SCIENCE CHINA
Information Sciences

July 2024, Vol. 67, Iss. 7, 170304:1–170304:19
https://doi.org/10.1007/s11432-023-4067-x

• RESEARCH PAPER •

Special Topic: XAI-Enabled Network Automation for 5G and Beyond: Move Closer to Zero-Touch

HEN: a novel hybrid explainable neural network based framework for robust network intrusion detection

Wei WEI[1], Sijin CHEN[2], Cen CHEN[3,4*], Heshi WANG[5], Jing LIU[2*], Zhongyao CHENG[6] & Xiaofeng ZOU[3]

[1]*School of Computer Science and Engineering, Xi'an University of Technology, Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an* 710048*, China;*
[2]*School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan* 430065*, China;*
[3]*School of Future Technology, South China University of Technology, Guangzhou* 510641*, China;*
[4]*Shenzhen Research Institute of Hunan University, Shenzhen* 518052*, China;*
[5]*School of Computer Science, Hunan University of Technology and Business, Changsha* 410205*, China;*
[6]*Institute for Infocomm Research (I2R), Agency for Science, Technology and Reseach (A*STAR), Singapore* 138632*, Singapore*

Received 2 August 2023/Revised 14 February 2024/Accepted 7 June 2024/Published online 28 June 2024

Abstract With the rapid development of network technology and the automation process for 5G, cyber-attacks have become increasingly complex and threatening. In response to these threats, researchers have developed various network intrusion detection systems (NIDS) to monitor network traffic. However, the incessant emergence of new attack techniques and the lack of system interpretability pose challenges to improving the detection performance of NIDS. To address these issues, this paper proposes a hybrid explainable neural network-based framework that improves both the interpretability of our model and the performance in detecting new attacks through the innovative application of the explainable artificial intelligence (XAI) method. We effectively introduce the Shapley additive explanations (SHAP) method to explain a light gradient boosting machine (LightGBM) model. Additionally, we propose an autoencoder long-term short-term memory (AE-LSTM) network to reconstruct SHAP values previously generated. Furthermore, we define a threshold based on reconstruction errors observed during the training phase. Any network flow that surpasses the specified threshold is classified as an attack flow. This approach enhances the framework's ability to accurately identify attacks. We achieve an accuracy of 92.65%, a recall of 95.26%, a precision of 92.57%, and an F1-score of 93.90% on the dataset NSL-KDD. Experimental results demonstrate that our approach generates detection performance on par with state-of-the-art methods.

Keywords explainable artificial intelligence, light gradient boosting machine, machine learning, network intrusion detection, Shapley additive explanation, hybrid explainable neural network (HEN)

1 Introduction

With the explosive growth of Internet technology and radical revolution in industry 4.0 networking [1,2], cybersecurity issues are increasingly posing significant threats to both corporate and individual assets and privacy [3]. The continual growth of online traffic has complicated the task of discerning different types of network activities, as malicious traffic effectively blends with regular traffic, making it challenging for cybersecurity personnel to distinguish. If such malicious traffic is not accurately identified and intercepted, it could potentially compromise user accounts or launch attacks on websites and servers.

At the same time, with the popularization of 5G technology, the threat of cyber attacks is also increasing. Due to the unique high speed and low latency of 5G networks, hackers have more opportunities to conduct network intrusions. In addition, with the development of hardware, electronic money or e-Cash is becoming increasingly popular as the preferred strategy for making purchases [4]. This also raises the issue of network security concerning people's property. To meet the future demand for a huge traffic

* Corresponding author (email: chencen@scut.edu.cn, luijing_cs@wust.edu.cn)

© Science China Press 2024

volume of data service and 5G systems high spectral [5] and energy efficiencies [6], it is important to further exploit the potential of antenna systems [7–11] and detection networks.

In response to these threats, both the industry and academia are striving to develop network intrusion detection systems [12]. However, with the constant evolution of network communication, network activities are becoming increasingly sophisticated, and novel methods of network attacks continue to emerge. These changes present serious challenges to the design and detection capabilities of network intrusion detection systems (NIDS) that have remained strong for some time.

In recent years, with the emergence and development of machine learning, deep learning has made amazing achievements in image recognition, speech recognition, natural language processing, etc. Machine learning technology has excellent performance in dealing with complex and large-scale data, which also brings new ideas for dealing with multi-feature intrusion data. Therefore, many experts and researchers apply machine learning (ML) to network intrusion detection. Since it can learn from network traffic data sets and simulate real traffic, the flexible application of deep learning in the field of network intrusion detection can effectively improve the detection rate and reduce the false positive rate and false negative rate, and ML has achieved remarkable achievements in the field of NIDS [13–15].

For instance, Panigrahi et al. [16, 17] presented a host-based intrusion detection system using a C4.5-based detector on top of the popular consolidated tree construction (CTC) algorithm. Taher et al. [15] utilized artificial neural networks (ANN) and support vector machines (SVM) to conduct intrusion detection through supervised ML. However, supervised ML struggles to address novel (zero-day) attacks effectively. Unsupervised learning methods, such as autoencoders, have shown promising potential in this aspect [18–20]. For example, Yang et al. [20] combined deep neural networks (DNN) with improved conditional variational autoencoders (ICVAE), enhancing the detection rate of minority attacks and unknown attacks. Ravi et al. [21] proposed an end-to-end model using deep learning-based recurrent models for network attack detection and network attack classification. They employed a kernel-based principal component analysis (KPCA) to identify optimal features. Wang et al. [22] were the first to propose an effective autoencoder (AE) detection scheme called MANDA to defend against attacks. They were also the first to investigate AE attacks for intrusion detection systems (IDS) in problem space rather than in feature space. Issa et al. [23] proposed a novel deep learning classification method by mixing two common deep learning algorithms, achieving higher detection capabilities against distributed denial of service (DDoS) attacks. In addition, the autoencoder long-term short-term memory (AE-LSTM) model is an autoencoder architecture that utilizes long short-term memory (LSTM) and employs unsupervised learning to efficiently learn data representations. It consists of multiple LSTM networks, where the encoder network's output is directly fed into the encoder network. Many researchers use AE-LSTM [19, 24] to reconstruct traffic, and they all reflect the powerful reconstruction ability of AE-LSTM.

Although these models have achieved high accuracy in network intrusion detection, there are still some challenges in practical application. Except for high detection performance, several network security also think about interpretability to be a more important feature of steady-point NIDS. In particular, the recent emergence of machine learning-based network intrusion detection systems, and their inherent "black box" nature means that many of today's deep neural networks do not have the means to fully understand the model's decisions from a human perspective. Even if malicious traffic is successfully detected, significant human resources are needed to find out why the flow is malicious and how to respond to such malicious traffic [25]. In recent years, interpretable artificial intelligence (AI)-related work has emerged [26–28], giving rise to a new wave of cybersecurity work, most of which includes additional layers of interpretability to machine learning models. For example, Barnard et al. [29] employed DNN for encoding and decoding the explanations produced by Shapley additive explanations (SHAP). Their method has shown good performance in predicting attacks. However, the XGBoost model lags significantly behind other tree models, including LightGBM. Wang et al. [28] proposed a framework that gives local and global explanations to any IDS. This framework presented the first application of SHAP method [30] to improve the transparency of IDSs, and ultimately help build cyber users' trust in the IDSs. This method effectively analyzes the contributions of each feature in the model, thereby assisting in a better understanding of the prediction made by the model. SHAP is based on the concept of Shapley values from game theory, a concept widely recognized and used for fair allocation of cooperative game profits. The core idea of Shapley values is that the contribution of each feature to the prediction outcome should equal the average change in the prediction outcome when it is added to the model. From previous research, we found that SHAP can explain deep learning models to generate a SHAP value for each feature, which helps researchers analyze and defend against attacks. The resulting interpretation can reflect

the traffic characteristics well, which makes it possible to detect by reconstructing the interpretation. Unfortunately, these studies only use explainable methods such as SHAP to explain the black box model to improve the explainability of the model. However, these explainable methods are not used to cooperate with popular deep learning methods to further improve the performance of network intrusion detection. At the same time, these methods are not very capable of detecting a large number of new attacks. Therefore, how to make the interpretability method improve the interpretability of the deep learning model while further improving the model's prediction ability of new attacks is worth exploring.

In this paper, we propose a novel hybrid explainable neural network framework termed HEN for robust network intrusion detection. This framework is composed of three parts: explanation generation, explanation reconstruction, and anomaly detection. In the first phase, we enhance network intrusion data using synthetic minority over-sampling technique (SMOTE), after which we train a light gradient boosting machine model [31] on this data and generate model explanations using the SHAP method. The generated explanations maintain the same dimensionality as the original data. In the second phase, to detect all normal flows in the test stream, including old attacks present in the test set and new attacks not present in the test set, we extract normal flow explanations from the outputs of the first phase. We construct an AE-LSTM model [32] that employs LSTM networks to encode and decode the data. We train the AE-LSTM model using the extracted explanations. Additionally, we apply supervised contrastive learning to bring similar data samples closer in the learned feature space and separate dissimilar ones, aiding in the acquisition of more informative and discriminative representations. In the third phase, we use the trained AE-LSTM model to compute the reconstruction error of the explanations for the normal training flow and the test flow. We define a threshold using the reconstruction error of the explanations for the normal flow, treating test flows with a reconstruction error exceeding this threshold as anomalous traffic. Experiments demonstrate that our proposed framework outperforms state-of-the-art methods in terms of performance and interpretability.

