P-DNN: An Effective Intrusion Detection Method based on Pruning Deep Neural Network

Mingjian Lei, Xiaoyong Li, Binsi Cai, Yunfeng Li, Limengwei Liu, Wenping Kong
Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education
Beijing University of Posts and Telecommunications
Beijing, China
Email: {chnleimingjian, lxyxjtu, cbsbupt, liyfbupt, mwdoublel, kwenping}@163.com

***Abstract*—Today, the scale of global Internet users continues to grow; the Internet has become the main driver of global economic growth; IoT technology is also constantly pushing the process of the Internet of Everything. However, the ever-changing cybersecurity situation is not optimistic and the people's demand for secure network is also increasing. In this paper, for the biggest challenge of building anomaly-based Network Intrusion Detection System: building a high-performance intrusion detection classifier model, we first propose an effective intrusion detection method based on pruning deep neural network: P-DNN. Firstly, we train a deep neural network with complex structure and good intrusion detection performance. Secondly, through the pruning operation, only the connections with more important information in the weight are reserved, reducing the complexity of the model. Finally, retrain the deep neural network to find the best model. We use the KDD Cup 99 dataset to evaluate the effectiveness of the method and achieve exciting results. The model constructed by P-DNN achieves a detection rate of 0.9904 for known attacks and a detection rate of 0.1050 for unknown attacks. By comparing with related work, the model achieves the best intrusion detection performance: COST is reduced to 0.1875 and ACC is increased to 0.9317.**

***Index Terms*—pruning, deep neural network, intrusion detection**

I. Introduction

In 2018, the number of Internet users worldwide had reached 3.8 billion, accounting for 51% of the global total [1]. As of December 2018, 7 of the 10 companies with the highest market capitalization were Internet technology companies, namely Microsoft, Amazon, Apple, Alphabet, Facebook, Alibaba, and Tencent [1]. By 2020, NB-IoT will achieve deep coverage in China for indoor, traffic network and underground pipe network; the base station scale will reach 1.5 million [2]. In 2018, National Internet Emergency Center handled about 106,000 network security incidents and captured more than 100 million computer malware samples [3].

Today, the scale of global Internet users continues to grow; the Internet has become the main driver of global economic growth; IoT technology is also constantly pushing the process of the Internet of Everything. However, the ever-changing cybersecurity situation is not optimistic and the people's demand for secure network is also increasing to protect systems, services and data from unexpected threats. Since it was first proposed by Heberlein in 1991 [4], the Network Intrusion Detection System (NIDS) has received continuous attention from academia and industry. According to the detected technology, NIDS can be divided into signature-based NIDS and anomaly-based NIDS. Signature-based NIDS maintains a signature database, and compares the signature of the traffic with the signature database to determine whether it is normal traffic or attack traffic. Anomaly-based NIDS models normal and abnormal network traffic, and distinguishes normal traffic and abnormal traffic by calculating the similarity between the traffic and the model. Signature-based NIDS (such as the well-known open source NIDS: Snort [5] and Suricata [6]) has high detection rate when detecting known attacks. However, it cannot detect unknown attacks (0-day attacks and variants of attacks) that are not included in the signature database. Anomaly-based NIDS can overcome this weakness. On the basis of achieving a high detection rate for known attacks, anomaly-based NIDS can detect unknown attacks, increasing the probability of detecting high-risk attacks. In summary, anomaly-based NIDS has a greater possibility to build a more secure network environment and meet people's demand for a secure network.

Contributions. The contributions of this paper are summarized as follows:

1) The relationship between the importance of the information owned by the connection and the absolute value of the weight. Through comparative experiments of three pruning methods, we prove that in the deep neural network under the intrusion detection environment, the connections with a larger absolute value of the weight have more important information than the connections with a smaller absolute value of the weight.

2) Application of pruning deep neural network in the field of intrusion detection. For the biggest challenge of building anomaly-based NIDS: building a high-performance intrusion detection classifier model, we first propose an effective intrusion detection method based on pruning deep neural network: P-DNN. We use the KDD Cup 99 dataset to evaluate the effectiveness of the method and achieve exciting results. The model constructed by P-DNN achieves a detection rate of 0.9904 for known attacks and a detection rate of 0.1050 for unknown attacks. By comparing with related work, the model achieves the best intrusion detection performance: COST is reduced to 0.1875 and ACC is increased to 0.9317.

