IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 20, 2020, accepted April 13, 2020, date of publication April 16, 2020, date of current version April 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2988359

An Explainable Machine Learning Framework for Intrusion Detection Systems

MAONAN WANG[1], **KANGFENG ZHENG**[1], **YANQING YANG**[1,2], **AND XIUJUAN WANG**[3]

[1] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2] College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
[3] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

Corresponding author: Kangfeng Zheng (zkf_bupt@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802703, and in part by the National Natural Science Foundation of China under Grant 61602052.

ABSTRACT In recent years, machine learning-based intrusion detection systems (IDSs) have proven to be effective; especially, deep neural networks improve the detection rates of intrusion detection models. However, as models become more and more complex, people can hardly get the explanations behind their decisions. At the same time, most of the works about model interpretation focuses on other fields like computer vision, natural language processing, and biology. This leads to the fact that in practical use, cybersecurity experts can hardly optimize their decisions according to the judgments of the model. To solve these issues, a framework is proposed in this paper to give an explanation for IDSs. This framework uses SHapley Additive exPlanations (SHAP), and combines local and global explanations to improve the interpretation of IDSs. The local explanations give the reasons why the model makes certain decisions on the specific input. The global explanations give the important features extracted from IDSs, present the relationships between the feature values and different types of attacks. At the same time, the interpretations between two different classifiers, one-vs-all classifier and multiclass classifier, are compared. NSL-KDD dataset is used to test the feasibility of the framework. The framework proposed in this paper leads to improve the transparency of any IDS, and helps the cybersecurity staff have a better understanding of IDSs' judgments. Furthermore, the different interpretations between different kinds of classifiers can also help security experts better design the structures of the IDSs. More importantly, this work is unique in the intrusion detection field, presenting the first use of the SHAP method to give explanations for IDSs.

INDEX TERMS Intrusion detection system, Shapley value, SHapley Additive exPlanations, model interpretation, machine learning.

I. INTRODUCTION

With the enormous growth of cyber networks' usage and the vast applications running on it, network security is becoming increasingly important. To be specific, the cloud computing, 5G communication, and Internet of things (IoT) ushered in a vigorous development [1]–[3]. It is estimated that there will be a trillion physical devices connected to the Internet until 2022 [4]. However, these new technological developments have raised some security and privacy concerns. For example, Internet equipment with wide distribution and openness are ideal targets for cyber attacks. Moreover, as many Internet equipments are collecting and processing private information, they are becoming a goldmine of data for malicious attackers [5]. Therefore, intrusion detection systems (IDSs) have become essential tools in computer networks to provide a more secure network environment. The objective of an IDS is to detect misuse, unauthorized use, and abuse in the host's network [6], [7].

Over the last few years, many IDSs using different approaches have been proposed, developed, and evaluated. Some systems use shallow methods to detect intrusions. The classification algorithms like Decision tree [8]–[10], SVM [11]–[13], K-Nearest Neighbors [14], Bayes Classifier [15], etc., have been used for intrusion detection tasks. Some other systems utilize feature selection or ensemble approaches of classifiers to build intrusion detection models [16]–[18]. In recent years, more and more IDSs have started to use deep learning methods, such as deep neural network (DNN) [19], convolutional neural networks

The associate editor coordinating the review of this manuscript and approving it for publication was An-An Liu.

(CNNs) [20], recurrent neural network (RNN) [21], [22], variational autoencoder (VAE) [23]–[25], etc. Most of these methods exhibit excellent detection accuracy and low false positive rate on the detection of attacks [26].

However, there are still some problems, especially with the transparency of the systems, in the field of intrusion detection. Cybersecurity experts now usually make decisions based on the recommendations of an IDS; therefore, the predictions of the model should be understandable. Consequently, despite the impressive accuracies achieved by the aforementioned models, their increasing complexities are major drawbacks when humans are involved, as these models can not provide any information about the reasons behind their decisions, especially since DNNs are still used as black boxes. Therefore, it is imperative to provide some information about the reasons behind IDSs predictions, and provide cybersecurity personnel with some explanations about the detected intrusions. There are a small amount of works [27]–[29] to give the explanations to the results made by IDSs currently, but most of these works do not have an excellent theoretical foundation.

For the sake of solving these disadvantages and giving a better explanation to IDSs, a framework based on SHapley Additive exPlanations (SHAP) [30] is proposed in this paper. The SHAP has a solid theoretical foundation, and can be used for any model, whether it is a shallow model or a deep learning model. So this framework can give interpretations to any IDS. This framework provides local and global explanations which can improve the interpretability of any IDS. Local explanations used in this framework can give the details that each feature value that either increases or decreases the predicted probabilities. There are two kinds of global explanations in this framework. The first is to extract important features from any IDS, and the second can explore the relationships between the values of features and specific types of attacks. NSL-KDD dataset [31] is used to demonstrate the feasibility of the framework that is designed in this thesis.

The major contributions of this paper are as follows:

- We propose a framework that gives local and global explanations to any IDS. This framework contributes to a deeper understanding of the predictions made from IDSs, and ultimately help build cyber users' trust in the IDSs. At the same time, this framework also helps cybersecurity experts better understand the cyber attacks, such as knowing the typical characteristics of the specific attacks.
- This work is unique in the IDS field, as it presents the first application of SHAP method to improve the transparency of IDSs. Compared with other methods, SHAP has a better theoretical foundation.
- We explore the differences in interpretation between the one-vs-all classifier and the multiclass classifier. By comparing the interpretation of the same attacks by two types of classifiers, security experts can optimize the structures of the IDSs. Then the optimized structure can increase the human operators' trust in IDSs.

The rest of the paper is organized as follows. Section II describes the related works. The background in this experiment is introduced in section III, including Local interpretable model-agnostic explanations (LIME) [32], Shapley value [33], and SHAP [30]. Section IV proposes the framework that is used to improve the interpretability of any IDS and shows the details of how it works. Section V presents the experiments carried out using the NSL-KDD dataset and shows the detailed results. Finally, section VI concludes the paper and discusses the future direction.

II. RELATED WORK

Although there are some works related to SHAP in other areas, there are no previous reported works that use SHAP in IDSs. Not only the SHAP method, but also other methods of model interpretation are rarely used in the field of intrusion detection. The works in [34]–[36] just focuses on the model explanations for computer vision, and the works in [37], [38] use LIME to give the explanations in the fields of natural language processing and acoustic analysis. The works in [39], [40] use SHAP to improve the transparency of the models in the field of biology. Most of these works do not directly design or use for IDSs.

In the recent past, deep learning methods have shown state-of-the-art performances in a multitude of fields, such as computer vision and natural language processing. As a result, deep learning models have been widely used in the field of intrusion detection. A deep neural network (DNN) with four hidden layers and 100 hidden units with the ReLU function was employed for the intrusion detection model in [19]. Vinayakumar *et al.* used CNNs [20] to build an IDS, and they evaluated effectiveness of various shallow and deep networks in IDSs [41]. Meanwhile, Yin *et al.* used LSTMs [22] as the deep learning approach for intrusion detection. Papamartzivanos *et al.* used a self-taught learning method to deliver a self-adaptive IDS [42]. Furthermore, Yang *et al.* proposed a new intrusion detection model that combined an improved conditional variational AutoEncoder (ICVAE) with a DNN [24] to enhance detection rates. Despite the more accurate predictions, the intrusion detection models are becoming more and more complex. This leads to the predictions from these models become more difficult to understand.

In order to overcome this limitation, some works for interpreting the intrusion detection model have emerged. Amareasinghe and Manic [27] used a method named Layer-wise Relevance Propagation (LRP) [43] to calculate input feature contributions and generated online and offline feedback for users to help them understand what features drove the predictions in IDSs. Marino *et al.* [28] used an adversarial approach to generate explanations for incorrect classifications made by IDSs. Li *et al.* [29] used "Local Explanation Method using Nonlinear Approximation" (LEMNA) [44] to explain the output of an anomaly-based IDS. However, the methods used in the above works lack a solid theoretical foundation. In this paper, SHAP, having a solid theoretical foundation in game theory, is used to improve the interpretability of the IDS. Specifically, a framework proposed in this paper uses SHAP to give local and global interpretability. At the same time, the differences in explanations between the one-vs-all classifier and the multiclass classifier are also discussed.

III. BACKGROUND

Model interpretability can be divided into two categories: global interpretability and local interpretability [45]. Global interpretability means the users can understand the model directly from its overall structure. Local interpretability just exams an input, and it tries to find out why the model makes a certain decision. In this paper, SHAP [30] is used to increase the interpretability of the intrusion detection systems. SHAP is a method that can do local and global interpretability at the same time, and it has a solid theoretical foundation compared to other methods. SHAP connects LIME [32] and Shapley Values [33]. Therefore, LIME and Shapely value are firstly introduced in this section. Then, a brief analysis of SHAP is included.

A. LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS (LIME)

LIME, proposed in [32], is a concrete implementation of local surrogate models. In this paper, the authors trained surrogate models to approximate the predictions of the black box models, which are needed to give explanations. Instead of training a global surrogate model, LIME just focuses on training the local surrogate model to interpret the individual predictions.

The idea behind LIME is quite intuitive. It generates a new dataset consisting of permuted samples, and the corresponding predictions of the black box model are also computed. LIME then trains an interpretable model on this new dataset. The interpretable models can be any understandable models, like linear regression, logistic regression, and decision tree. And the local surrogate model should be a good approximation of the black box model predictions locally. The local surrogate model can be calculated in the following way.

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \left\{ \mathcal{L}\left(f, g, w^x\right) + \Omega(g) \right\}, \qquad (1)$$

where:

- g represents the explanation model for the instance x, (e.g., linear regression).
- G is the family of possible explanations. For example, all possible linear regression models.
- $\mathcal{L}$ is the loss function (e.g., mean squared error), which is used to measure how close the predictions from the explanation model are to the original model.
- f represents the original model.
- w^x defines the weight between the sampled data and the original data. If the sampled data is similar to the original data, the weight is greater, and vice versa.
- $\Omega(g)$ represents the complexity of model g.

According to (1), LIME wants to train a local, interpretable surrogate model g on the new dataset by minimizing $\mathcal{L}(f, g, w^x) + \Omega(g)$, and then explain the prediction of an instance x by interpreting the local model $\xi(x)$.

B. SHAPLEY VALUE

The Shapley value, introduced by Shapley in [33], is a technique used in game theory to determine how much each player in a collaborative game has contributed to the success. This method can be used to interpret the machine learning predictions. The Shapley value is the average contribution of a feature value to the prediction in all possible coalitions:

$$\phi_i(f, x') = \sum_{z' \subseteq \{x'_1, \ldots, x'_n\} \setminus \{x'_i\}} \frac{|z'|!\left(M - |z'| - 1\right)!}{M!} * \left[f\left(z' \cup x'_i\right) - f\left(z'\right)\right], \qquad (2)$$

where:

- z' is a subset of the features used in the model.
- x' is the vector of feature values of the instance to be explained. The x'_i is explained in (2).
- M is the number of features.
- $f\left(z'\right)$ is the prediction for feature values in set z'. When calculating $f(z')$, the ith feature is masked out and then simulated by drawing random instances or the random values of the ith feature from the dataset.

The Shapley value is the only method which satisfies three properties: Symmetry, Dummy, and Additivity. These three properties can be considered for a definition of a fair payout.

Symmetry: The contributions of two feature values i and j should be the same if they contribute equally to all possible coalitions. If

$$f\left(z' \cup x'_i\right) = f\left(z' \cup x'_j\right), \quad \text{for all } z' \subseteq \{x'_1, \ldots, x'_n\} \setminus \{x'_i, x'_j\}, \qquad (3)$$

then $\phi_i(f, x) = \phi_j(f, x)$.

Dummy: If a feature i does not change the predicted value regardless of which coalition of feature values it is added to, this feature should have a Shapley value of 0. If

$$f\left(z' \cup x'_i\right) = f\left(z'\right), \quad \text{for all } z' \subseteq \{x'_1, \ldots, x'_n\} \setminus \{x'_i\}, \qquad (4)$$

then $\phi_i(f, x) = 0$.

Additivity: If $f\left(z' \cup x'_i\right) = f^1\left(z' \cup x'_i\right) + f^2\left(z' \cup x'_i\right)$, then $\phi_i(f, x) = \phi_i(f^1, x) + \phi_i(f^2, x)$

However, the Shapley value requires a lot of computing time. An exact computation of the Shapley value is computationally expensive because there are 2^k possible coalitions of the feature values, and the ''absence'' of a feature has to be simulated by drawing random instances, which increases the variance for the estimation of the Shapley values.

C. SHAP (SHAPLEY ADDITIVE EXPLANATIONS)

SHAP is a unified framework proposed by Lundberg and Lee [30] for interpreting predictions. It explains the prediction of an instance x by computing the contribution of each

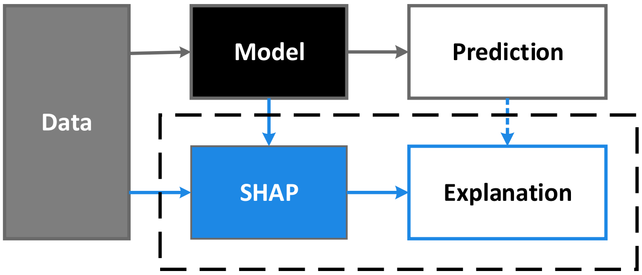


FIGURE 1. Overview of how to use SHAP to interpret the predictions of any model.

feature to the prediction. Fig. 1 presents an overview of how to use SHAP to interpret the predictions of any model.

One innovation that the method SHAP brings to the table is that the Shapley value explanation is represented as a linear model. That view connects the two methods, LIME and Shapley Values. Each SHAP value measures how much each feature in the model contributes, either positively or negatively. SHAP value offers two essential benefits. First, SHAP value can be calculated for any model rather than just simple, linear models. Second, each record has its own set of SHAP value.

SHAP specifies the explanation for an instance x as:

$$g\left(z'\right) = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j, \tag{5}$$

where:

- g is the explanation model.
- z' is the coalition vector (also called simplified features), and $z' \in \{0, 1\}^M$. The 1 in z' means the features in the new data are the same as those of the original data (the instance x), while the 0 means the features in the new data are different from those of the original data (the instance x).
- M is the maximum coalition size.
- $\phi_j \in \mathbb{R}$ is the feature attribution for the feature j for instance x. It is the Shapley value. If ϕ_j is a large positive number, it means feature j has a large positive impact on the prediction made by the model.

The big difference to LIME is the weighting of the instances in the regression model. LIME weighs the instances according to how close they are to the original instance. The more 1's in the coalition vector, the bigger the weight in LIME. However, SHAP weighs the sampled instances according to the weight which the coalition would get in the Shapley value estimation. Small coalitions (few 1's) and large coalitions (i.e., many 1's) get the largest weights. Lundberg *et al.* proposed the SHAP kernel in [30] as:

$$\pi_x\left(z'\right) = \frac{(M-1)}{\binom{M}{|z'|}|z'|\left(M-|z'|\right)}, \tag{6}$$

where:

- M is the maximum coalition size.
- $|z'|$ is the total number of the entry of 1 in instance z'.

Lundberg *et al.* showed that linear regression with this kernel weight yields Shapley values. The details of estimating SHAP values for an instance x consists of 5 steps:

- Sample coalitions $z'_k \in \{0, 1\}^M, k \in \{1, \ldots, K\}$, where an entry of 1 means that the corresponding feature value is ''present'' and 0 means that it is ''absent''.
- Convert z'_k to the original feature space and the get prediction by applying model $f(h_x(z'_k))$, where $f(x)$ is the the original model and $h_x(z') : \{0, 1\}^M \to \mathbb{R}$, maps 1's to the corresponding value from the instance x that we want to explain, and maps 0's to the values of another instance that we sample from the data.
- Compute the weight for each z'_k according to (6).
- Fit weighted linear model by optimizing the following loss function L:

$$L\left(f, g, \pi_x\right) = \sum_{z' \in Z}\left[f\left(h_x\left(z'\right)\right) - g\left(z'\right)\right]^2 \pi_x\left(z'\right), \tag{7}$$

where Z is the training data.

- Get Shapley values ϕ_k, the coefficients from the linear model.

The SHAP values provide three significant advantages compared to other methods. First, SHAP has a solid theoretical foundation in game theory. Shapley values are the only solutions that satisfy three properties: Symmetry, Dummy, and Additivity. SHAP can also satisfy these since it gets Shapley values from linear models. Second, SHAP connects LIME and Shapley values. It helps to unify the field of interpretable machine learning. At last, SHAP has a fast computation for machine learning models compared to calculating Shapley value directly.

IV. PROPOSED METHOD

This section introduces the proposed framework, which is used to improve the interpretability of any IDS. When human operators use IDSs to make decisions, interpretability is almost as important as the accuracy. Therefore, a framework that can improve the transparency of the IDS is necessary. Fig. 2 shows the framework used in this paper. There are two parts in Fig. 2, the left part is the traditional structure of the IDS, and the right part is used to improve the interpretability of the IDS.

For the traditional IDS, the dataset, trained intrusion detection models, and the predictions of the models are included. The trained intrusion detection models contain two different classifiers, the one-vs-all classifier [46] and the multiclass classifier. The one-vs-all classifier consists of multiple binary classifiers. These two kinds of classifiers are used to compare the results of the interpretation. The results can provide cybersecurity experts guidance and reference when they design the structure of the intrusion detection model.