The key contributions of our work are as follows:

• We introduce SHAP, an explainable artificial intelligence method, to explain our intrusion detection framework, rendering it highly interpretable.

• In our study, we propose the AE-LSTM network to reconstruct SHAP values for the first time and combine the good reconstruction function of AE-LSTM and the advantage that SHAP can well reflect the traffic features, which significantly improves the performance of our framework.

• We integrate supervised contrastive learning into our framework, which assists in enhancing the frameworks discriminatory capacity and robustness. This approach maintains effective summarization even when the quality of the original input is poor.

Our network intrusion detection system provides new approaches to protect network security, prevent data breaches, minimize system interruptions and losses, and ensure compliance and regulatory requirements are met. After the model has worked, our classification results based on our framework will be submitted to the network security personnel for further analysis. The traditional model requires a lot of human resources to further analyze the abnormal traffic. Because our framework is based on an explainable method, it can help network security personnel to better analyze network traffic and deal with attacks. Therefore, in addition to reducing the harm caused by the high volume of traffic in the 5G network era, our framework can also reduce a lot of human resources.

The rest of this paper is organized as follows. Section 2 presents some of the related work and studies concerning our proposed framework, along with a multitude of work related to network intrusion detection using machine learning methodologies. Section 3 describes our proposed explainable neural network-based framework in detail and introduces the background knowledge we use, including the LightGBM algorithm, SHAP method, LSTM Network, and supervised contrastive learning. In Section 4, we describe the datasets we use and part of the experimental setup. We analyze the experimental results and the reasons for its formation. In addition, we further deepen the explanations through the SHAP plots. In Section 5, we conclude this paper.

2 Related work

2.1 Previous studies

NIDS faces considerable challenges in terms of improving their accuracy and optimization due to the imbalanced nature of traffic data, lack of interpretability, and inadequate feature extraction capabilities. With the evolution of the internet, a substantial number of studies on network intrusion detection utilizing various deep–learning methodologies have been published. The following summarizes relevant research on machine learning and deep learning algorithms for network intrusion detection.

When machine learning was initially introduced for intrusion detection, researchers employed classification algorithms such as decision trees [33], SVM [34], K-nearest neighbors [35], and Bayesian classifiers [36] for intrusion detection tasks. However, the detection performance of these methods often proved unsatisfactory. SVM classification can detect a set of hyperplanes as delimiters in a high-dimensional space. However, the problem of such an SVM based on IDS is finding appropriate kernel functions and the high time complexity of the learning phase. In contrast, Bayes classifiers utilize Bayes' rules to perform output categories. Furthermore, association rules are utilized to separate normal or anomaly flow. In naive Bayes models, features are assumed to be conditional independent. Although experiments have proved its good performance, this assumption does not hold in practice.

So many researchers have made relevant investigations and studies, using convolutional neural network (CNN) [37], recurrent neural network (RNN) [38], autoencoder [39, 40], and other deep learning methods to carry out a certain network intrusion system. Most of these methods perform superb detection accuracy on the detection of anomaly flow [41].

However, in the field of intrusion detection, there are still some problems, especially with the transparency of the systems. These methods above generally lack interpretability and fail to provide any information about the reasons behind decisions, making the predictions of these models even more difficult to understand, which makes it difficult for cybersecurity workers to resolve problems in the network through decision results.

2.2 Gradient boosting decision tree

In the domain of network intrusion detection tasks, there exists a high level of complexity due to the abundance of diverse features present in network flows and the non-linear relationships that exist among these features. Gradient boosting decision trees (GBDT) have demonstrated significant efficacy in handling such classification tasks. This approach can be seamlessly integrated with the SHAP method for the interpretation of the model. The prominent algorithms for GBDT implementation include XGBoost [42], LightGBM [31], and CatBoost [43].

XGBoost is an implementation of the gradient boosting algorithm, optimizing loss functions through gradient descent. In each iterative step, a new model is trained with the aim of minimizing the predictive error of the preceding model. The primary distinguishing feature between XGBoost and traditional gradient boosting methods lies in the application of advanced regularization techniques, coupled with the ease of parallelization across resources.

CatBoost is a machine learning algorithm based on gradient-boosted decision trees. It has demonstrated exceptional performance in numerous competitions and practical problems, notably in classification and regression tasks. Key features of CatBoost include ordered target statistics, feature combinations based on a greedy strategy, and ordered boosting. These features enable CatBoost to effectively manage a high number of categorical features, prevent overfitting of the model, and accelerate model training.

LightGBM serves as another variant of the GBDT algorithm. Despite the proven efficacy of XGBoost and CatBoost across a multitude of challenges, they can encounter issues related to efficiency and scalability when dealing with high feature dimensions and large volumes of data. To tackle this challenge, LightGBM introduces three distinctive methods: gradient-based one-side sampling (GOSS), exclusive feature bundling (EFB), and histogram algorithm. The histogram algorithm, as shown in Figure 1 first determines how many boxes are needed for each feature and assigns an integer to each box. Then the range of float numbers is divided into several intervals; the number of intervals is equal to the number of boxes, and the sample value belonging to the box is updated to the value of the box. As demonstrated by empirical results, LightGBM can significantly reduce training time compared to conventional GBDT algorithms while maintaining similar levels of predictive performance.

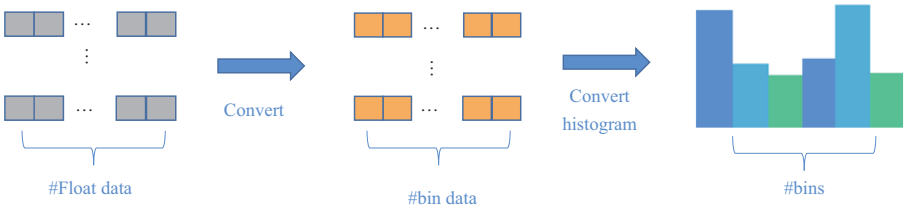

Figure 1 (Color online) Process of constructing a histogram from floating features.

Table 1 Summary of performance comparison between SHAP and LIME for all six performance metrics. Overall, SHAP provides the best performance compared to LIME for the six metrics analyzed in this work

Metric	Descriptive accuracy	Sparsity	Stability	Completeness	Robustness	Efficiency
SHAP	✓	✓	✓	✓	✓	✓
LIME			✓		✓	✓

2.3 Explainable artificial intelligence

Currently, while ML-based NIDS demonstrates remarkable performance on synthetic datasets, they remain internally complex and invisible 'black boxes' [44]. The inherent complexity of these ML models poses challenges in deciphering the reasoning behind classification predictions. Thus, in highly sensitive domains such as network intrusion detection, the question of whether to trust decisions driven by machine learning becomes a critical issue for the evolution of ML models. To address this, explainable artificial intelligence (XAI) methods are increasingly employed in contemporary systems to explain and validate decisions made by ML models [45].

Interpretable ML outputs can foster maintenance and troubleshooting of ML-based NIDS deployments by providing insights into the factors influencing model decisions and enabling alterations. Common global explainable methods are SHAP [30] and LIME. Based on previous studies, we compared the two methods in Table 1 with respect to the six XAI evaluation metrics. The results of the comparison provide a way for us to choose the XAI method. SHAP assigns quantifiable metrics to each data feature, indicating its contribution to and impact on model decisions. This facilitates the identification of key features used in model predictions and promotes a comprehensive understanding of the machine learning decision process. This rigorous ML model interpretability method holds the potential to establish trust and reliability within these systems, especially for sensitive applications such as NIDS.

2.4 Auto-endocer for anomaly detection

Auto-endocer. Auto-encoders (AE) are a form of deep neural networks designed to learn compressed representations of input data. The core concept lies in attempting to capture the significant features of input data with reduced information, or encoding, which can then be utilized to reconstruct the original input. AEs are particularly applicable in network intrusion detection due to their inherent ability to reconstruct normal network traffic data through training, thereby learning the characteristics of such traffic. During the detection phase, if a certain piece of network traffic data cannot be effectively reconstructed by the autoencoder, it could be inferred as anomalous network traffic.

Long short-term memory, a unique type of recurrent neural network, was first introduced by Hochreiter and Schmidhuber [46]. LSTM was primarily designed to address the challenges of vanishing or exploding gradients that conventional RNNs often encounter while handling long sequences. Network traffic data often resemble time-series data, characterized by notable temporal dependencies. Integrating LSTM with AEs can efficaciously capture these temporal dependencies, enabling the detection of anomalies based on time patterns and thereby enhancing the model's capacity to counter various traffic attacks.

Supervised contrastive learning. In the anomaly detection task, the normal sample and the abnormal samples have some similar features. The similarity of these features may cause the auto-encoder to be insensitive to the anomaly samples. Supervised contrastive learning is a commonly used technology to alleviate this issue. Supervised contrastive learning is an approach proposed by Khosla et al. [47], employed in machine learning for representation learning and discriminative feature extraction. It is an extension of the contrastive learning method that incorporates class labels or annotations during training to improve the model's ability to differentiate between different classes.

The contrastive learning framework aims to bring similar data samples closer in the learned feature space while pushing dissimilar samples apart. It relies on a network architecture, where two identical neural networks share weights and process two different augmented versions of the same input data. The network's objective is to minimize the distance between positive pairs while maximizing the distance between negative pairs.