II. Related Work

In 2000, of the 24 entries in the KDD Cup 99 competition, the top three winners used some variants of the decision tree. The winner of the competition made use of an ensemble of 50 $\times$ 10 C5 decision trees, using cost-sensitive bagged boosting [7]. The runner-up first constructed a set of decision trees and then a problem-specific global optimization criterion was used to select an optimal subset of trees to give the final prediction [8]. The third-placed approach used two-layer decision trees. The first layer was trained on the connections which cannot be classified by security experts, and the second layer was built on the connections which cannot be classified by the first layer [9].

In 2001, Ramesh Agarwal et al. proposed a new framework, PNrule. The main idea is learning a rule-based model in two stages: First, find P-rules to predict the presence of a class and then find N-rules to predict the absence of the class. this strategy helps in overcoming the problem of small disjuncts often faced by other sequential covering based algorithms. Another key point in PNrule is the mechanism used for scoring. It allows to selectively tune the effect of each N-rule on a given P-rule. Experimental results showed that the PNrule framework held promise of performing well for real-world multiclassification problems with widely varying class distributions [10].

In 2003, Maheshkumar Sabhnani et al. studied the performance of classical machine learning algorithms on the KDD Cup 99 dataset. They found that some machine learning algorithms performed better for a given attack category. They then built a multi-classifier model in which a specific detection algorithm was associated with an attack category for which it was the most promising. The experimental results showed that significant improvements had been achieved in the detection of Probe, DoS, and U2R [11].

In 2004, Nahla Ben Amor et al. studied the application of Naive Bayes in intrusion detection and proved that even with a simple structure, Naive Bayes could provide very competitive results. In addition, they also compared Naive Bayes with the decision tree. The experimental results showed that the results of using naive Bayes or decision trees were slightly better than those of the KDD Cup 99 winner [12].

In 2005, Chi-Ho Tsang et al. proposed a multi-objective genetic fuzzy system called MOGFIDS for anomaly intrusion detection. The system extracted accurate and interpretable fuzzy rule-based knowledge from network data using an agent-based evolutionary computation framework. The experimental results showed that MOGFIDS achieved robust performance for classifying both intrusion attacks and normal network traffic. In addition, it could search for a reduced feature subset and obtain interpretable fuzzy systems [13].

In 2008, Jiong Zhang et al. used the random forest algorithm to construct the intrusion mode for the limitations of rule-based intrusion detection systems in detecting new attack traffic. By learning the training data, the random forest algorithm could automatically build patterns instead of manual coding rules. The experimental results showed that this method outperformed the winner of the KDD Cup 99 competition [14].

In 2012, Khaled Badran et al. presented a multi-dimensional multi-objective genetic programming feature extraction approach that maps the input feature space into a multi-dimensional decision space to maximize the discrimination between classes. A simple, normal-discriminant-function classifier was used for multi-category classification in the transformed decision space. They applied this approach to the KDD Cup 99 dataset and obtained results that are highly competitive with the KDD Cup 99 winner but with a significantly simpler classification framework [15].

In 2014, Saman Masarat et al. presented a new multi-step framework for intrusion detection systems. In the random feature selection step, features with a higher gain ratio are obtained by using Using Roulette Wheel based on Gain Ratio of features. In classifiers' combination step, adding the fuzzy weighted combiner can tag weights to classifiers related to their cost and performance. Experimental results showed that this approach returned better results than other similar methods [16].

In 2019, R Vinayakumar et al. explored a deep learning model DNN and proposed a highly scalable and hybrid DNNs framework. In this framework, network attacks were detected and classified by learning the abstract and high-dimensional feature representation of the IDS data by passing them into many hidden layers. The experimental results showed that DNN performs well compared to the classic machine learning classifier on the KDD Cup 99 dataset [17].

