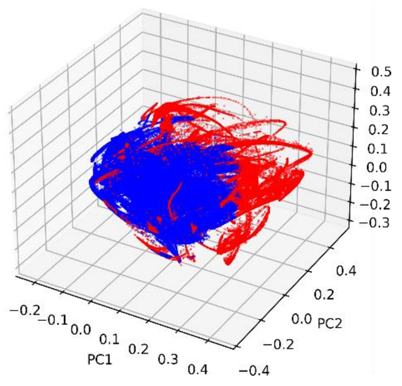

Fig. 7 3D visualization of anomaly data dataset

From Fig. 7, it can be observed that the data, after feature extraction, is uniformly distributed, and the detected abnormal samples by the algorithm are located at the edges of the overall data distribution. These data were captured from a real network environment without manual labeling. Validating the source messages for all 230,000 abnormal records one by one is a highly time-consuming task. Therefore, to save time, the analysis primarily focuses on tracing anomalies at the level of abnormal IP addresses. This involves exporting the abnormal messages associated with a specific IP to a CSV file, followed by verifying whether they indicate a network attack.

During the data collection process, the intrusion detection system deployed in the network identified some real network attack events, such as IP scanning, vulnerability exploitation, and worm incidents. The source data of these actual network attack events, organized by IP, were extracted to create an anomaly template. During validation, the CSV file organized by IP was compared against these anomaly templates, and the authenticity of abnormal messages was verified by analyzing and comparing features such as source port, destination port, and IP addresses. Since these data were captured in an intranet, the host IPs are internal LAN addresses, ensuring the confidentiality of sensitive information. Fig. 8 illustrates the top 20 IP addresses with the highest counts of abnormal messages detected by the ACOPOD algorithm.

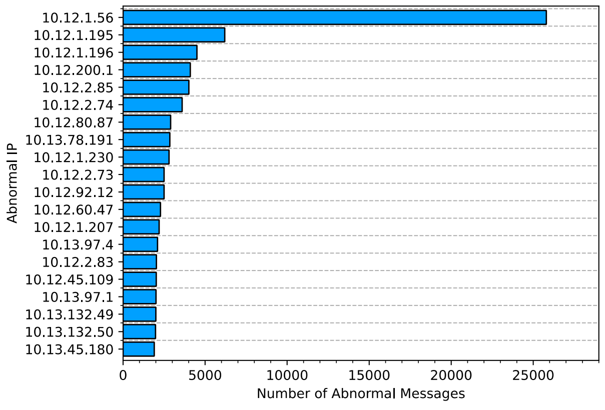

Fig. 8 Abnormal IP address statistics

The following analysis will use several typical IPs from Fig. 8 as examples to demonstrate how manual verification is performed. From Fig. 8, it can be observed that the IP with the highest count of abnormal messages is “10.12.1.56”. Analyzing its destination ports and destination IPs reveals that this IP exhaustively accesses common high-risk ports across multiple IP ranges during fixed time intervals (each scanning period is approximately two minutes). These ports include 445, 161, 137, 139, among others, which are commonly targeted high-risk ports for port scanning or network attacks. Hence, it can be concluded that this IP is engaged in network scanning behavior. For IPs “10.12.1.195” and “10.12.1.196”, they establish long-term connections with external IPs through port 4000. Port 4000 is an open port for certain communication software. Investigation revealed that the majority of the external IPs belong to overseas locations. Upon further inquiry, it was confirmed that the real network in question does not connect to the external network, let alone establish connections with foreign IPs. Therefore, it can be deduced that these hosts are undergoing a network attack.

It is worth noting that for the IP “10.12.80.87”, analysis revealed that this IP communicated multiple times with other hosts using port 8000. Upon inquiry, it was found that port 8000 is a commonly used port for a social software. However, in this network, the use of unnecessary software is not allowed, and there is no connection to the external network. Therefore, the IP does exhibit anomalous behavior. However, further verification indicated that this behavior did not constitute a network attack. This proves that in a real network environment, anomalous data detected by unsupervised algorithms may not necessarily be indicative of a network attack. It could also be anomalous behavior deviating from the norm. In other words, while a network attack is always an anomalous behavior, not all anomalous behaviors are network attacks. Network attacks are a subset of anomalous behavior.

The network environment utilized in this method is a real civil aviation network with 1279 active nodes, from which traffic data were collected over a period of 30 days, totaling 203 gigabytes. This civil aviation network environment is quite common and typical within the entire civil aviation system. Therefore, this method is equally applicable to other networks within the civil aviation system, as well as to other similarly sized real-world networks. False positives are inevitable in unsupervised detection. For this real network, there are several skilled technicians at the same time to mannually handle false positives.