The focus of the presented framework is on improving the interpretability of the IDS. Therefore, in addition to the IDS's predictions, local explanations and global explanations are generated to improve the security experts' trust in the IDS. The SHAP is used to provide an explanation in this paper.

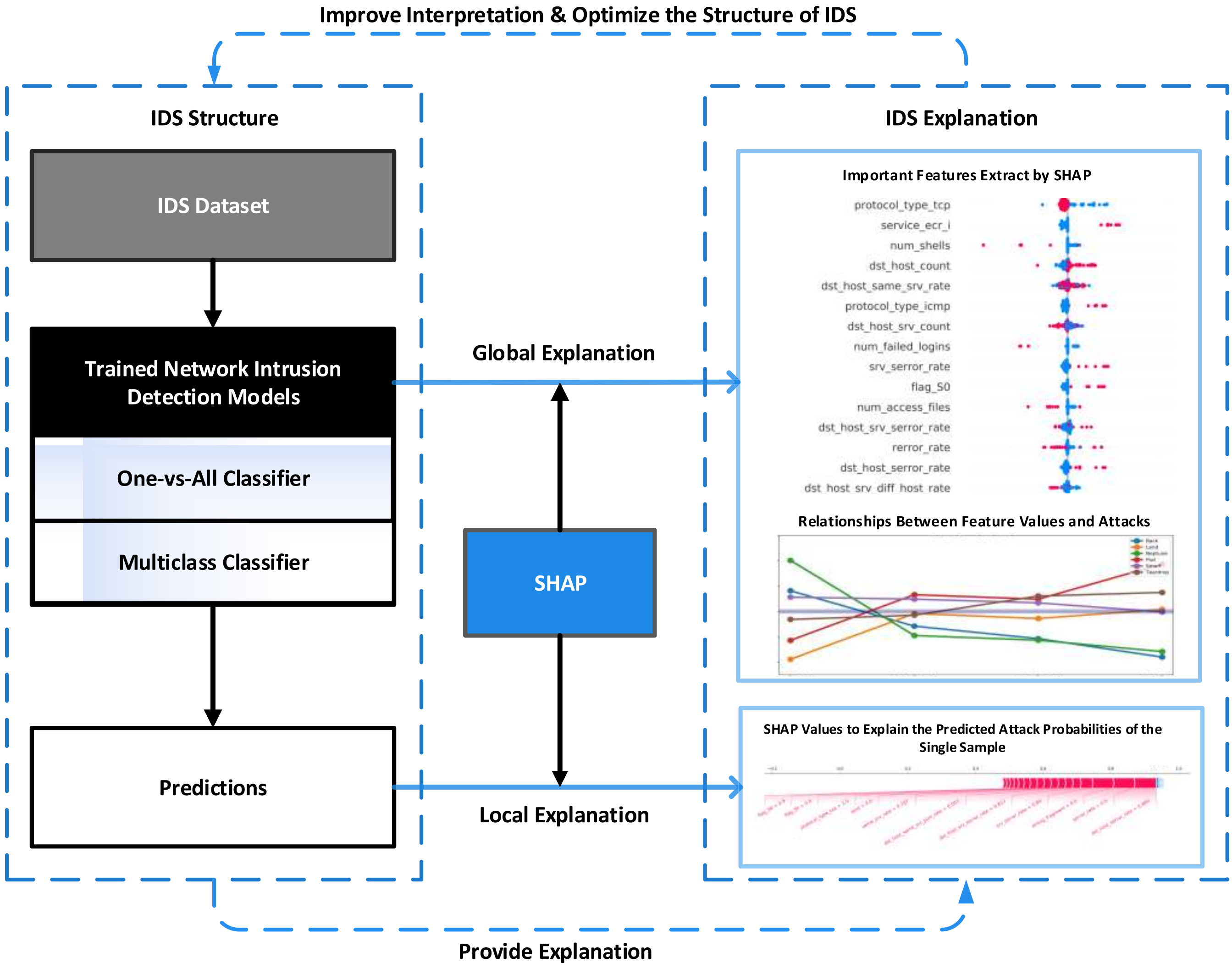


FIGURE 2. Overview of the structure of the proposed framework.

There are two methods of global explanations in this presented framework. The first method can analyze the important features of the IDS (the detailed information and the clear pictures are shown in section V-F1). The second method presents relationships between the value of a feature and the impact on the prediction (the clear pictures are shown in section V-F2). The local explanation explains the output of an IDS. This method also gives the relevance of input features for the IDS prediction (the clear pictures are shown in section V-E).

For any IDS, the presented framework, which is shown in Fig. 2, can be used to improve the transparency of the system. The cybersecurity experts can have the ability to validate the IDSs decisions by using local and global interpretations. By combining the above two methods, it can ultimately help security experts to have a better understanding of IDSs. Furthermore, one-vs-all classifier and multiclass classifier are used in this framework. Thus, by comparing the differences in interpretation between these two classifiers, cybersecurity personnel can adjust the structure of the IDS, then make the predictions from this IDS easier to understand.

A. LOCAL EXPLANATION

SHAP can compute Shapley values and then give the local explanations to the model. The Shapley values show how much each feature contributes to the final prediction. For example, high *dst_host_serror_rate* contributes to the model to judge the data as DoS. At the same time, a new visualization method [47] is used to make the results more intuitive. Through this method, security experts can clearly understand the reasons why the IDS makes the judgments under the specific data. To be specific, the security experts can find whether the features increase or decrease the predicted probabilities made by the model. Detailed experiments and results are shown in the section V-E.

B. GLOBAL EXPLANATION

1) IMPORTANT FEATURES EXTRACTED BY SHAP

Shapley values can be combined to get the global explanations. A matrix of Shapley values is generated by running SHAP for every instance. This matrix has one row per data instance and one column per feature. The entire model can be interpreted by analyzing the Shapley values in this matrix.

The important features can be obtained by using the matrix of Shapley values. First, we average the absolute Shapley values per feature across this matrix:

$$I_j = \sum_{i=1}^{N} \|\phi_j(x_i)\|, \tag{8}$$

where:

- I_j refers to the average Shapley value of the j-th feature.
- $\phi_j(x_i)$ is the Shapley value of the j-th feature in the i-th data.
- N is the total number of samples in the dataset.

Next, the features can be sorted in the decreasing order of their feature importance, and then the top N most important features of the IDS can be obtained.

2) RELATIONSHIPS BETWEEN FEATURE VALUES AND ATTACKS

In most cases, only knowing the important features of the model is not enough, further understanding the relationships between the values of the feature and the predictions of the IDSs can help cybersecurity experts better understand the model. In particular, when security experts understand the relationships between feature values and different types of attacks, they can have more knowledge about the IDS and network attacks.

In this paper, a method based on SHAP to calculate relationships between feature values and the types of attacks is proposed, and a new way of information visualization is also given.

The idea behind this new method is simple: first, the value of the feature is divided into several intervals; then, the Shapley values of this feature are calculated across the data. Finally, the Shapley values in each interval are averaged. The above calculation procedure is shown in (9), where M denotes the number of samples per interval, and x_i denotes the i-th feature in x. The $\overline{\phi_i(x)}$ represents the average Shapley value of the i-th feature when $x_i \in interval_n$. The visualization method is described in detail in section V-F2.

$$\overline{\phi_i(x)} = \frac{1}{M} * \sum_{x_i \in interval_n} \phi_i(x). \tag{9}$$

V. EXPERIMENTS AND RESULTS

In this section, the experimental setup is discussed, including the dataset used in the experiment, the structure of the IDSs, the way for training the intrusion detection models, and the performances of the models. Then, the experimental results are presented. The experiments show the results of the local explanation and two different kinds of global explanations used in our framework. At the same time, the differences in explanations between the one-vs-all classifier and the multi-class classifier are mentioned. The goal of this experiment is to get insight into what the models have learned and provide explanations to IDSs' predictions.

A. NSL-KDD DATASET

In intrusion detection, the KDD'99 dataset [48] has been widely used. However, there are several problems with the KDD'99 dataset, such as an unbalanced distribution of data and a huge number of redundant records. Therefore, NSL-KDD [31] was proposed to solve these issues. The NSL-KDD dataset is well suited for intrusion detection compared to the KDD'99 dataset.

Therefore, the experiments in this paper are carried out by using KDDTrain+.txt as the training data, and KDDTest+.txt as the testing data; both of them come from NSL-KDD. There are five classes in the dataset: Normal, Probe, denial of service (DoS), user to root (U2R), and remote to local (R2L). The training dataset is made up of 21 different attacks, while the test dataset is made up of 37 different attacks [49]. This means that there are 16 novel attacks in the test dataset. Each record in the NSL-KDD dataset has 41 features, and the detailed descriptions of the features are given in [50].