Since the KDD Cup competition in 1999, researchers have explored many techniques for building intrusion detection classifier model using the KDD Cup 99 dataset. These have promoted more comprehensive development of research on intrusion detection. But there is still a lot of room for research on high-performance classifier models. On the basis of these investigations, we first propose an effective intrusion detection method based on pruning deep neural network: P-DNN, which aims to build a higher performance intrusion detection classifier model and promote the further development of intrusion detection research.

III. Methodology

The subsections below describe the proposed intrusion detection framework and the details of how P-DNN works in the framework.

A. *Intrusion detection framework*

As shown in Figure 1, in the proposed intrusion detection framework, in order to protect demilitarized zone (DMZ) and internal network, the anomaly-based NIDS plays the following role by using port mirroring technology on the key switches: First, detect and alert in real time before the intrusion causes damage. Second, when the intrusion occurs, dynamic defense is performed through linkage with the firewall. Third, after the invasion, the forensic analysis is performed through the logs. Among it, we use P-DNN to complete the biggest

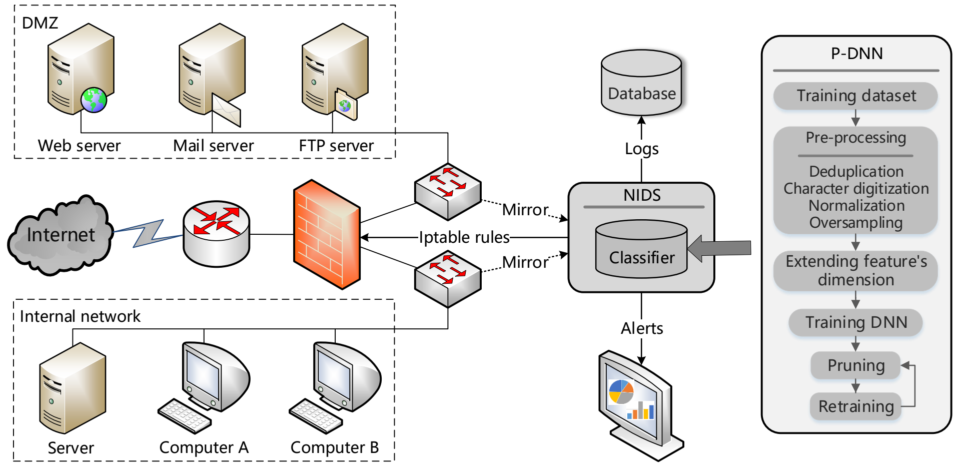

Fig. 1. Intrusion detection framework.

challenge of building an anomaly-based NIDS: building a high-performance intrusion detection classifier model.

B. Why choose the KDD Cup 99 dataset

In 1998, The Defense Advanced Research Projects Agency conducted an intrusion detection assessment project at the MIT Lincoln Laboratory to investigate and evaluate intrusion detection research. They built a network environment that simulates the US Air Force LAN and simulated various user types, various network traffic, and attack methods. Then the DARPA 1998 dataset [18] was constructed with 7 weeks of training data and 2 weeks of testing data. Subsequently, Professor Sal Stolfo and Professor Wenke Lee used data mining technology to perform feature analysis and data preprocessing on this dataset to form a new dataset, namely the well-known KDD Cup 99 dataset [19]. In 2011, Vegard Engen et al. stated that "despite the criticisms, researchers continue to use the data due to a lack of better publicly available alternatives [20]." In 2018, Serhat PEKER et al. investigated the application of neural network in network intrusion detection in the past decade and found that 19 of the 43 articles surveyed used the KDD Cup 99 dataset (44.2%, the largest proportion) [21]. In 2018, Hindy et al. investigated the most cited NIDS researches in the past decade and found that 44 of the 69 articles surveyed used the KDD Cup 99 dataset (63.8%, the largest proportion) [22].

The KDD Cup 99 dataset has been criticized by some network intrusion detection researchers [23] [24] [20], but the authority of the birth and the recognition of many related researchers indicate that it is still the most suitable dataset for evaluating the effectiveness of network intrusion detection methods. This is why we chose the KDD Cup 99 dataset.

C. Selection of data

As shown in Table I, the training dataset includes 494021 instances (kddcup.data_10_percent_corrected [19]), and the testing dataset includes 311029 instances (corrected [19]). The five data types in the table are as follows:

- Normal: normal network connections.