In the supervised contrastive learning setting, the method leverages class labels to form positive and negative pairs during training. Specifically, for each data sample, the algorithm selects another sample from the same class as the positive pair and a sample from a different class as the negative pair. By comparing the representations of these pairs, the model learns to distinguish between classes more effectively.

The contrastive loss function used in supervised contrastive learning can be formulated as a combination of the InfoNCE (normalized contrastive estimation) loss and cross-entropy loss. The InfoNCE loss encourages similarity between positive pairs, while the cross-entropy loss enforces proper class separation.

Supervised contrastive learning has shown promising results in various tasks, including image classification, speech recognition, and natural language processing. It helps to learn more informative and discriminative representations, contributing to improved performance in downstream tasks like classification and clustering.

3 Methodology

In this paper, we propose a novel hybrid explainable neural network-based framework, termed HEN, of which the main technical components consist of three stages: explanation generation, explanation reconstruction, and anomaly detection. (1) In the first stage, we first build a LightGBM model. Then we use the entire training set to train this model and use the test set to evaluate the LightGBM model as the first stage of detection results of our framework. The feature importance provided by traditional algorithms such as LightGBM only tells us which features are important, but it is not clear how each feature of each traffic affects the prediction results. SHAP is a model interpretation package developed in Python that can interpret the output of any machine learning model. Therefore, we use the SHAP method to build an interpreter and let this interpreter interpret the LightGBM model. The explanation appears in the form of SHAP values. Each feature of each flow will generate a SHAP value, so the latitude of the generated explanation is the same as that of the original data set. The biggest advantage of the SHAP value is that it can reflect the influence of the features in each sample, and it also shows the positive and negative effects, thus improving the interpretability of the model. (2) In the second stage, we build an autoencoder that encodes and decodes explanations using LSTM networks. Since there are not only unseen new attacks in the test set but also old attacks that have been seen during training. Therefore, to detect all these attacks, we only use the explanations generated by normal traffic in the training set as the input of AE-LSTM to train this network. (3) In the final stage of our framework, we use this trained network to reconstruct the explanations for normal traffic generation in the training set and all traffic generation in the test set. In general, whenever an AE-LSTM network encounters a new attack or new normal traffic, its reconstruction error will increase. We use the change in the reconstruction error phase as a metric for network intrusion detection. To achieve this, we set the threshold at the 86th percentile of RE values observed during the training phase. All flows that cause the reconstruction error to be higher than the threshold are then considered abnormal flows. Our proposed framework is illustrated in Figure 2.

3.1 Explanations generation

3.1.1 *SMOTE for imbalanced data*

As mentioned above, network intrusion data are the typical imbalanced data. This imbalance can cause biased model performance, where the model performs well on the majority class but poorly on the minority class, which is often the class of interest. In our task, the majority class consists of normal network activities, while the minority class comprises dozens of different types of network intrusion, with some of them accounting for less than 1% of the total data. Synthetic minority over-sampling technique (SMOTE) is a popular algorithm used to address the problem of imbalanced data in machine learning. The SMOTE algorithm works by generating synthetic samples for the minority class, effectively increasing

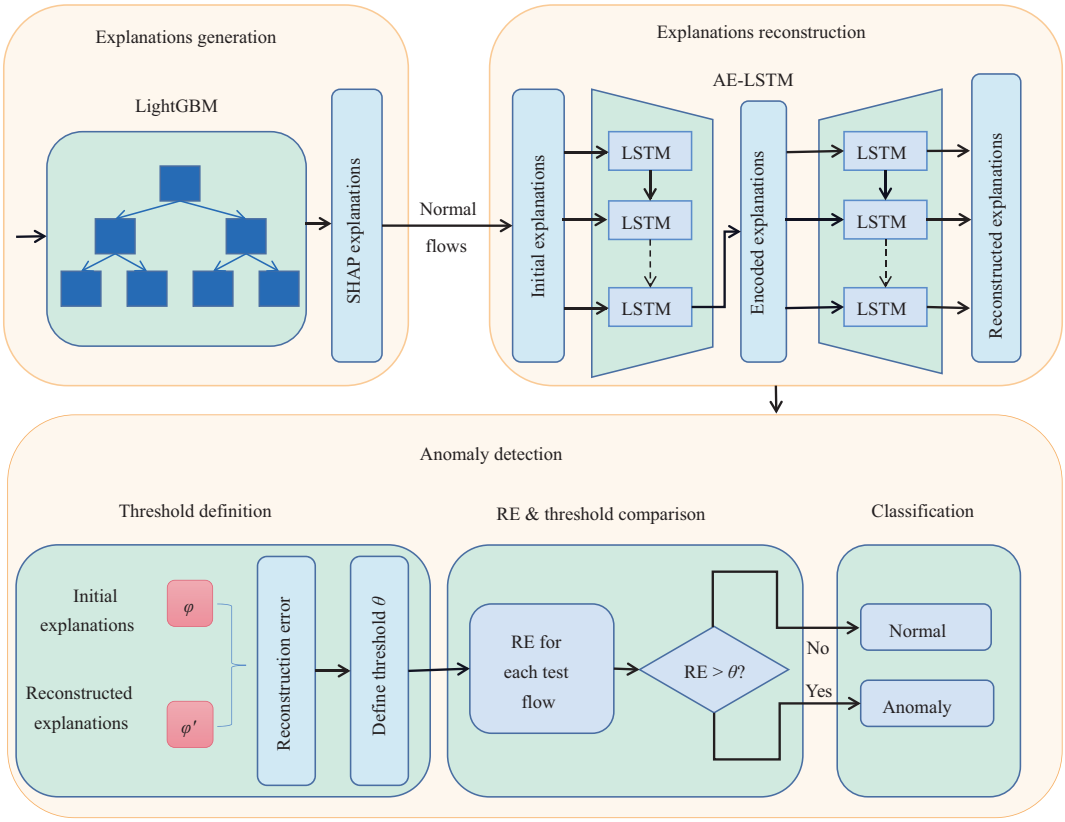

Figure 2 (Color online) Main technical components of the proposed framework consist of three parts: explanation generation, explanation reconstruction, and anomaly detection.

its size and balancing the dataset. This helps the model learn from the minority class and improves its ability to make accurate predictions. The synthetic samples are created by interpolating between existing minority class samples.

3.1.2 *LightGBM model*

Light gradient boosting machine (LightGBM) is a highly efficient, distributed gradient boosting framework grounded on decision tree algorithms and developed by Microsoft. This framework is exceptionally proficient in handling machine learning tasks such as classification, ranking, and regression. By deploying LightGBM, the efficiency of GBDT has been considerably enhanced.

GBDT employs classification and regression trees (CART) as base classifiers. The training process is initiated by continually reducing residuals to train base classifiers, followed by weighting these classifiers to construct a stronger and composite classifier. This implies that the original model remains unchanged with each iteration while new functions are incorporated into the model to ensure the predicted values continually approximate actual values as closely as possible.

The training objective function demonstrated as

$$\mathrm{Obj}^k = \sum_i L\left(y_i, y_i^k\right) + \Omega\left(f_k\right) + c^{k-1} = \sum_i L\left(y_i, y_i^{k-1} + f_k\left(x_i\right)\right) + \Omega\left(f_k\right) + c^{k-1} \tag{1}$$

is composed of the actual value of the label (y_i), the result of the k-th learning ($\hat{y}_i^k$), the sum of regularization terms of the first $k-1$ trees (c^{k-1}), and the predicted value of the new model to be added ($f_k(x_i)$). The regularization term of the model is represented by Ω. The purpose of the objective function is to identify an optimal tree, f_k, that minimizes the value of the function to the lowest possible level [48].

The objective function is expanded utilizing the Taylor series, represented as

$$f\left(x + \Delta x\right) = f\left(x\right) + f'\left(x\right)\Delta x + \frac{1}{2} f''\left(x\right)\Delta x^2. \tag{2}$$

The result of the second-order Taylor expansion for the loss function is shown as

$$\sum_i L\left(y_i, y_i^{k-1} + f_k(x_i)\right) = \sum_i \left[L\left(y_i, y_i^{k-1}\right) + L'\left(y_i, y_i^{k-1}\right) f_k\left(x_i\right) + \frac{1}{2} L''\left(y_i, y_i^{k-1}\right) f_k^2\left(x_i\right)\right]. \tag{3}$$

The first-order derivative of the objective function for the i-th sample is defined as g_i, while the second-order derivative of the objective function for the i-th sample is defined as h_i. These two variables are obtained using (4) and (5), respectively.

$$g_i = L'\left(y_i, y_i^{k-1}\right), \tag{4}$$

$$h_i = L''\left(y_i, y_i^{k-1}\right). \tag{5}$$

The simplified form of the objective function can be represented as

$$\mathrm{Obj}^k = \sum_i \left[L\left(y_i, y_i^{k-1}\right) + g_i f_k\left(x_i\right) + \frac{1}{2} h_i f_k^2\left(x_i\right) + \Omega\left(f_k\right) + c\right]. \tag{6}$$

As the GBDT algorithm adopts a pre-sorting iteration mechanism, it necessitates repeated access to the entire dataset. This drawback results in excessive time and space complexity. To mitigate these constraints, LightGBM applies an enhanced histogram technique to segment continuous floating-point feature values into k intervals. By selecting optimal split points within the k intervals, both the speed of training and efficiency in space consumption have been significantly improved.