B. DATA PREPROCESSING

1) SYMBOLIC FEATURES

There are three symbolic data types in the NSL-KDD dataset: *protocol_type*, *flag*, and *service*. NSL-KDD dataset has 3 distinct protocols, namely TCP, UDP, and ICMP. There are also 11 *flag* values (e.g., SF, REJ, etc.) and 70 *protocol_type* values (e.g., http, telnet, etc.). A detailed introduction can be found in Table 1. A one-hot encoder is used to convert these symbolic features in this experiment.

2) BINARY FEATURES

There are six binary data types in the NSL-KDD dataset: *land*, *logged_in*, *root_shell*, *su_attempted*, *Is_hot_login*, and *Is_guest_login*. These binary variables remain unchanged in this experiment.

3) CONTINUOUS FEATURES

There are 32 continuous data types in the NSL-KDD dataset: *duration*, *src_bytes*, *dst_bytes*, etc. In this paper, the min-max normalization method is used for these continuous variables. That is,

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{10}$$

After the normalization process, continuous data is in the range [0, 1].

After preprocessing, each record in the NSL-KDD dataset has 122 dimensions.

C. PERFORMANCE EVALUATION

Accuracy, precision, recall, and F1-score are used as evaluation indicators to test the algorithm's performance.

Before introducing the indicator, the ground truth value is discussed. In the experiment, true positive (TP) represents the number of connections correctly classified as attacks. True negative (TN) represents the number of connections correctly

TABLE 1. The detailed information about three symbolic features.

Protocol Type	Service					Flag
No. 1-3	No. 1-14	No. 15-28	No. 29-42	No. 43-56	No. 57-70	No. 1-11
tcp	aol	exec	klogin	pop_2	telnet	OTH
udp	auth	finger	kshell	pop_3	tftp_u	REJ
icmp	bgp	ftp	ldap	printer	tim_i	RSTO
	courier	ftp_data	link	private	time	RSTOS0
	csnet_ns	gopher	login	red_i	whois	RSTR
	ctf	harvest	name	mtp	smtp	S0
	daytime	hostnames	netbios_dgm	remote_job	urh_i	S1
	discard	http	netbios_ns	uucp	urp_i	S2
	domain	http_2784	netbios_ssn	uucp_path	rje	S3
	domain_u	http_443	netstat	sql_net	shell	SF
	echo	http_8001	nnsp	ssh	vmnet	SH
	eco_i	imap4	nntp	sunrpc	X11	
	ecr_i	IRC	ntp_u	supdup	Z39_50	
	efs	iso_tsap	pm_dump	systat	other	

classified as others. False positive (FP) represents the number of attack connections wrongly classified as others. False negative (FN) represents the number of normal connections wrongly classified as attacks.

Based on the above terms, the following are ways to calculate the four statistical measures.

(1) **Accuracy:** This measures the ratio of correctly recognized records to the entire test dataset (Accuracy $\in [0, 1]$).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

(2) **Precision:** The ratio of the number of samples correctly classified as attack to all the samples classified as attack (Precision $\in [0, 1]$).

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

(3) **Recall:** The ratio of the number of samples correctly classified as attack to all the attack records (Recall $\in [0, 1]$).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

(4) **F1-score:** F1-score is used to measure precision and recall at the same time. It uses the harmonic mean in place of the arithmetic mean (F1-score $\in [0, 1]$).

$$F1\text{ -score } = \frac{2 * \text{ Recall } * \text{ Precision}}{\text{Recall } + \text{ Precision}} \tag{14}$$

D. STRUCTURE AND PERFORMANCE OF THE INTRUSION DETECTION MODEL

In order to compare the differences in interpretation between the one-vs-all classifier [46] and the multiclass classifier, these two classifiers are trained separately. In this experiment, the one-vs-all classifier involves five distinct binary classifiers, each designs for recognizing a particular class — Normal, DoS, Probe, U2R, and R2L. The multiclass classifier in the experiment is a single model used to distinguish five classes in the NSL-KDD data set, one normal and four attacks which are mentioned above.

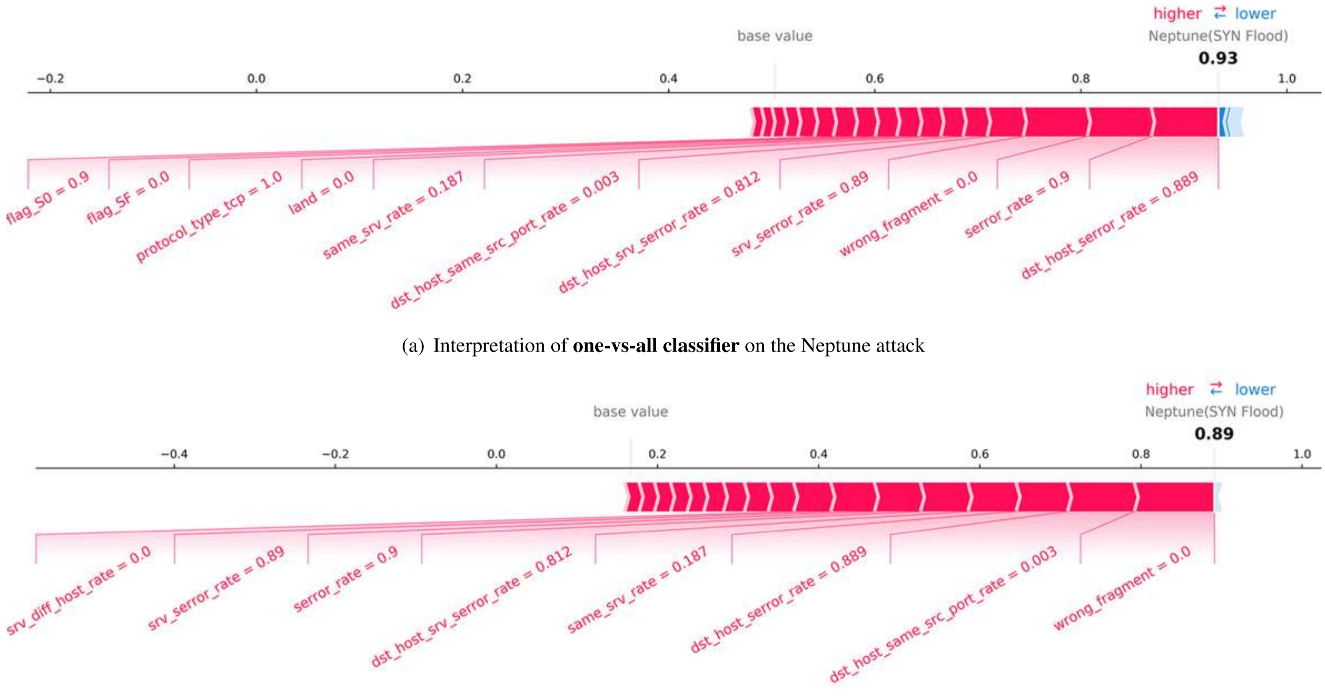
TABLE 2. The structures of one-vs-all classifier and multiclass classifier.

Model Name	Model Structure
The Binary Classifiers in One-vs-All Classifier	[122,100,50,30,2]
The Multiclass Classifier	[122,100,50,30,5]

The fully-connected networks with ReLU activation are used in this experiment. The structures of each binary classifier in the one-vs-all classifier and the structure of the multiclass classifier are shown in Table 2. The input dimension of each classifier is 122. Each classifier has the same number of hidden layers, and the hidden layers contain 100, 50 and 30 neurons respectively. The only difference is that the output dimension of each binary classifier is 2, while the output dimension of multiclass classifier is 5. This is performed mainly to observe the differences between a one-vs-all classifier and a multiclass classifier when they are interpreted.

Pytorch [51] is used to build the intrusion detection model mentioned above. All of the classifiers use the Adam optimizer [52], with an initial learning rate of 0.01, a decay of 0.5 per 100 epochs, and a total of 500 epochs.

Although the two classifiers in this experiment aim to compare the differences in interpretations, we still need to make sure that these two classifiers have good detection rates. The NSL-KDD test dataset KDDTest+ is used to evaluate



(a) Interpretation of **one-vs-all classifier** on the Neptune attack

(b) Interpretation of **multiclass classifier** on the Neptune attack

FIGURE 3. **Interpretation of two classifiers on Neptune attack.**

TABLE 3. **Comparison results on the NSL-KDD (KDDTest+) dataset.**

Model	Performance			
	Accuracy	Precision	Recall	F1-score
KNN	0.731	0.720	0.731	0.684
Random Forest	0.753	0.814	0.753	0.715
SVM-rbf	0.702	0.689	0.702	0.656
ResNet50 [53]	0.791	-	-	-
DNN 5 Layers [54]	0.785	0.810	0.785	0.765
One-vs-All Classifier	**0.806**	**0.828**	**0.806**	**0.807**
Multiclass Classifier	**0.803**	**0.828**	**0.803**	**0.792**

the performance of the model. The overall performance of these two classifiers is shown in Table 3. Generally speaking, the two classifiers used in the experiment have good classification performances.