Thus, ACOPOD is suitable for small and medium-sized networks with a certain number of skill technicians. For large-scale networks, supervised intrusion detection systems should be added to handle false positives, so as to reduce the workload of manual verification.

Conclusion

This paper proposes a novel unsupervised intrusion detection algorithm. Firstly, it utilizes the simplified structure of the Adam Non-symmetric Auto-Encoder(ANAE) to achieve unsupervised feature extraction. This not only ensures detection accuracy but also enhances operational efficiency. Subsequently, the algorithm employs the Probability-based COPOD method to perform anomaly detection on the low-dimensional data obtained through feature extraction. Experimental results on the NSL-KDD public dataset demonstrate that the ACOPOD algorithm achieves favorable precision, high ROC-AUC values, and overall outperforms benchmark algorithms. Additionally, the detection runtime on the complete NSL-KDD dataset is significantly lower than benchmark algorithms. Experimental verification on the real network’s

RCAN dataset, through manual validation of the anomalous data detected by ACOPOD, indicates that the method can effectively detect intrusion behavior in unlabeled real network data. Finally, traceback analysis of anomalous IPs demonstrates that network attacks are encompassed by anomalous behavior, affirming that a network attack is always anomalous behavior, but anomalous behavior is not necessarily a network attack.

There are still some issues to be improved in the future. As a retrospective algorithm, unsupervised intrusion detection suffers from lower precision compared to supervised algorithms. For instance, approximately 20% of DoS attacks are overlooked in the experiments. Therefore, we will continue to explore methods to enhance the precision of unsupervised intrusion detection in the future work. Additionally, there are plans to apply the ACOPOD algorithm to detect other network attacks.

Acknowledgements

This work was supported by National Science Foundation of China under Grants U2333201; National Key R&D Program of China under Grant 2021YFF0603902; and Civil Aviation Safety Capacity Building Foundation of China under Grants PESA2022093 and PESA2023101.

References

[1] H. Zhang, X. Zhang, Z. Zhang, and W. Li, “Summary of intrusion detection models based on deep learning,” Computer Engineering and Applications(in Chinese), vol.58, no.06, pp.17-28, 2022.

[2] S. Hou, A. Saas, L. Chen, and Y. Ye, “Deep4maldroid: A deep learning framework for android mal-ware detection based on linux kernel system call graphs,” Proc. Conf. on Web Intelligence Workshops (WIW), pp.104-111, 2016.

[3] E. Schubert, A. Koos, T. Emrich, A. Züfle, K.A. Schmid, and A. Zimek, “A framework for clustering uncertain data,” Proceedings of the VLDB Endowment, vol.8, no.12, pp.1976-1979, 2015.

[4] S. Ramaswamy, R. Rastogi, and K. Shim, “Efficient algorithms for mining outliers from large data sets,” Proc. Conf. on Management of data, Dallas, Texas, USA, pp.427-438, May. 2000.

[5] M.M. Breunig, H.-P. Kriegel, R.T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” Proc. Conf. on Management of data, Dailas, Texas, USA, pp.93-104, May. 2000.

[6] J. Tang, Z. Chen, A.W.-C. Fu, and D.W Cheung, “Enhancing effectiveness of outlier detections for low density patterns,” Proc. Conf. on Knowledge Discovery and Data Mining, Berlin, Heidelberg, pp.535-548, Jan. 2002.

[7] M. Goldstein and A. Dengel, “Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm,” Proc. Conf. KI-2012: Poster and Demo Track, pp.59-63, Sep. 2012.

[8] H.-P. Kriegel, M. Schubert, and A. Zimek, “Angle-based outlier detection in high-dimensional data,” Proc. Conf. on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, pp.444-452, Aug. 2008.

[9] M. Amer, M. Goldstein, and S. Abdennadher, “Enhancing one-class support vector machines for unsupervised anomaly detection,” Proc. Conf. on Outlier Detection Description, Chicago, Illinois pp.8-15, Aug. 2013.

[10] F. Liu, K. Ting, and Z. Zhou, “Isolation forest,” Proc. Conf. on Data Ming, Pisa, Italy, pp.413-422, Dec. 2008.

[11] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He, “Generative adversarial active learning for unsupervised outlier detection,” IEEE Transactions on Knowledge and Data Engineering, vol.32, no.8, pp.1517-1528, 2020.

[12] X. Wang and L. Wang, “Research on intrusion detection based on feature extraction of autoencoder and the improved k-means algorithm,” Proc. Conf. on Computational Intelligence and Design, Hangzhou, China, pp.352-356, Dec. 2017.

[13] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, “An intrusion detection model based on feature reduction and convolutional neural networks,” IEEE Access, vol.7, pp.42210-42219, 2019.