In addition, the methodology implemented by LightGBM adopts a leaf-wise strategy as opposed to a level-wise strategy during the decision tree creation. This approach augments the maximum depth limit while ensuring superior efficiency, thereby mitigating overfitting and curtailing the training data volume. To deliver precise information gain estimates while dealing with a decreased data volume, gradient-based one-side sampling is applied. This technique maintains instances with larger gradients while executing random sampling on instances with smaller gradients. In an attempt to achieve dimensionality reduction without compromising information integrity, LightGBM utilizes the exclusive feature bundling method, which bundles mutually exclusive features at a predetermined conflict rate.

In our research, tasks related to network intrusion detection involve dealing with copious amounts of data and features, including IP addresses, port numbers, protocol types, and more. LightGBM has the ability to handle categorical features directly without the need for preprocessing such as one-hot encoding. This capability saves significant data processing time and potentially enhances the predictive performance of the model. Furthermore, LightGBM incorporates parameters for handling imbalanced data, which can enhance the model's accuracy in predicting attack traffic, thereby bolstering system security.

3.1.3 *Shapley additive explanation*

While LightGBM boasts exceptional predictive capabilities, it often functions as a "black box" model with internal workings that are difficult to interpret. For this purpose, we have integrated the SHAP method to interpret the model, enabling a clearer understanding of how each feature influences the prediction outcome.

SHAP is a model interpretation method designed to elucidate the extent to which a machine learning model depends on individual input features. SHAP is based on the concept of Shapley values from game theory, a concept widely recognized and used for fair allocation of cooperative game profits. The core idea of Shapley values is that the contribution of each feature to the prediction outcome should equal the average change in the prediction outcome when it is added to the model. This change is averaged over all possible feature orders, i.e., the order in which features are added to the model. Based on this concept, the SHAP method can assign a SHAP value to each feature, quantifying its contribution to the prediction outcome. The Shapley value is defined as follows:

$$\Phi(x_i) = \sum_{S \subseteq \{1,2,\ldots,k\} \setminus \{i\}} \frac{|S|!\,(K - |S| - 1)!}{K!} \left[f_x\left(S \cup \{i\}\right) - f_x\left(S\right)\right], \tag{7}$$

where K is the number of stakeholders. The contribution of entity i can be defined as a marginal contribution, i.e., the difference between the profit obtained by group S members only: $f_x\left(S\right)$ and that

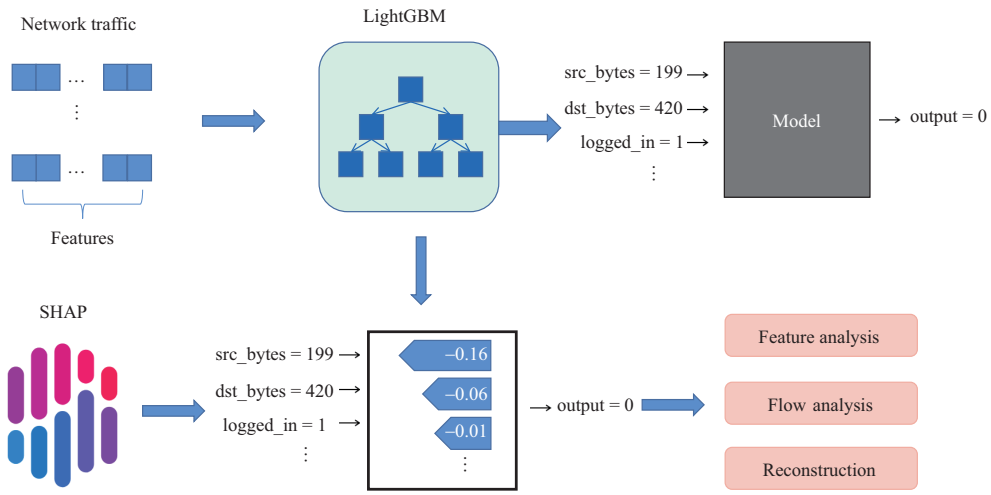

Figure 3 (Color online) Comparison of black-box models like LightGBM and explainable models produced by SHAP, and extensive uses of the explanations of features. Traditional black-box models only output prediction results based on network information. Our explainable framework also provides explanations in the form of SHAP value.

of both entity i and the group members: $f_x(S \cup \{i\})$. However, group members can change the gain which brings inconsistency. The Shapley value is the only profit allocation method that satisfies the following four properties: efficiency, symmetry, linearity, and null player. The sum of each feature i 's contribution $\phi_i(x_i^{(j)})$ is represented as the predicted outcome of flow j: $f(x^{(j)})$ through the use of Shapley value.

$$\phi_0 = E\left(f\left(x^{(j)}\right)\right) = \frac{1}{N}\sum_{j=1}^{N} f\left(x^{(j)}\right), \tag{8}$$

$$\phi_i\left({x_i}^{(j)}\right) = \Phi\left({x_i}^{(j)}\right) - \frac{1}{N}\sum_{k=1}^{N} \Phi\left({x_i}^{(k)}\right), \tag{9}$$

$$f\left(x^{(j)}\right) = \phi_0 + \sum_{i=1}^{K} \phi_i\left(x_i^{(j)}\right), \tag{10}$$

where N is the number of network flows. We derived $\forall i$, $E\left(\phi\left(X_i\right)\right) = \frac{1}{N}\sum_{j=1}^{N}\phi\left({x_i}^{(j)}\right) = 0$ from (9).

The SHAP value has been proved to be consistent [49] and is appropriate for all machine learning algorithms, including GLM. Moreover, a specific polynomial time algorithm was proposed for tree models like LightGBM, XGBoost, and CatBoost.

The SHAP method has proven to be a significant advancement in the field of machine learning model interpretation. SHAP amalgamates various existing methods to create a visually appealing, intuitive, and theoretically sound method for explaining predictions made by any model. In our work, we utilized the SHAP framework to explain the LightGBM model. Through an analysis of Shapley values and visualizations, we are better able to explain the characteristics of network intrusion traffic, thereby enabling more precise enhancements in NIDS.

In our framework, we explain the LightGBM model using the SHAP method. Figure 3 shows the difference between our explainable framework and traditional black-box models. Traditional models only tell us the predicted results based on the network information provided. In contrast, the explainable framework we propose also tells how the feature value affects the prediction result through the output SHAP value. From these explanations, feature analysis, network flow analysis, and explanation reconstruction are carried out in the next stage. However, in specific tasks, we still hope to obtain the relationship between each feature of the sample and the prediction result, especially for those samples where the model predicts errors. If it can be analyzed from the perspective of features and results, it will be helpful for improving the model effect, or analyzing unusual samples can be very helpful. Therefore, in our work, we use the SHAP method to interpret the LightGBM black-box model.

What is different from before is that when we pass the characteristics of one traffic to the interpretable model trained using network traffic for prediction; the model not only tells us the prediction results but

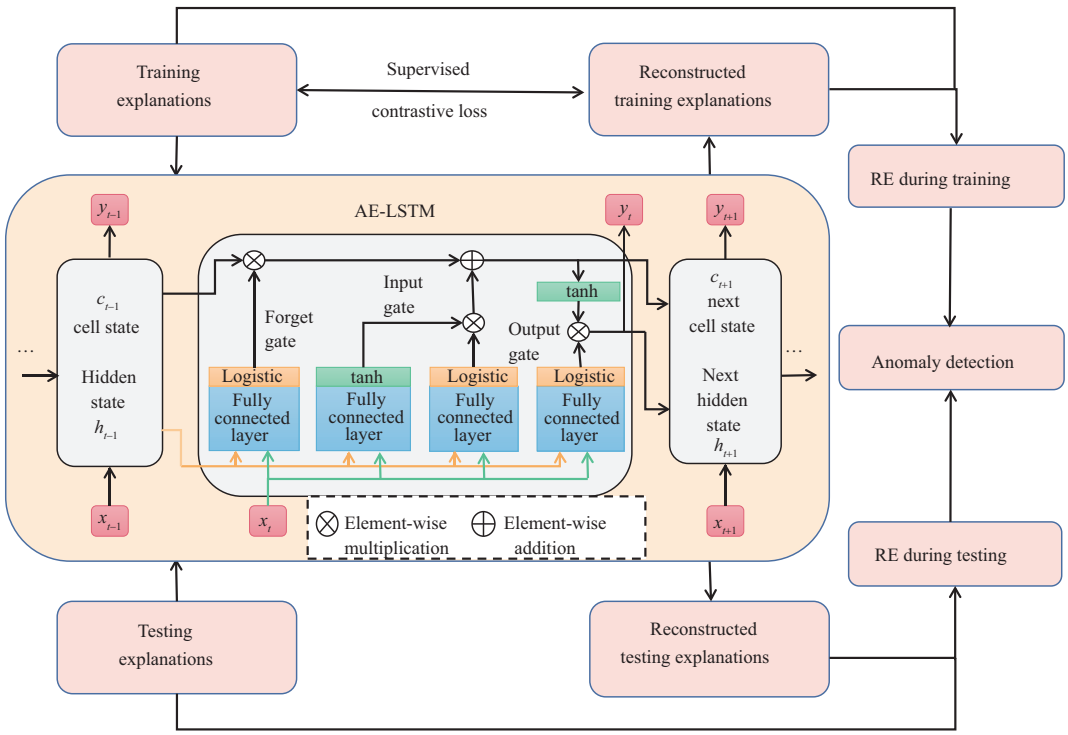

Figure 4 (Color online) Explanation reconstruction. Train AE-LSTM with training explanations (normal) and supervised contrastive loss, then using fitted AE-LSTM network reconstructs training explanations (normal) and testing explanations (all), respectively, and calculate absolute reconstruction error according to the explanations before and after reconstruction during training and testing, which are sent to the next stage for anomaly detection.

also tells us the magnitude and positive or negative impact of each feature on the prediction results. For example, the feature src_bytes has a negative impact of 0.16 on the prediction results, the feature dst_bytes has a negative impact of 0.06 on the prediction results, and the feature logged_in has a negative impact of 0.01 on the prediction results. We can then use these explanations to perform feature analysis, flow analysis, and explanation reconstruction, which are carried out in the next stage. Among them, feature analysis and traffic analysis can improve the interpretability of the model, and explanation reconstruction can enhance the model's ability to predict new attacks.