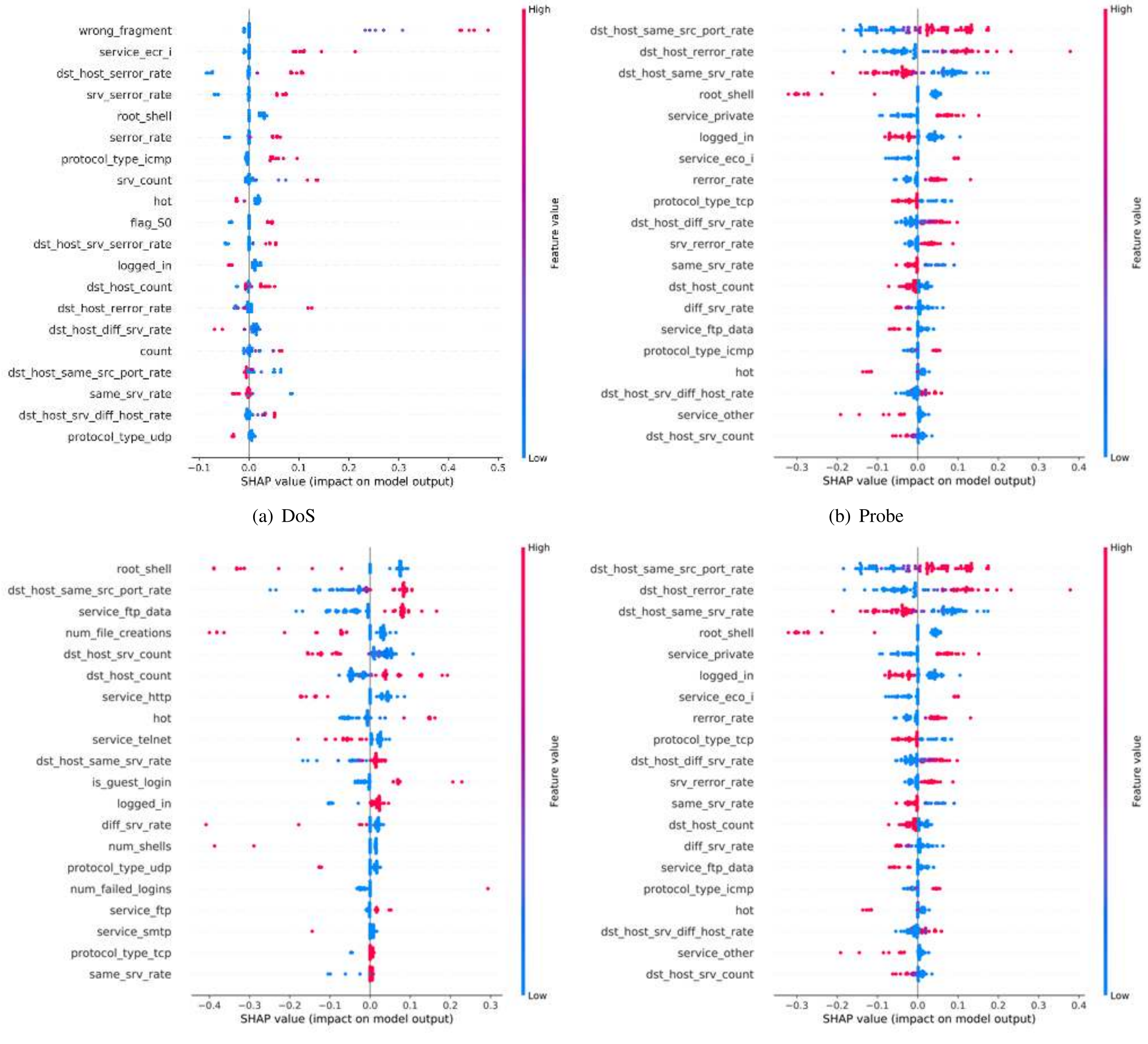
E. RESULTS ABOUT LOCAL EXPLANATION

The results about local explanation of IDS are discussed in this section. At the same time, the differences in local interpretations between the one-vs-all classifier and the multiclass classifier are checked. A specific attack, Neptune, is used as an example to conduct experiments. The visualization method in this section can refer to [47]. In a Neptune attack, the attacker often uses a fake IP address and random ports to send repeated SYN packets to every port on the targeted server. This can cause network saturation [55]. Therefore, Neptune attacks always have high SYN error connections compared with other attacks.

100 Neptune attacks are randomly selected, and then the average of Shapley values of each feature is calculated to avoid randomness. Fig. 3 shows the contributions of each feature value to these two models' judgments. Each feature value is a force that either increases or decreases the prediction. The bold font in Fig. 3(a) shows that the one-vs-all classifier is 93% sure that these attacks are DoS, and the bold font in Fig. 3(b) shows that the multiclass is 89% sure that these are DoS. It can be seen that the prediction probabilities of the two classifiers have a relatively small difference ($\approx 4\%$). However, there are big differences between the results of their explanations.

As shown in Fig. 3(a), the top four features are *dst_host_serror_rate*, *serror_rate*, *wrong_fragment*, and *srv_serror_rate*. And these values of features increase the probability that the one-vs-all model judge the data as a DoS attack. Three of these features are related to SYN connection errors, which are the typical characteristics of Neptune (as shown in Table 4). Therefore, such solid pieces of evidence make cybersecurity experts more convinced with the IDS's judgments.

As shown in Fig. 3(b), multiclass classifier considers that *wrong_fragment*, *dst_host_same_src_port_rate*, *dst_host_serror_rate*, and *same_srv_rate* play important roles in classification. For example, *wrong_fragment* $= 0$



(a) DoS

(b) Probe

(c) R2L

(d) U2R

FIGURE 4. Top 20 important features of four attacks (Extracted from one-vs-all classifier).

and *dst_host_same_src_port_rate* = 0.003 have a strong impact on the judgment that the data is DoS attack (Neptune). However, only *dst_host_serror_rate* is directly related to Neptune. Therefore, this cannot give security personnel valid reasons to trust the results of the models. In conclusion, the features extracted by multiclass classifier are not directly related to the Neptune attack compared with those extracted by one-vs-all classifier, as shown in Fig. 3.