TABLE I
TRAINING DATASET AND TESTING DATASET

	training	%	testing	%
Normal	97278	19.691	60593	19.481
Probe	4107	0.831	4166	1.339
DoS	391458	79.239	229853	73.901
U2R	52	0.011	228	0.073
R2L	1126	0.228	16189	5.205
Total	494021	100	311029	100

- Probe: attackers attempt to collect information about a computer network to circumvent its security controls.
- DoS: attackers prevent certain services from processing legitimate requests because memory resources are exhausted.
- U2R: attackers attempt to exploit certain vulnerabilities to gain root access to the system after getting normal access
- R2L: attackers exploit certain vulnerabilities to get access as local users of the computer through a remote connection.

D. Pre-processing

Data pre-processing includes deduplication, character digitization, normalization and oversampling.

1) Deduplication. As shown in Table II, after the training dataset is deduplicated, the number of instances is reduced from 494021 to 145585. Deduplication preserves non-repeating instances and provides baseline data for subsequent oversampling operation.

TABLE II
DEDUPLICATION OF TRAINING DATASET

	Number of record in training dataset			
	All	%	Distinct	%
Normal	97278	19.691	87832	60.330
Probe	4107	0.831	2130	1.463
DoS	391458	79.239	54572	37.484
U2R	52	0.011	52	0.036
R2L	1126	0.228	999	0.686
Total	494021	100	145585	100

2) Character digitization. The classifier based on deep neural network only uses numerical data for calculation. Therefore, two tasks need to be completed: First, convert non-numeric

features of the dataset instance to numbers. Second, convert the attack type of the dataset instance to the number corresponding to its category. As shown in Table III, there are a total of 3 protocol types, 70 service types, 11 connection states, and 5 attack types converted into corresponding digital identifiers.

TABLE III
CHARACTER DIGITIZATION

	Example	Number
Protocol	tcp, udp, icmp	0,1,2
Service	aol, auth, bgp, courier, csnet_ns, ctf, daytime, discard, domain, domain_u, echo, ...	0-69
Connection	OTH, REJ, RSTO, RSTOS0, RSTR, S0, S1, S2, S3, SF, SH	0-10
Normal	Normal network connections	0
Probe	ipsweep, mscan, nmap, portsweep, saint, satan	1
DoS	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm	2
U2R	buffer_overflow, httptunnel, loadmodule, perl, ps, rootkit, sqlattack, xterm	3
R2L	ftp_write, guess_passwd, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, worm, xlock, xsnoop	4

3) Normalization. Data normalization is the process of scaling the value of each feature to a uniform range, thereby eliminating the bias caused by large numerical features, which is defined as (1).

$$x_i' = \frac{x_i}{\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}} \quad (1)$$

where the vector x(x_1, x_2, ... , x_n) represents the original value of the instance feature, and the vector x$'$(x_1', x_2', ..., x_n') represents the value of the instance feature after the normalization operation.

4) Oversampling. It is easy to see from Table II that the training dataset is highly unbalanced, which is one of the data distribution characteristics of the intrusion detection environment. According to analysis [24], the main reason for the poor performance of low-frequency data by the classifier trained by the KDD Cup 99 dataset is the imbalance of the dataset. In view of this situation, in order to improve the classification effect of low frequency data and the overall performance of the classifier, this paper integrates the oversampling technique [25] into the pre-processing operation. As shown in Figure 2, Probe, U2R, and R2L are oversampled with the number of instances of DOS as the baseline to construct a new relatively balanced training dataset. In new dataset, the number of instances of Probe, U2R, and R2L has been expanded by 25 (54572/2130), 1049 (54572/52), and 54 (54572/999) times respectively.