3.2 Explanation reconstruction

3.2.1 *AE-LSTM model*

In order to further enhance our framework's capacity to recognize novel traffic attacks, we employ an AE-LSTM model to learn and reconstruct the interpretations of normal traffic generated in the preceding step. The AE-LSTM model is an AE architecture based on LSTM [50], which learns efficient representations of data through unsupervised learning.

In deep learning, the AE is an unsupervised neural network model that learns latent information to encode the features of input data and uses the newly learned features to reconstruct the original input data [51], known as decoding. Autoencoders are trained by optimizing the reconstruction loss, i.e., the difference between the input and the output, which allows the model to learn low-dimensional representations of data and to reconstruct the original data as accurately as possible from these representations.

LSTM is a special type of RNN designed to address the problems traditional RNNs face when learning long-term dependencies, such as vanishing and exploding gradients. LSTM introduces a mechanism called "gates" to resolve this issue. The key idea of LSTM is its cell state, akin to a conveyor belt running directly along the chain with only minor linear interactions. It is relatively easy for information to remain unchanged as it passes along this belt. LSTM possesses the capability to delete or add information to the cell state, controlled by structures called gates, including the input gate, forget gate, and output gate, as depicted in the LSTM block in Figure 4. The computation formula for each variable in LSTM is outlined

below:

$$\begin{aligned} f_t &= \sigma\left(W_f \cdot [x_t, h_{t-1}] + b_f\right), \\ i_t &= \sigma\left(W_i \cdot [x_t, h_{t-1}] + b_i\right), \\ c'_t &= \tanh\left(W_c \cdot [x_t, h_{t-1}] + b_c\right), \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot c'_t, \\ o_t &= \sigma\left(W_o \cdot [x_t, h_{t-1}] + b_o\right), \\ h_t &= o_t \cdot \tanh\left(c_t\right), \end{aligned} \tag{11}$$

where x_t is the network input, h_t is the output of the hidden state, h_{t-1} is the hidden state from the last hidden layer, σ denotes the sigmoid activation function, c_{t-1} is the old cell state, c_t is the new cell state, and the state candidate values are defined using c'_t. f_t, i_t, and o_t are the results of the forget gate, input gate, and output gate. W_f, W_i, W_o, and W_c are the weights used on the forget gate, input gate, output gate, and memory cell. Similarly, b_f, b_i, b_o, and b_c are the bias used on the forget gate, input gate, output gate, and memory cell.

In this study, we use an LSTM network as the encoder and decoder of the AE, forming the AE-LSTM. Figure 4 illustrates our explanation-based AE-LSTM network and explanation reconstruction through this network. The middle section of Figure 4 represents the architecture of the LSTM network. In the explanation reconstruction phase, we begin by extracting explanations from normal training flows to train the AE-LSTM network. During the training and testing stages, we input the "Training explanations" of the training flows and the "Testing explanations" of the test flows into the AE-LSTM network, respectively. This yields the "Reconstructed training explanations" for the reconstructed training flows' explanations, as well as the "Reconstructed testing explanations" for the reconstructed test flows' explanations. The reconstruction process involves the forward propagation of the AE-LSTM network. It starts by reducing the dimensionality of the explanation and then expanding it back using LSTM, resulting in a reconstructed explanation of the same size as the original. Subsequently, we calculate the mean absolute error (MAE) between the "Training explanations" and the "Reconstructed training explanations" to quantify the reconstruction error for the training flow. Similarly, we compute the mean absolute error between the "Testing explanations" and the "Reconstructed testing explanations" for the test flow. These two reconstruction errors are then forwarded to the subsequent stage for anomaly detection.

3.2.2 *Supervised contrastive learning*

The AE-LSTM heavily relies on the quality and discriminative nature of the input data to learn meaningful representations and patterns. When the input lacks the ability to clearly differentiate between normal activities and network intrusions, the model may struggle to accurately detect and classify network intrusions, leading to suboptimal results.

To avoid the above issue, we introduce supervised contrastive learning into our approach. This innovative technique enhances the model's ability to learn discriminative features from the data by creating distinct representations for instances belonging to different classes. By leveraging the labeled data to form meaningful clusters, supervised contrastive learning enables the model to better capture the inherent patterns and variations that characterize each class. In our proposed model, the initial explanations are passed to the supervised contrastive model to generate the discriminative features. These discriminative features of normal network activities are then used in training AE-LSTM to detect network intrusions. The generated features of normal network activities and network intrusions exhibit a significantly more pronounced dissimilarity. Consequently, our AE-LSTM model becomes more sensitive to network intrusions. The supervised contrastive loss function is described as follows:

$$\mathcal{L}_{\text{supcon}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_p / \tau\right)}{\sum_{a \in A(i)} \exp\left(\boldsymbol{z}_i \cdot \boldsymbol{z}_a / \tau\right)}, \tag{12}$$

where $\boldsymbol{z}_i$ denotes the feature representation of the i-th sample, τ is the temperature parameter, $\boldsymbol{z}_a$ is the feature representation for the samples a belonging to class A. $P(i)$ is the set of indices of all positives in the batch distinct from i, and $|P(i)|$ is the corresponding cardinality.

3.3 Anomaly detection

The part of anomaly detection in Figure 2 shows our specific steps for explanations-based anomaly detection. In the previous part, we fit the AE-LSTM network by reconstructing the normal flow and then calculate the absolute reconstruction error (RE) of the normal flows. The formal expression is as follows:

$$\mathrm{RE}\left(\phi, \phi'\right) = \frac{\sum_{i=1}^{N} |\phi_i - \phi_i'|}{N}, \tag{13}$$

where ϕ represents the original explanations, ϕ' represents explanations reconstructed, which have similar dimensions to the original explanations.

We anticipate that the RE of the AE-LSTM network will significantly increase each time it encounters both old attack flows and new ones. We use the changes in RE as an indicator for attack detection in network intrusion scenarios. To achieve this, we set a threshold at the 86th percentile of RE values observed during the training phase. This percentile-based threshold technique operates on the assumption that normal flows typically result in RE values within a normal range, that is, below this threshold, while attack flows generate REs exceeding this limit.

Adopting this approach, any network flow that generates an RE above the specified threshold is classified as an attack flow. This method offers a balanced trade-off between precision and recall, minimizing false positives to the greatest extent possible while maximizing the true positive rate. Consequently, we anticipate that our proposed model will exhibit enhanced performance and reliability in the detection of network traffic anomalies.

4 Experiment

4.1 Dataset

We conduct experiments on one dataset called the NSL-KDD dataset, an improved version of the KDD'99 dataset, proposed by Tavallaee et al. [52]. It was initially released by DARPA, marking it as the first publicly available dataset containing extensive real-world intrusion attacks on military networks. It consists of four subsets: KDDTrain+, KDDTest+, KDDtest-21, and KDDTrain+_20Percent.

The NSL-KDD dataset consists of 125973 training flow and 22544 testing flow and comprises 43 features, out of which 41 are network-related features exported from TCP/IP dumps. The final two features are labels (classification of incoming traffic) and scores (severity of the incoming traffic). Four types of attacks are present in the dataset: denial of service, probe, user to root, and remote to local. These are further subdivided into 39 attack subclasses. The training set of the NSL-KDD dataset does not include redundant records, thus ensuring classifiers are not biased toward more frequent records. Similarly, the test set of the NSL-KDD dataset does not contain duplicate records, enhancing the precision of detection rates.

Despite the NSL-KDD dataset having some limitations in terms of capturing samples of zero-day attack types, it offers one of the few publicly available intrusion datasets for evaluating the performance of trained and tested NIDS. In this study, we employ the 'KDDTrain+' for training and the 'KDDTest+' for evaluating our framework.

4.2 Evaluation metrics

We assess the performance in our experiments by calculating the error between the predictions made by our proposed framework and the actual results. The metrics we used include accuracy, recall, and precision, formulated as follows:

$$\begin{aligned} \text{accuracy} &= \frac{\mathrm{TN} + \mathrm{TP}}{\mathrm{TN} + \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}, \\ \text{recall} &= \frac{\mathrm{TP}}{\mathrm{FN} + \mathrm{TP}}, \\ \text{precision} &= \frac{\mathrm{TP}}{\mathrm{FP} + \mathrm{TP}}, \end{aligned} \tag{14}$$

Table 2 Performance of our proposed framework against LightGBM and AE-LSTM based on raw testing flow

Method	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)
LightGBM	78.40	64.19	96.80	77.19
AE-LSTM	86.07	83.14	91.61	87.17
Proposed framework	92.95	95.26	92.57	93.90

where TP represents true positive, FP represents false positive, TN represents true negative, and FN represents false negative. Generally, high precision and recall are both required, which can be represented by a fused metric F-measure,

$$F_\beta = \frac{(1+\beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}, \tag{15}$$

where β^2 is set as 1 to give the precision the same weight as recall here.