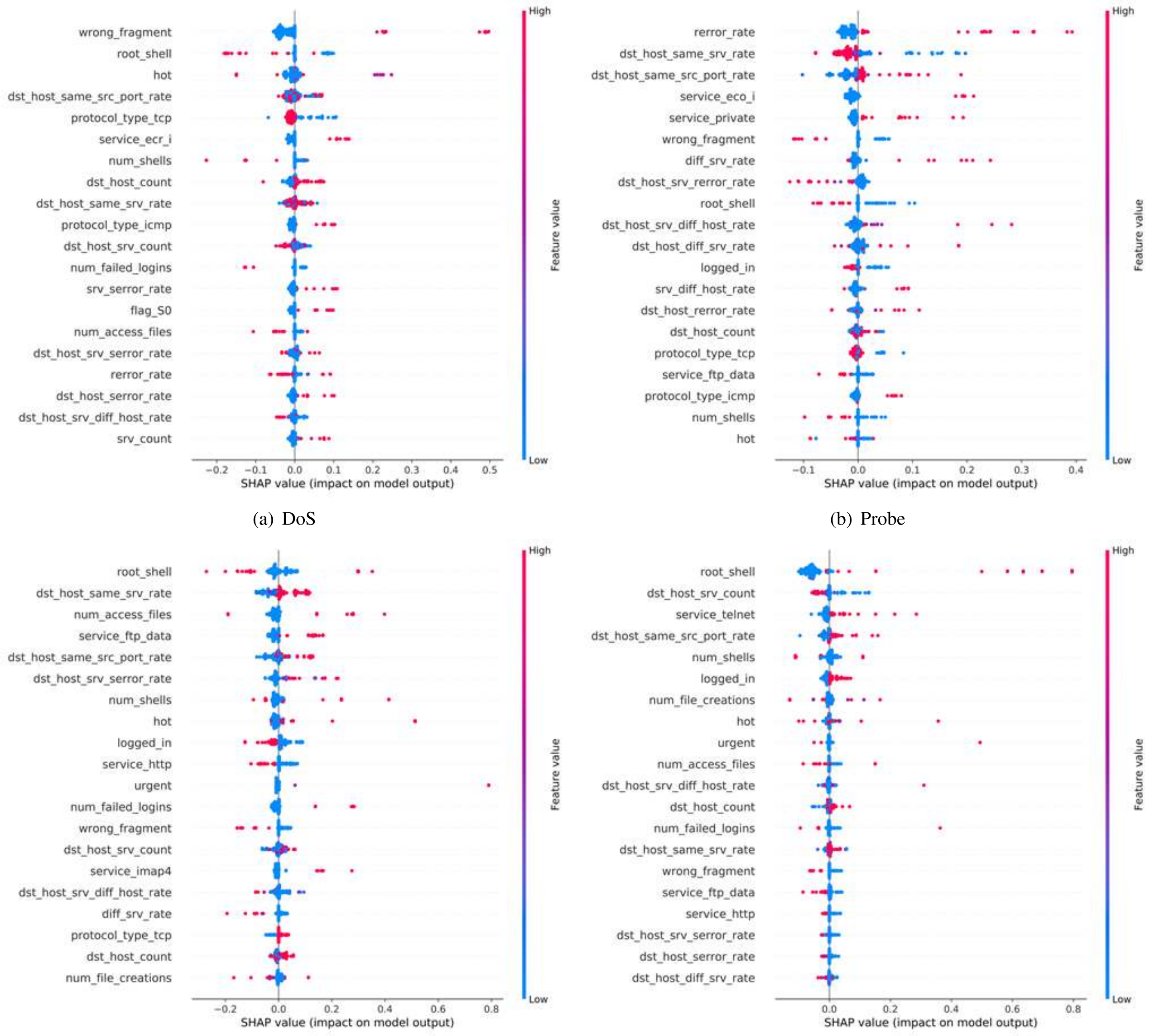
F. RESULTS ABOUT GLOBAL EXPLANATION

1) IMPORTANT FEATURES EXTRACTED BY SHAP

The results of the important features of IDS are discussed in this section. Besides, the differences in important features extracted by SHAP between the one-vs-all classifier and the multiclass classifier are also discussed.

Fig. 4 shows the top 20 features extracted for each type of attack from one-vs-all classifier, and Fig. 5 shows the 20 most important features extracted for each type of attacks from multiclass classifier. Each point in Fig. 4 and Fig. 5 is a Shapley value for a feature and an instance. The position on the y-axis is determined by the feature, and on the x-axis is determined by the Shapley value. The color represents the value of the features from low to high. As the intensity of red color increases, the value of the features increases. Conversely, as the intensity of blue color increases, the value of the features decreases. Overlapping points are jittered in the y-axis direction, and this represents the distribution of the Shapley values per feature. The features are ordered according to their importance.

For instance, Fig. 4(a) shows the top 20 features of DoS attack extracted through the one-vs-all classifier, and Fig. 5(a)



(a) DoS

(b) Probe

(c) R2L

(d) U2R

FIGURE 5. Top 20 important features of four attacks (Extracted from multiclass classifier).

shows the top 20 features of DoS attack extracted through the multiclass classifier. In Fig. 4(a) and Fig. 5(a), it can be seen that the larger the value of *dst_host_serror_rate* (the redder the color), the larger the Shapley value. This means that when the feature value of *dst_host_serror_rate* is larger, both one-vs-all classifier and multiclass classifier are more likely to consider the data as a DoS attack.

A brief summary of differences between the top 20 important features extracted by the one-vs-all classifier and multiclass classifier is discussed here. For DoS, Probe, and R2L attack, 13 of the 20 features are identical, and for the U2R attack, 10 of the 20 features are the same. Then, a detailed analysis of the important features of the DoS attack extracted by these two classifiers is presented.

Table 4 describes every kind of DoS attack that is included in the NSL-KDD dataset. As shown in Table 4, the feature *wrong_fragment* relates to Teardrop and Pod because these attacks send some wrong packets to the target. The feature *service_ecr_i* associates with Smurf because the targets are flooded with ECHO REPLY packets from every host on the broadcast address in this attack. The *srv_serror_rate* relates to Land and Neptune because these attacks send spoofed TCP SYN packets, and *srv_count* relates to Back for all Back attacks use the service of *http*. The features described above — *wrong_fragment*, *service_ecr_i*, *dst_host_serror_rate*, and *srv_serror_rate* — are the top four important features extracted by one-vs-all classifier in Fig. 4(a). The *srv_count* is the eighth most important feature in Fig. 4(a). At the same time, these five features are

TABLE 4. Detailed descriptions of different types of DoS attacks.

Attack Types	Attack Description
Back	This attack causes some versions of the Apache web server to consume excessive CPU time by sending an *http* request with 6000–7000 slashes.
Land	This attack crashes SunOS 4.1 by sending a spoofed TCP SYN packet with the source address equal to the destination address.
Neptune	This attack is also called "SYN flood" or "half open" attack. It causes the web server to exhaust memory and refuse connections until the spoofed connections time out by flooding the target with SYN packets with spoofed source addresses.
Pod	This attack is also known as the "ping of death." It crashes some older operating systems by sending an oversize fragmented IP packet (wrong fragment) that reassembles to more than 65,535 bytes, the maximum allowed by the IP protocol.
Smurf	This attack causes the web server to exhaust memory by sending ICMP ECHO REQUEST packets (ping) to a broadcast address with the spoofed source address of the target. The target is then flooded with ECHO REPLY packets from every host on the broadcast address.
Teardrop	This attack reboots the Linux host by sending a fragmented IP packet that cannot be reassembled because of a gap between the fragments (wrong fragment).

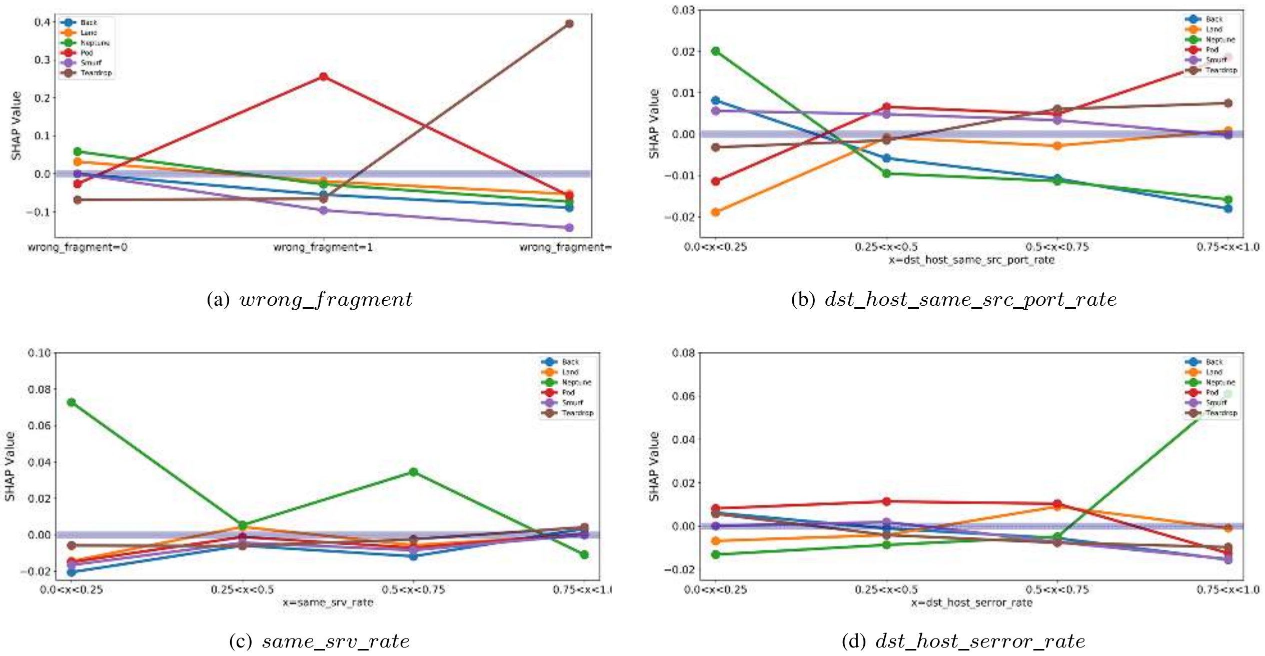
TABLE 5. Important features of four attack types extracted by three methods (feature no.).