[14] J. Liu and S. Chung, “Automatic feature extraction and selection for machine learning based intrusion detection,” Proc. Conf. 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, Leicester, UK, pp.1400-1405, Aug. 2019.

[15] Y.N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, Firdaus, and Jasmir, “Automatic features extraction using autoencoder in intrusion detection system,” Proc. Conf. on Electrical Engineering and Computer Science, Pangkal, Indonesia, pp.219-224, Oct. 2018.

[16] Y. Song, B. Hou, and Z. Cai, “Network intrusion detection method based on deep learning feature extraction,” Journal of Huazhong University of Science and Technology(Natural Science Edition)(in Chinese) , vol.49, no.02, pp.115-120, 2021.

[17] B.H. Yan, G.D. Han, “Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system,” IEEE Access, vol.6, pp.41238-41248, 2018.

[18] R. Yao, C. Liu, L. Zhang, and P. Peng, “Unsupervised anomaly detection using variational auto-encoder based feature extraction,” Proc. Conf. on Prognostics and Health Management, San Francisco, CA, USA, pp.1-7, June. 2019.

[19] S.N Mighan and M. Kahani, “Deep learning based latent feature extraction for intrusion detection,” Proc. Conf. on Electrical Engineering, Mashhad, Iran, pp.1511-1516, May. 2018.

[20] A. Wang, X. Gong, and J. Lu, “Deep feature extraction in intrusion detection system,” Proc. Conf. on Smart Cloud (SmartCloud), Tokyo, Japan, pp.104-109, Dec. 2019.

[21] N. Shone, T.N Ngoc, V.D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” IEEE Transactions on Emerging Topics in Computational Intelligence, vol.2, no.1, pp.41-50, Feb. 2018.

[22] S. Wang, H. Chen, L. Ding, H. Sui, and J.L. Ding, “GAN-SR anomaly detection model based on imbalanced data,” IEICE TRANSACTIONS on Information and Systems, vol.E106-D, no.7, pp.1209-1218, 2023.

[23] D.P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[24] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, “COPOD: copula-based outlier detection,” Proc. Conf. on Data Mining, Sorrento, Italy, pp.1118-1123, Nov. 2020.

[25] J. Lai, X. Wang, Q. Xiang, Y. Song, and W. Quan, “Review on autoencoder and its application,” Journal on Communications (in Chinese) vol.42, no.9, pp.218-230, 2021.

[26] G.E. Hinton and R.R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” science, vol.313, no.5786, pp.504-507, 2006.

[27] Y.S. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” Advances in neural information processing systems 19, pp.153-160, 2006.

[28] Y. Zhao, Z. Nasrullah, and Z. Li, “Pyod: A python toolbox for scalable outlier detection,” arXiv preprint arXiv:1901.01588,

2019.

[29] H. Liu and B. Lang, “Machine learning and deep learning methods for intrusion detection systems: A survey,” applied sciences, vol.9, no.20, pp.4396, 2019.

Chunbo Liu received the B.S. and M.E. degrees in Computer Science from Nankai University, China, in 1999 and 2004, respectively. He is an associate professor in the Information Security Evaluation Center, Civil Aviation University of China. His research interests include cyber security and intelligent detection.

Liyin Wang received the B.E. degree in Software Engineering from Tiangong University, China, in 2019, and the M.E. degree in Computer Technology from Civil Aviation University of China in 2022. He is an assistant engineer in the Aeronautical Information Service Center, ATMB, CAAC. His research include intrusion detection and machine learning.

Zhikai Zhang received the B.E. degree in Transportation from Civil Aviation University of China in 2020. He is currently pursuing the M.E. degree in Computer Science with Civil Aviation University of China. His research interests include log anomaly detection and natural language processing.

Chunmiao Xiang received the B.E. degree in Software Engineering from Chengdu University of Information Technology, China, in 2022. She is currently pursuing the M.E. degree in Computer Science with Civil Aviation University of China. Her research interests include log anomaly detection and intrusion detection.

Zhaojun Gu received the M.E. degree in Computer Science from Harbin Institute of Technology and D.S. degree in Computer Science from Nankai University, China, in 1996 and 2004, respectively. He is a professor in the Information Security Evaluation Center, Civil Aviation University of China. His research interests include cyber security and information systems in civil aviation.

Shuang Wang received the M.E. degree in Computer Science from Civil Aviation University of China, in 2013. She is an assistant researcher in the Civil Aviation University of China. She is currently pursuing the Ph.D. degree with Civil Aviation University of China. Her research interests include computational intelligence and machine learning, industrial control system information security.

Zhi Wang received the B.E. degree in Computer Science from Hebei University of Technology, China, in 2005, and D.S. degree in Computer Science from Nankai University, China, in 2012. He is an associate professor in the School of Cyber Science, Nankai University. His research interests include malware analysis and binary reverse engineering.