4.3 Implementation details

Under the circumstances of our experiment, to ensure that we take the features of the digital format as input met during the training and testing of our framework, we utilize the 'LabelEncoder' method in the ScikitLearn [53] library to apply label encoding to all non-numerical features transmitted to the model. Then we change all features into the range $(-1, 1)$ through ScikitLearn. Since we only consider binary classification in the present work, we combine all abnormal labels in the NSL-KDD dataset into a single attack class. We use values 0 and 1 to represent labels for normal traffic and labels for abnormal traffic, respectively.

We used Scikit-Learn's ML API to implement our LightGBM model. During the training phase of the LightGBM model, the following parameters were set: learning rate of 0.1, num_leaves of 31, num_iterations of 100, and other parameters kept as default. The raw flow features are transmitted into the LightGBM model and designed to implement classification, where the output of the model, $y \in [0, 1]$, is a value that indicates whether the network flow is malicious, where $y \geqslant 0.5$ represents malicious flow and $y < 0.5$ represents normal flow.

The SHAP method provides a number of explainers, such as 'GradientExplainer', 'Deepexplainer', and 'KernelExplainer'. While SHAP can explain the output of any machine learning model, we have chosen a high-speed exact 'TreeExplainer' for tree ensemble methods. After we get the 'TreeExplainer' for the LightGBM model, we just need to pass the raw traffic data to the 'TreeExplainer' to get the value of that data.

We implement an AE-LSTM network using the TensorFlow API. We split normal explanations into two subsets and use the data for about 80% of the normal explanations as the new training set and the remaining 20% as the validation set. For our AE-LSTM network, we use two LSTM layers, for each layer, 24 and 41 neurons, respectively. We are also applying a 0.2 strength dropout at the first layer. We use Adam optimizer and a Contrastive Learning loss. The following parameters were set: learning rate of 0.03, epochs of 100, batch_size of 512, and shuffle of True.

4.4 Results

Table 2 summarizes the performance of the LightGBM model, AE-LSTM based on the raw test set, and our framework of AE-LSTM based on explanations. The LightGBM model achieves a reasonable accuracy of 78.40% against general attack, with a high precision of 96.8% but a low recall of 64.19%. In addition, we also tested the performance of LightBGM against old attacks that appeared during training and new attacks that only appeared during testing. We found that LightGBM can detect 75.21% old attacks but only 41.76% new ones, which reveals the weak performance of LightGBM against zero-day attacks.

On the other hand, our AE-LSTM based on the raw test set achieves a relatively low recall of 83.14% but a high accuracy of 86.07%, and a high precision of 91.61%. Remarkably, our proposed framework performs strongly across all three metrics, achieving an overall accuracy of 92.95%, precision of 95.26%, and recall of 92.57%. The experimental results show that our method of intrusion detection using the AE-LSTM network can be used together with other machine learning classification methods to greatly improve the performance of intrusion detection by reconstructing interpretation.

Table 3 Overall performance of our framework against various state-of-the-art studies[a)]

Method	Accuracy (%)	Recall (%)	Precision (%)	F1-score (%)	XAI
Wang et al. [28]	80.60	80.60	82.80	81.69	✓
Ieracitano et al. [54]	84.21	80.37	87.00	83.55	×
Yang et al. [20]	89.36	84.86	95.98	90.09	×
Xu et al. [40]	90.61	98.43	86.83	92.26	×
Marino et al. [27]	95.50	–	–	–	✓
Deng et al. [55]	91.25	92.57	92.92	92.74	✓
Our framework	**92.95**	**95.26**	**92.57**	**93.90**	✓

a) bold text represents our result.

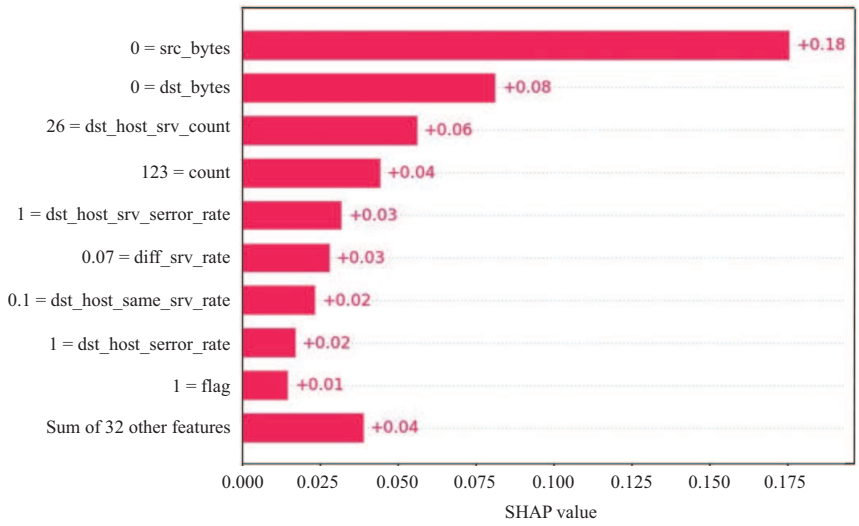

Figure 5 (Color online) SHAP bar plot of an anomaly flow, showing how each feature impacts predicted result.

Table 3 compares the performance of our framework against various state-of-the-art studies from the literature. We can see from Table 3 that we achieved an accuracy of 92.95%, a recall of 95.26%, a precision of 92.57%, and an F1-score of 93.90%. Our framework is able to surpass all benchmarks in the case of its F1-score, but taking into account recall and precision are slightly behind compared to [40] and [18], respectively. On the whole, our framework has achieved the best detection performance than other recent studies, especially explainable methods like [22]. In addition, we also added a novel teacher-student (TS) model [54] consisting of a teacher encoder and a student decoder to perform intrusion detection, which proved to be slightly worse than our framework.

4.5 Explanations

In our framework, to visualize the interpretability of the model, we rely on the bar plot to visualize the impact of each feature of a sample on the prediction result and the scatter plot to visualize the relationship of a feature to the predicted results of all samples.

SHAP explainer can give a baseline value of the model. For all features of each predicted sample, the model will generate a SHAP value, which indicates the influence of the feature on the sample prediction based on the baseline value. In the SHAP bar plot, features that make the prediction higher than the base value are shown in red, and features that make the prediction lower than the baseline are shown in blue. The longer the arrow corresponds to the feature, the greater the influence of the feature on the output.

For example, the bar plot illustrated in Figure 5, corresponds to an explanation of a training flow previously considered to be malicious by our model, whose features have a positive impact on the result of our framework on the whole. From Figure 5, we can see the raw value and SHAP value of all features considered by the model. Especially, the 'src_bytes' feature, a value of 0 bytes in this flow, has raised the predicted result by 0.18%, and the 'dst_bytes', a value of 0 bytes in this flow, has raised the predicted result by 0.08%, which shows that these two features have a large and positive impact on the prediction

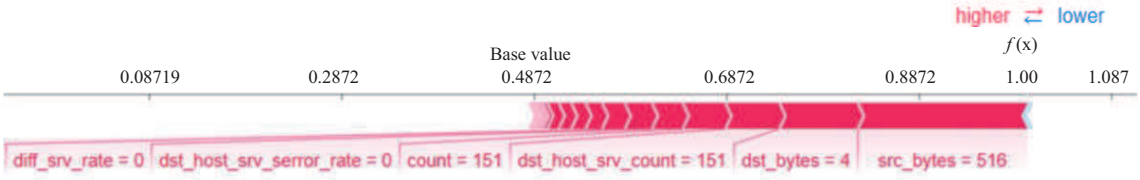

Figure 6 (Color online) SHAP force plot, showing us the process by which the baseline value is pulled towards the predicted result by the various features.

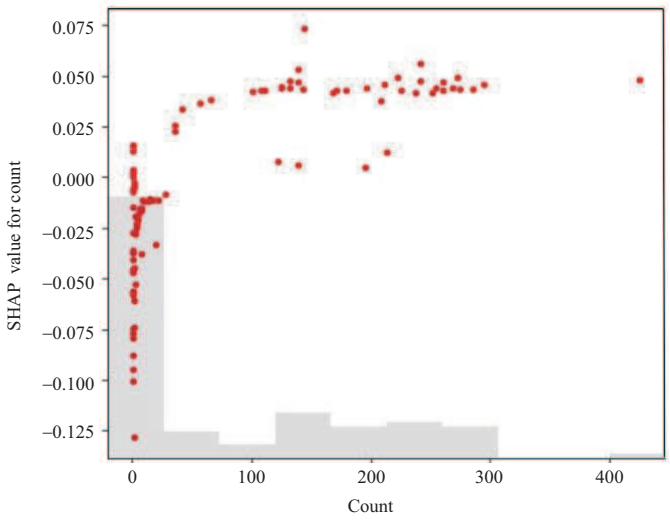

Figure 7 (Color online) SHAP scatter plot of feature count, showing the relationship between SHAP values of that feature and the values of the feature for all the examples.

results. In addition, we see that other features, such as the features 'dst_host_serror_rate' and 'flag' also have small but positive impacts on the result of our framework, while the 32 other features have an overall positive but subtle impact on the result of our framework.