Attack Types	Important Features in [56]	Important Features Extracted by One-vs-All Classifier	Important Features Extracted by Multiclass Classifier
DoS	2, 3, 4, 8, 10, 14, 24, 25, 26, 38, 39	4, 8, 10, 12, 14, 25, 26, 35, 36, 38, 39	2, 3, 4, 8, 10, 11, 14, 18, 19, 24, 26
Probe	2, 3, 4, 5, 6, 12, 29, 32, 33, 34, 35, 36, 37, 40	2, 3, 10, 12, 14, 27, 29, 30, 32, 34, 35, 36, 37, 40	2, 3, 8, 10, 12, 14, 18, 27, 30, 31, 32, 34, 35, 36
R2L	1, 2, 3, 5, 6, 10, 12, 14, 16, 24, 32, 33, 35, 36, 37, 38, 39, 41	1, 3, 5, 10, 11, 12, 14, 17, 18, 22, 23, 29, 30, 31, 33, 34, 36, 40	2, 3, 8, 9, 10, 11, 12, 14, 17, 18, 19, 30, 32, 33, 34, 36, 37, 39
U2R	3, 5, 6, 10, 14, 17, 32, 33	3, 10, 14, 17, 33, 34, 35, 36	3, 8, 9, 10, 11, 12, 14, 17

among the top 20 features extracted by multiclass classifier, as shown in Fig. 5(a). However, the ordering of these features in multiclass classifier is not as high as their ordering in one-vs-all classifier.

There are also some other differences in the important features extracted by these two classifiers. Among the top 20 features about DoS attack extracted by multiclass classifier, *num_shells*, *num_failed_logins*, *num_access_file* are included. As a matter of fact, these features are related to the R2L attack. The R2L attack refers to unauthorized access from a remote machine, the attacker intrudes into a remote machine and gains local access of the victim machine. During the process, the attackers might try to log in or modify the files in the victim machine. As shown in Fig. 5(c), when the feature values of *num_failed_logins*, or *num_access_file* are larger, the classifier is more likely to consider the data as an R2L attack. Therefore, when these feature values are small, the multiclass classifier considers the data as a DoS attack. When these features take large values, the multiclass classifier considers the data to be not a DoS attack. In conclusion, the important features extracted by a multiclass classifier may not be directly related to the characteristics of the attacks.

Next, the important features extracted by these two classifiers are compared with the features extracted by Staudemeyer and Omlin [56]. Staudemeyer used a decision tree to extract the features for four common types of attacks. In his paper, 11 features were extracted for the DoS attack, 14 features were extracted for the Probe attack, 18 features were extracted for the R2L attack, and eight features were extracted for the U2R attack. The features extracted by Staudemeyer and the features extracted in this experiment are listed in Table 5, where “Feature No.” corresponds to [50]. For the DoS attack, the eight features extracted by the one-vs-all classifier are the same as those extracted by Staudemeyer. There are also eight features extracted by the multiclass classifier that are the same as those extracted by Staudemeyer, totally accounting for 73%. A more detailed information can be seen in Table 6. In general, the important features extracted by the one-vs-all classifier are more coincident with the important features extracted by the decision tree in [56].



(a) *wrong_fragment*

(b) *dst_host_same_src_port_rate*

(c) *same_srv_rate*

(d) *dst_host_serror_rate*

FIGURE 6. **Relationships between the values of features and specific types of attacks.**

TABLE 6. **The coincidence rate of the features extracted by the two classifiers and the features extracted in [56].**

Attack Types	One-vs-All Classifier	Multiclass Classifier
DoS	73%	73%
Probe	71%	50%
R2L	45%	55%
U2R	63%	50%

At the same time, it should be noted that the important features extracted by using SHAP are better according to the characteristics of attacks. For example, the features *wrong_fragment*, *service_ecr_i*, *dst_host_serror_rate*, and *srv_serror_rate* all have strong relationships with specific attacks, which has been discussed above.

2) RELATIONSHIPS BETWEEN FEATURE VALUES AND ATTACKS

The relationships between feature values and different types of attacks are discussed in this section, and a new way of information visualization about IDS is also described. With this new approach, cybersecurity experts can have a better understanding of the model's judgments. Specifically, they will better know the relationships between the value of a feature and the impact on the prediction. This experiment focuses on the different types of DoS attacks. One-vs-all classifier is used in this experiment.

The results of visualizing the four features — *wrong_fragment*, *dst_host_same_src_port_rate*, *same_srv_rate*, and *dst_host_serror_rate* — are shown in Fig. 6. The broken-line graph in Fig. 6(a) shows the changes in *wrong_fragment*'s Shapley values at different intervals and in different specific attacks. For example, when *wrong_fragment* = 0, the average Shapley values of Pod (red) and Teardrop (brown) are negative numbers. This means that when the value of *wrong_fragment* is 0, it has a negative impact on the judgment that the data is Pod or Teardrop. When *wrong_fragment* = 1, it has a large positive impact on the judgment that the data is Pod. When *wrong_fragment* = 3, it has a large positive impact on the judgment that the data is Teardrop. This matches with the characteristics of the specific attacks shown in Table 4.

Similarly, Fig. 6(b) shows the relationships between the *dst_host_same_src_port_rate*'s values and different specific attacks. Each type of attack can be judged according to the range of *dst_host_same_src_port_rate*. As shown in Fig. 6(b), *dst_host_same_src_port_rate* $<$ 0.25 has a positive impact on the judgment that the data is Neptune, Back, or Smurf, and has a negative impact on the judgment that the data is Pod, Land, or Teardrop.

Furthermore, the *same_srv_rate*'s Shapley values in different intervals and in different specific attacks are shown in Fig. 6(c). The feature *same_srv_rate* represents the percentage of connections that are to the same service to the same destination host as the current connection in the past two seconds. The small value of *same_srv_rate* means that there is a large number of different services connected in the past two seconds. As shown in Fig. 6(c), a small value of *same_srv_rate* has a positive impact on the judgment that the data is Neptune. This is because Neptune has more types of services than other attacks, as shown in Table 7.

TABLE 7. Each attack and its services.

DoS Type	Data	
	Number of Services	Examples of Service
Back	1	http
Land	1	finger, telnet
Neptune	**54**	private, telnet, http, ftp_data, finger, uucp, whois, other 47 services
Pod	2	ecr_i, tim_i
Smurf	1	ecr_i
Teardrop	1	private

In conclusion, this result of visualization fits with the characteristics of the Neptune attack.

The broken-line graph in Fig. 6(d) shows the changes in *dst_host_serror_rate*'s Shapley values at different intervals and in different specific attacks. When *dst_host_serror_rate* $>$ 0.75, it has a large positive impact on the judgment that the data is Neptune. One of the most important characteristics of Neptune is sending a large number of SYN packages, thus the value of *dst_host_serror_rate* is larger. That means this result of visualization matches well with the characteristics of attacks.

Therefore, the relationships between feature values and the attacks match well with the characteristics of attacks. This can help cybersecurity experts to have a good understanding of the predictions made by the IDS, and thereby they can make more accurate judgments. Moreover, this also helps cybersecurity personnel to understand the cyberattacks better.

VI. CONCLUSION AND FUTURE WORK

Nowadays, the intrusion detection models provide no information about the reasons behind their decisions, and most of the works that explain machine learning models focus on other fields like computer vision or natural language processing. Therefore, this study sets out to improve the interpretability of the IDSs.

A new framework, providing local and global explanations, is proposed in this paper. A method called SHAP is used in this framework. This method has a good theoretical foundation and can be applied to any model. The local explanation used in this framework can interpret the predictions made by the IDS. The global explanation contains the important features, the relationships between the values of features and different types of attacks. In addition, the NSL-KDD dataset is used to verify the feasibility of the proposed framework. The experimental results show that the interpretation results, generated by our framework, are consistent with the characteristics of the specific attacks, and the results are very intuitive. Moreover, the differences in the explanations between the one-vs-all classifier and the multiclass classifier are also discussed in this paper. This can help network security staff choose the right structure when they design the intrusion detection system.

The research contributes to the understanding of the reasons behind the judgments made by the IDSs. Through the framework proposed in this paper, the transparency of the IDSs can be improved. Therefore, cybersecurity experts can make better decisions and know the characteristics of different attacks well. They can also optimize the structure of IDSs according to the different explanations between different classifiers, and then design a better intrusion detection system.

The present work has room for improvement. Firstly, more datasets for network intrusion detection systems can be used to demonstrate the feasibility of the framework. Secondly, although SHAP has fast computation for interpreting machine learning models compared with computing Shapley value directly, it is still not possible to work in real-time. Thirdly, the SHAP method can be explored on more sophisticated attacks, like Advanced Persistent Attacks (APTs).

Notwithstanding these limitations, this study offers valuable insight into the interpretability of the IDSs. Further work can focus on experimenting on more datasets, making the framework work in real-time, and explaining more sophisticated attacks.

ACKNOWLEDGMENT

The authors would like to appreciate Joseph Chacko, Bo Yan, Zhe Wang, and Ying Yu for their insightful comments and constructive suggestions. The author would also like to thank his parents, Ling Wang, and Jianfang Jiang, for their endless love, support and encouragement.