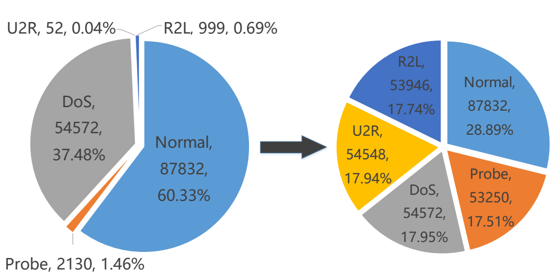

Fig. 2. Oversampling

E. Extending feature's dimension

A lot of research has shown an important conclusion: By training a larger and more complex neural network model, and then gradually pruning to get a smaller and simpler model, the results are better than those obtained by directly training such a small and simple model [26]. Our idea is that by expanding the original feature's dimension of the dataset, the dimension of the input layer of the neural network becomes larger and the complexity of the model becomes higher. This operation prepares for the next pruning operation, which aims to reduce the complexity of the model. As shown in Figure 3, the feature's dimension of the input data is expanded by repeating the original features in order.

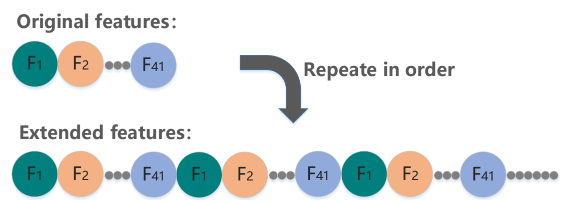

Fig. 3. Extending feature's dimension.

F. Deep neural network

The details of the proposed deep neural network (DNN) architecture are shown in Table IV. The key points are as follows:

TABLE IV
DNN ARCHITECTURE

Layers	Type	units	Activation function	param
0	Input layer	820	/	/
0-1	Full connected	512	ReLU	420352
1-2	Dropout=0.5	/	/	0
2-3	Full connected	256	ReLU	131328
3-4	Dropout=0.5	/	/	0
4-5	Full connected	128	ReLU	32896
5-6	Dropout=0.5	/	/	0
6-7	Full connected	64	ReLU	8256
7-8	Dropout=0.5	/	/	0
8-9	Full connected	32	ReLU	2080
9-10	Dropout=0.5	/	/	0
10-11	Full connected	5	Softmax	165

⋄ Structure: The input layer includes 820 neurons. The five hidden layers include 512, 256, 128, 64, and 32 neurons respectively. The output layer includes 5 neurons.

⋄ Connection mode: Full connection. Each neuron in the current layer is connected to all neurons in the next layer.

⋄ Hidden layer activation function: ReLU. ReLU is a nonlinear activation function. Compared with the linear activation function, it can better express complex classification boundaries and more closely related to the signal excitation principle of neurons, which can improve the performance of the model [27]. In addition, ReLU can helps to reduce the state of vanishing and error gradient issue [28].

⋄ Output layer activation function: Softmax. In the five-category environment of this experiment, Softmax that solves the multi-classification problem is more suitable than Logistic that solves the two-category problem. In addition, we can get the probability distribution that the prediction result belongs to a certain class through Softmax.

◇ Loss function: Categorical cross-entropy. Combining the output layer with Softmax as the activation function, we choose categorical cross-entropy as the loss function, which is defined as (2).

$$loss(pd, ed) = -\sum_x pd(x) \log(ed(x)) \tag{2}$$

where ed is true probability distribution, pd is predicted probability distribution.

◇ Optimizer: Adam. It is designed to combine the advantages of two recently popular methods: AdaGrad [29], which works well with sparse gradients, and RMSProp [30], which works well in on-line and non-stationary settings. Experiments have shown that Adam performs better than other stochastic optimization methods [31].

◇ Training algorithm: Back Propagation algorithm. The algorithm compares the error generated by the theoretical output with the actual output, and reversely adjusts the weight and bias of each layer connection to optimize the parameters of the whole network [32]. From a mathematical point of view, the training of neural network is the iterative process of each layer input under the nonlinear activation function. The process of this iteration is similar to the process of biological growth and evolution. This further explains why neural network can simulate part of human brain function and succeed in many fields.

◇ Dropout: This is a powerful technique to reduce overfitting and improve the generalization of neural network [33]. The key idea is that when the forward propagation, the activation value of the neuron stops working with a fixed probability, so that the neuron does not rely too much on some local features, and the generalization ability of the model is stronger.