Similar to the bar plot, the force plot shown in Figure 6 can also show the value of each feature of a traffic flow, the magnitude of the influence on the decision result, and the positive and negative. The difference is that we can more intuitively observe the base value, a value of 0.4872 in this example, and the process by which the baseline value is pulled towards the predicted result by the various features.

To understand how a single feature affects the output of the model, we can make a scatter plot that reveals the relationship between the SHAP values of that feature and the values of the feature for all the examples in a dataset. Figure 7 shows the distribution of the raw data of feature 'count' and SHAP values for 'count'. From Figure 7, we can find that when the value of count is relatively small, the value of SHAP tends to be negative, which means 'count' has a negative impact on the prediction result. On the contrary, when the value of 'count' is relatively large, the value of SHAP tends to be positive, which means 'count' has a positive impact on the prediction result.

In addition, beeswarm plots can also reveal the impact of individual features on prediction results. From Figure 8, we can easily see that when the value of feature 'dst_host_srv_count' is greater than 0, it often has a negative impact on the prediction results, making the traffic more likely to be predicted as normal traffic. On the contrary, when the value of feature 'dst_host_srv_serror_rate' is greater than 0, it tends to have a positive impact on the prediction results, making the traffic more likely to be predicted as an attack. Therefore, after a network is predicted, network personnel can choose to start with these regular features for analysis.

4.6 Analysis

Gradient boosting decision tree models, like LightGBM, XGBoost, and CatBoost are often the object of interpretation. We use these three tree models to train the NSL-KDD data set respectively and pass the trained model into the SHAP explainer. The interpreter will generate a SHAP value for each feature based

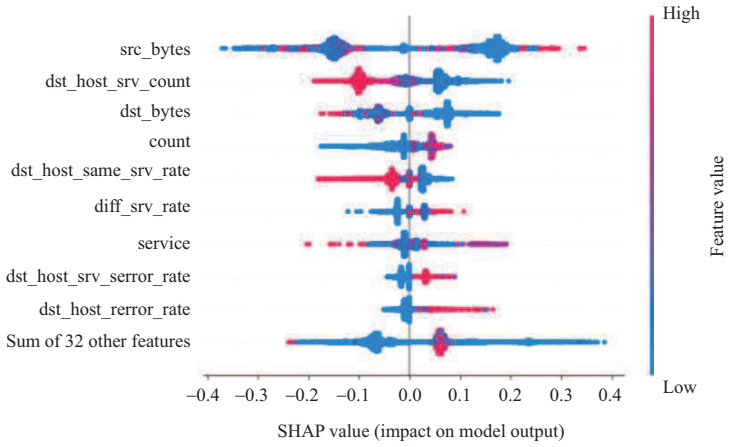

Figure 8 (Color online) SHAP beeswarm plot of feature count, showing us how the features affect the predictions in general.

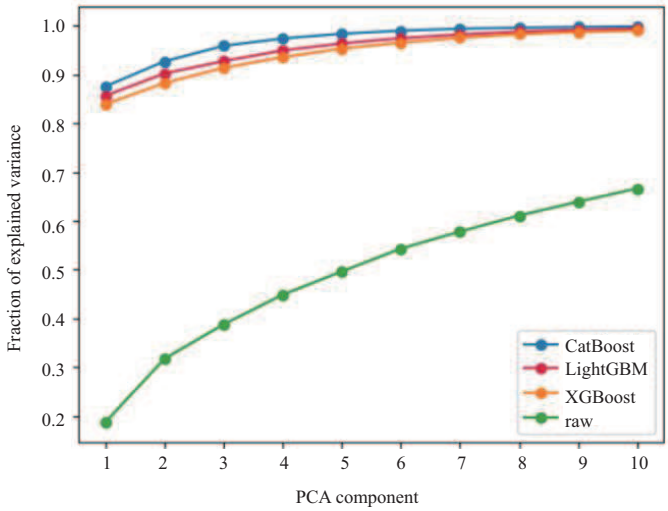

Figure 9 (Color online) Cumulative percentage of variance (PoV) explained by each PCA component of raw training data and explanations generated by CatBoost, LightGBM, XGBoost.

on the trained model and training data without a label, thus obtaining the interpretation generated by the three models of XGBoost, LightGBM, and CatBoost. In order to choose the best model to generate explanations, we consider both efficiency and performance. We apply principal component analysis (PCA) to the raw training data and explanations generated by CatBoost, LightGBM, and XGBoost. After reducing the dimensionality of the original data, we are left with ten features, so ten explainable variances are returned. The method that comes with PCA can check the percentage of the information amount of each new feature vector after dimensionality reduction to the total information amount of the original data. Accumulating these percentages is the cumulative percentage of variance (PoV). When we examine the cumulative percentage of variance explained by each PCA component, as shown in Figure 9, we find that 10 components of the raw training data only explain 66.62% of the data variance, compared to 99.35% in the case of LightGBM, 98.98% in the case of XGBoost, and 99.84% in the case of CatBoost. Because PoV is considered to be the standard of how much original information each PCA component can represent. Therefore, the higher the PoV of the explanation, the more information the explanation can compress and represent, which explains why the explanation data is more conducive to anomaly detection than the original data.

CatBoost has the best compression ability and LightGBM is slightly behind. But considering the efficiency of the three tree ensemble models, LightGBM has a huge lead. LightGBM reduces time consumption by 95.47% compared with XGBoost and 95.04% compared with CatBoost. Considering the above analysis, LightGBM is the most suitable model to be explained, which is proven by our experimental results.

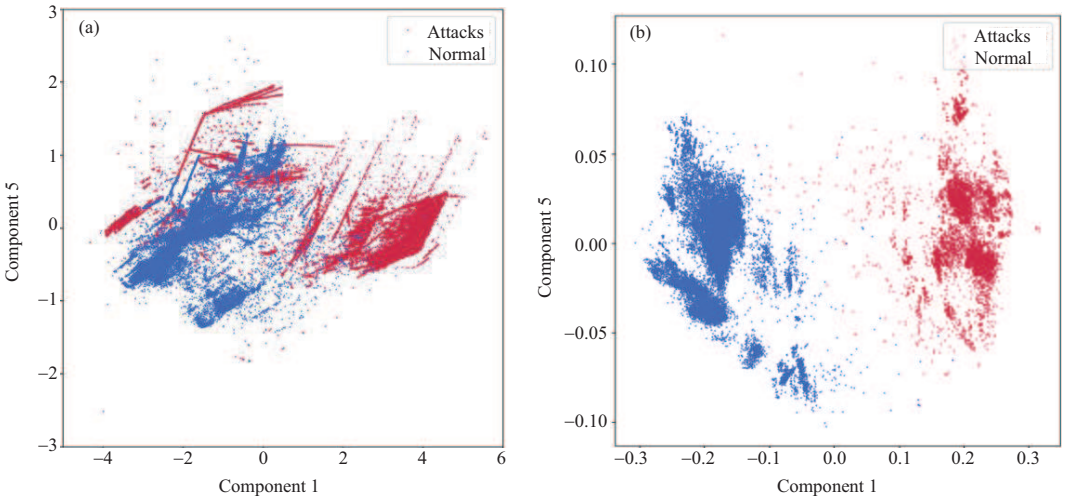

Figure 10 (Color online) PCA applied to the training data (a) and the explanations generated by LightGBM (b).

Additionally, to further figure out why our framework performs better, we plot the first and fifth PCA components of both the raw data and the interpretive data generated based on the LightGBM model as seen in Figure 10. The two components of the raw data show that there is a lot of overlap between normal and malicious flows, whereas interpreting the data distinguishes between normal and malicious flows. This difference indicates that the interpreted data can compress and reconstruct network features better than the raw data.

In our work, memory space and training time are not our primary considerations. The NSL-KDD dataset we used takes a long time to collect and is one-dimensional and small-scale, consisting of 125973 training flow and 22544 testing flow so it can be processed quickly. In our experiment, our AE-LSTM network processes each sample in just 0.00095 s.

Other models, for example, Karanam et al. [3] used Google Colaboratory as hardware with GPU acceleration. Their CNN-LSTM network processes each sample in just 0.00048 s and achieves an accuracy of 89.23%. Their model was trained for 100 periods. Our model training ended early after 60 epochs. Overall, our model lags slightly in efficiency but achieves a huge performance boost.

5 Conclusion

In this paper, we propose a novel hybrid explainable framework based on a neural network, termed HEN, for network intrusion detection. The framework mainly consists of three stages: explanation generation, explanation reconstruction, and anomaly detection. In short, we use the SHAP method to explain the LightGBM model, then utilize the AE-LSTM network to reconstruct the explanation and carry out anomaly detection through reconstruction errors. We use the NSL-KDD dataset to evaluate our proposed model. The LightGBM model is capable of detecting 75.21% of known attacks, but only 41.76% of new attacks. However, when combined with the SHAP method, its ability to detect new attacks is significantly enhanced. Our model achieved better performance over other methods with an accuracy of 92.65%, a recall of 95.26%, a precision of 92.57%, and an F1-score of 93.90%. Experimental results prove that our proposed framework is able to outperform numerous state-of-the-art studies in terms of several evaluation metrics, as well as improving interpretability through the SHAP method. However, there are still certain limitations in our work. For example, interpretability only explains gradient boosting decision trees, while the AE-LSTM network remains a black box model. We have not employed multiple interpretation methods to unravel the pros and cons of the interpretability model for the black box model. Additionally, we have not conducted training and testing on a wider range of datasets or more realistic network traffic data. We will test our framework on more real network data to improve the transferability, generalization, and practical significance of our proposed framework. Besides, we plan to figure out the differences in explanations generated by different models to enhance the performance of intrusion detection. We also intend to explore other XAI technologies, such as local interpretable model-agnostic explanations (LIME), to further improve the interpretability of deep learning models for anomaly detection.