REFERENCES

[1] P. K. Senyo, E. Addae, and R. Boateng, "Cloud computing research: A review of research themes, frameworks, methods and future research directions," *Int. J. Inf. Manage.*, vol. 38, no. 1, pp. 128–139, Feb. 2018.

[2] S. Li, L. D. Xu, and S. Zhao, "5G Internet of Things: A survey," *J. Ind. Inf. Integr.*, vol. 10, pp. 1–9, Jun. 2018.

[3] M. Ben-Daya, E. Hassini, and Z. Bahroun, "Internet of Things and supply chain management: A literature review," *Int. J. Prod. Res.*, vol. 57, nos. 15–16, pp. 4719–4742, Aug. 2019.

[4] L. Santos, C. Rabadao, and R. Goncalves, "Intrusion detection systems in Internet of Things: A literature review," in *Proc. 13th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2018, pp. 1–7.

[5] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.

[6] W. Lee, S. J. Stolfo, and K. W. Mok, "A data mining framework for building intrusion detection models," in *Proc. IEEE Symp. Secur. Privacy*, May 1999, pp. 120–132.

[7] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[8] I. B. Arya and R. Mishra, "Internet traffic classification: An enhancement in performance using classifiers combination," *Int. J. Comput. Sci. Inf. Technol.*, vol. 2, no. 2, pp. 663–667, 2011.

[9] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection," *Cluster Comput.*, vol. 21, no. 1, pp. 667–680, Mar. 2018.

[10] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *Proc. 15th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2019, pp. 228–233.

[11] F. E. Heba, A. Darwish, A. E. Hassanien, and A. Abraham, "Principle components analysis and support vector machine based intrusion detection system," in *Proc. 10th Int. Conf. Intell. Syst. Design Appl.*, Nov. 2010, pp. 363–367.

[12] E. Kabir, J. Hu, H. Wang, and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Gener. Comput. Syst.*, vol. 79, pp. 303–318, Feb. 2018.

[13] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," in *Proc. 10th Int. Conf. Mach. Learn. Comput. (ICMLC)*, 2018, pp. 26–30.

[14] G. Serpen and E. Aghaei, "Host-based misuse intrusion detection using PCA feature extraction and kNN classification algorithms," *Intell. Data Anal.*, vol. 22, no. 5, pp. 1101–1114, Sep. 2018.

[15] B. Zhang, Z. Liu, Y. Jia, J. Ren, and X. Zhao, "Network intrusion detection method based on PCA and Bayes algorithm," *Secur. Commun. Netw.*, vol. 2018, pp. 1–11, Nov. 2018.

[16] P. Srinivasu and P. S. Avadhani, "Genetic algorithm based weight extraction algorithm for artificial neural network classifier in intrusion detection," *Procedia Eng.*, vol. 38, pp. 144–153, 2012.

[17] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 29, no. 4, pp. 462–472, Oct. 2017.

[18] I. S. Thaseen, C. A. Kumar, and A. Ahmad, "Integrated intrusion detection model using chi-square feature selection and ensemble of classifiers," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 3357–3368, Apr. 2019.

[19] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 313–316.

[20] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.

[21] M. Sölch, "Detecting anomalies in robot time series data using stochastic recurrent networks," M.S. thesis, Dept. Math., Technische Univ. München, München, Germany, 2015.

[22] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[23] L. Vu, V. L. Cao, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Learning latent distribution for distinguishing network traffic in intrusion detection system," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[24] Y. Yang, K. Zheng, C. Wu, and Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational AutoEncoder and deep neural network," *Sensors*, vol. 19, no. 11, p. 2528, 2019.

[25] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Variational data generative model for intrusion detection," *Knowl. Inf. Syst.*, vol. 60, no. 1, pp. 569–590, Jul. 2019.

[26] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3369–3388, 4th Quart., 2018.

[27] K. Amarasinghe and M. Manic, "Improving user trust on deep neural networks based intrusion detection systems," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2018, pp. 3262–3268.

[28] D. L. Marino, C. S. Wickramasinghe, and M. Manic, "An adversarial approach for explainable AI in intrusion detection systems," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2018, pp. 3237–3243.

[29] H. Li, F. Wei, and H. Hu, "Enabling dynamic network access control with anomaly-based IDS and SDN," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization (SDN-NFVSec)*, 2019, pp. 13–16.

[30] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4765–4774.

[31] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. for Secur. Defense Appl.*, Jul. 2009, pp. 1–6.

[32] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144.

[33] L. S. Shapley, "A value for n-person games," *Contributions to Theory Games*, vol. 2, no. 28, pp. 307–317, 1953.

[34] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013, *arXiv:1312.6034*. [Online]. Available: http://arxiv.org/abs/1312.6034

[35] B. Zhou, Y. Sun, D. Bau, and A. Torralba, "Interpretable basis decomposition for visual explanation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 119–134.

[36] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara, "Transparency and explanation in deep reinforcement learning neural networks," in *Proc. AAAI/ACM Conf. AI, Ethics, Soc.*, Dec. 2018, pp. 144–150.

[37] S. Mishra, B. L. Sturm, and S. Dixon, "Local interpretable model-agnostic explanations for music content analysis," in *Proc. ISMIR*, 2017, pp. 537–543.

[38] S. Anjomshoae, K. Främling, and A. Najjar, "Explanations of black-box model predictions by contextual importance and utility," in *Proc. Int. Workshop Explainable, Transparent Auto. Agents Multi-Agent Syst.* Springer, 2019, pp. 95–109.

[39] T. Wood, C. Kelly, M. Roberts, and B. Walsh, "An interpretable machine learning model of biological age," *FResearch*, vol. 8, p. 17, Jan. 2019.

[40] T. Azevedo, L. Passamonti, P. Lió, and N. Toschi, "A machine learning tool for interpreting differences in cognition using brain features," in *Proc. IFIP Int. Conf. Artif. Intell. Appl. Innov.* Springer, 2019, pp. 475–486.

[41] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1282–1289.

[42] D. Papamartzivanos, F. G. Marmol, and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 7, pp. 13546–13560, 2019.

[43] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, 2015, Art. no. e0130140.

[44] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "LEMNA: Explaining deep learning based security applications," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 364–379.

[45] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," 2018, *arXiv:1808.00033*. [Online]. Available: http://arxiv.org/abs/1808.00033

[46] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *J. Mach. Learn. Res.*, vol. 5, pp. 101–141, Dec. 2004.

[47] S. M. Lundberg, B. Nair, M. S. Vavilala, M. Horibe, M. J. Eisses, T. Adams, D. E. Liston, D. K.-W. Low, S.-F. Newman, J. Kim, and S.-I. Lee, "Explainable machine-learning predictions for the prevention of hypoxaemia during surgery," *Nature Biomed. Eng.*, vol. 2, no. 10, pp. 749–760, Oct. 2018.

[48] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proc. 7th Conf. USENIX Secur. Symp. (SSYM)*, vol. 7. San Antonio, TX, USA: USENIX Association, 1998, p. 6.

[49] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1848–1853, 2013.

[50] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.

[51] P. Adam *et al.*, "Automatic differentiation in pytorch," in *Proc. Neural Inf. Process. Syst.*, 2017.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[53] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, ''Intrusion detection using convolutional neural networks for representation learning,'' in *Proc. Int. Conf. Neural Inf. Process.* Springer, 2017, pp. 858–866.

[54] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, ''Deep learning approach for intelligent intrusion detection system,'' *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[55] M. V. Mahoney and P. K. Chan, ''Learning nonstationary models of normal network traffic for detecting novel attacks,'' in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2002, pp. 376–385.

[56] R. C. Staudemeyer and C. W. Omlin, ''Extracting salient features for network intrusion detection using machine learning methods,'' *South Afr. Comput. J.*, vol. 52, pp. 82–96, Jun. 2014.

MAONAN WANG is currently pursuing the master's degree with the Cybersecurity Institute, Beijing University of Posts and Telecommunications. He has coauthored a conference publication. His current research interests include the internet security, non-invasive and semi-invasive attacks, intersection of machine learning, and security and privacy, especially involving intrusion detection systems.

KANGFENG ZHENG received the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications, in July 2006. He is currently an Associate Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include networking and system security, network information processing, and network coding.

YANQING YANG received the B.S. and M.S. degrees from Xinjiang University, Urumqi, China, in 2006 and 2009, respectively. He is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications. His research interests include network attack detection and anomaly detection.

XIUJUAN WANG received the Ph.D. degree in information and signal processing from the Beijing University of Posts and Telecommunications, in July 2006. She is currently an Instructor Lecturer with the Faculty of Information Technology, Beijing University of Technology. Her research interests include information and signal processing, network security, and network coding.