Acknowledgements This work was supported in part by Fundamental Research Funds for the Central Universities (Grant No. x2wjD2230230), Natural Science Foundation of Guangdong Province of China, CCF-Phytium Fund, and Cultivation of Shenzhen Excellent Technological and Innovative Talents (Ph.D. Basic Research Started) (Grant No. RCBS20200714114943014), and Basic Research of Shenzhen Science and Technology Plan (Grant No. JCYJ20210324123802006).

References

1 Gupta D, Rani S, Ahmed S H, et al. Edge caching based on collaborative filtering for heterogeneous ICN-IoT applications. Sensors, 2021, 21: 5491
2 Rani S, Koundal D, Kavita D, et al. An optimized framework for WSN routing in the context of industry 4.0. Sensors, 2021, 21: 6474
3 Karanam L, Pattanaik K K, Aldmour R. Intrusion detection mechanism for large scale networks using CNN-LSTM. In: Proceedings of the 13th International Conference on Developments in eSystems Engineering (DeSE), 2020. 323–328
4 Fragkos G, Minwalla C, Plusquellic J, et al. Artificially intelligent electronic money. IEEE Consumer Electron Mag, 2021, 10: 81–89
5 Alibakhshikenari M, Virdee B, Mariyanayagam D, et al. Virtual antenna array for reduced energy per bit transmission at sub-5 GHz mobile wireless communication systems. Alexandria Eng J, 2023, 71: 439–450
6 Sehrai D A, Khan J, Abdullah M, et al. Design of high gain base station antenna array for mm-wave cellular communication systems. Sci Rep, 2023, 13: 4907
7 Wang D M, Zhang Y, Wei H, et al. An overview of transmission theory and techniques of large-scale antenna systems for 5G wireless communications. Sci China Inf Sci, 2016, 59: 081301
8 Muqdad Z S, Alibakhshikenari M, Elwi T A, et al. Photonic controlled metasurface for intelligent antenna beam steering applications including 6G mobile communication systems. AEU-Int J Electron Commun, 2023, 166: 154652
9 Din I, Alibakhshikenari M, Virdee B S, et al. High performance antenna system in MIMO configuration for 5G wireless communications over sub-6 GHz spectrum. Radio Sci, 2023, 58: 1–22
10 Din I, Alibakhshikenari M, Virdee B S, et al. Frequency-selective surface-based MIMO antenna array for 5G millimeter-wave applications. Sensors, 2023, 23: 7009
11 Ghadeer S H, Rahim S K A, Alibakhshikenari M, et al. An innovative fractal monopole MIMO antenna for modern 5G applications. AEU-Int J Electron Commun, 2023, 159: 154480
12 Sultana N, Chilamkurti N, Peng W, et al. Survey on SDN based network intrusion detection system using machine learning approaches. Peer-to-Peer Netw Appl, 2019, 12: 493–501
13 Yin C, Zhu Y, Fei J, et al. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 2017, 5: 21954–21961
14 Yao H, Fu D, Zhang P, et al. MSML: a novel multilevel semi-supervised machine learning framework for intrusion detection system. IEEE Internet Things J, 2018, 6: 1949–1959
15 Taher K A, Jisan B M Y, Rahman M M. Network intrusion detection using supervised machine learning technique with feature selection. In: Proceedings of International conference on robotics, electrical and signal processing techniques (ICREST), 2019. 643–646
16 Panigrahi R, Borah S, Bhoi A K, et al. Performance assessment of supervised classifiers for designing intrusion detection systems: a comprehensive review and recommendations for future research. Mathematics, 2021, 9: 690
17 Panigrahi R, Borah S, Bhoi A K, et al. A consolidated decision tree-based intrusion detection system for binary and multiclass imbalanced datasets. Mathematics, 2021, 9: 751
18 Shone N, Ngoc T N, Phai V D, et al. A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell, 2018, 2: 41–50
19 Khan F A, Gumaei A, Derhab A, et al. A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access, 2019, 7: 30373–30385
20 Yang Y, Zheng K, Wu C, et al. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors, 2019, 19: 2528
21 Ravi V, Chaganti R, Alazab M. Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system. Comput Electrical Eng, 2022, 102: 108156
22 Wang N, Chen Y, Xiao Y, et al. MANDA: on adversarial example detection for network intrusion detection system. IEEE Trans Dependable Secure Comput, 2023, 20: 1139–1153
23 Issa A A, Albayrak Z. DDoS attack intrusion detection system based on hybridization of CNN and LSTM. Acta Polytech Hung, 2023, 20: 105–123
24 Hnamte V, Hussain J. An extensive survey on intrusion detection systems: datasets and challenges for modern scenario. In: Proceedings of the 3rd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE), 2021
25 Goodall J R, Ragan E D, Steed C A, et al. Situ: identifying and explaining suspicious behavior in networks. IEEE Trans Visual Comput Graphics, 2018, 25: 204–214
26 Amarasinghe K, Manic M. Improving user trust on deep neural networks based intrusion detection systems. In: Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society, 2018. 3262–3268
27 Marino D L, Wickramasinghe C S, Manic M. An adversarial approach for explainable AI in intrusion detection systems. In: Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society, 2018. 3237–3243
28 Wang M, Zheng K, Yang Y, et al. An explainable machine learning framework for intrusion detection systems. IEEE Access, 2020, 8: 73127–73141
29 Barnard P, Marchetti N, DaSilva L A. Robust network intrusion detection through explainable artificial intelligence (XAI). IEEE Netw Lett, 2022, 4: 167–171
30 Lundberg S M, Erion G, Chen H, et al. From local explanations to global understanding with explainable AI for trees. Nat Mach Intell, 2020, 2: 56–67
31 Ke G, Meng Q, Finley T, et al. LightGBM: a highly efficient gradient boosting decision tree. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 30
32 Mahmoud M, Kasem M, Abdallah A, et al. AE-LSTM: autoencoder with LSTM-based intrusion detection in IoT. In: Proceedings of International Telecommunications Conference (ITC-Egypt), 2022. 1–6
33 Ahmim A, Maglaras L, Ferrag M A, et al. A novel hierarchical intrusion detection system based on decision tree and rules-based models. In: Proceedings of the 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2019. 228–233

34 Kabir E, Hu J, Wang H, et al. A novel statistical technique for intrusion detection systems. Future Generation Comput Syst, 2018, 79: 303–318
35 Serpen G, Aghaei E. Host-based misuse intrusion detection using PCA feature extraction and kNN classification algorithms. Intell Data Analysis, 2018, 22: 1101–1114
36 Zhang B, Liu Z, Jia Y, et al. Network intrusion detection method based on PCA and Bayes algorithm. Secur Commun Netw, 2018, 2018: 1–11
37 Wen T, Keyes R. Time series anomaly detection using convolutional neural networks and transfer learning. 2019. ArXiv:1905.13628
38 Sohi S M, Seifert J P, Ganji F. RNNIDS: enhancing network intrusion detection systems through deep learning. Comput Secur, 2021, 102: 102151
39 Yang Y, Zheng K, Wu B, et al. Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. IEEE Access, 2020, 8: 42169–42184
40 Xu W, Jang-Jaccard J, Singh A, et al. Improving performance of autoencoder-based network anomaly detection on NSL-KDD dataset. IEEE Access, 2021, 9: 140136–140146
41 Nisioti A, Mylonas A, Yoo P D, et al. From intrusion detection to attacker attribution: a comprehensive survey of unsupervised methods. IEEE Commun Surv Tutorials, 2018, 20: 3369–3388
42 Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. 785–794
43 Prokhorenkova L, Gusev G, Vorobev A, et al. CatBoost: unbiased boosting with categorical features. In: Proceedings of Advances in Neural Information Processing Systems, 2018. 31
44 McGovern A, Lagerquist R, Gagne D J, et al. Making the black box more transparent: understanding the physical implications of machine learning. Bull Am Meteorol Soc, 2019, 100: 2175–2199
45 Adadi A, Berrada M. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access, 2018, 6: 52138–52160
46 Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation, 1997, 9: 1735–1780
47 Khosla P, Teterwak P, Wang C, et al. Supervised contrastive learning. In: Proceedings of Advances in Neural Information Processing Systems, 2020. 33: 18661–18673
48 Gu T, Xu G L, Li W L, et al. Intelligent house price evaluation model based on ensemble LightGBM and Bayesian optimization strategy (in Chinese). J Comput Appl, 2020, 40: 2762–2767
49 Lundberg S M, Lee S I. A unified approach to interpreting model predictions. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 30
50 Yu Y, Si X, Hu C, et al. A review of recurrent neural networks: LSTM cells and network architectures. Neural Computation, 2019, 31: 1235–1270
51 Tschannen M, Bachem O, Lucic M. Recent advances in autoencoder-based representation learning. 2018. ArXiv:1812.05069
52 Tavallaee M, Bagheri E, Lu W, et al. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009. 1–6
53 Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in Python. J Mach Learn Res, 2011, 12: 2825–2830
54 Ieracitano C, Adeel A, Morabito F C, et al. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. Neurocomputing, 2020, 387: 51–62
55 Deng H, Li X. Anomaly detection via reverse distillation from one-class embedding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022. 9737–9746