G. Pruning

Pruning neural network has excellent performance in the field of deep learning model compression [34]–[36]. The field is dedicated to reducing the storage requirements of the model while maintaining the accuracy of classical neural network models (such as LeNet-5 [37], AlexNet [38], VGG-16 [39]), and promoting the algorithm to be efficiently applied to resource-constrained hardware platforms. Different from the goal of deep learning model compression field, This paper is dedicated to the application of pruning technology in the field of intrusion detection to build a higher performance intrusion detection classifier model, and promote the construction of a more secure network environment. By predicting the parameters of the neural network, Misha Denil et al. pointed out that there are significant redundancy in the parameters used in the neural network [40]. Brandon Reagen et al. also acknowledge the fact that most neural network contain far more information than is needed for precise reasoning [41]. Min Lin et al. also showed that in the neural network, the fully connected layer is easy to cause over-fitting, and the simplification of the fully-connected layer contributes to the improvement of precision [42]. As shown in Figure 4, the idea of pruning in this paper is that by pruning the DNN, only the connections with more important information in the weights are reserved, and the complexity of the model is reduced, thereby improving the intrusion detection performance of the model. Assuming that there are 3 input neurons in a layer and 4 output neurons in the next layer, and the weight matrix is 3×4. The details of the pruning are as follows:

▷ Step 1: Sort. The weights between each neuron in the DNN and all neurons in the next layer are sorted by absolute value.

▷ Step 2: Prune. According to the pruning rate P (representing the proportion of all pruned connections in all connections of the DNN), the connections with smaller absolute value of the weight are pruned (the weights are assigned to 0). The position information of the pruned connections in the original neural network is recorded during the pruning process.

▷ Step 3: Retrain. Retrain DNN. Using the position information of the pruning, the weights of the pruned connections are assigned 0 after each round of retraining.

▷ Step 4: Complete. Get the best model or go to Step 3.

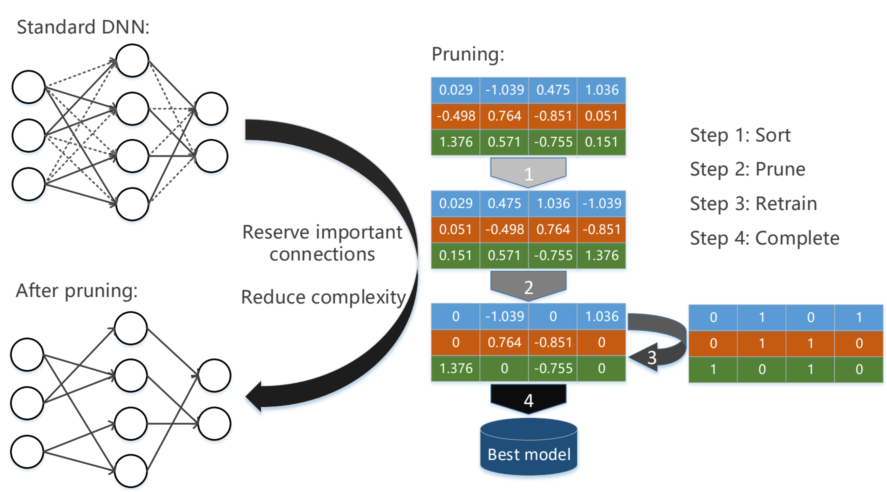

Fig. 4. Pruning.

IV. EXPERIMENTAL RESULTS

This section will first introduce the performance indicators used to evaluate the effectiveness of intrusion detection method. Then, the experimental results of this paper are presented and analyzed. Finally, compare our method with the excellent related work.

A. Performance indicators

In order to evaluate the effectiveness of our proposed method, we propose to use three evaluation indicators: COST, accuracy (ACC), detection rate (DR). Among them, COST is the most important, ACC is the second, and DR is the reference. As shown in Table V, we will explain the calculation details of the three evaluation indicators through an example of confusion matrix.

TABLE V
AN EXAMPLE OF CONFUSION MATRIX

		Predicted				
		Normal	Probe	DoS	U2R	R2L
Actual	Normal	x_{00}	x_{01}	x_{02}	x_{03}	x_{04}
	Probe	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
	DoS	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}
	U2R	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}
	R2L	x_{40}	x_{41}	x_{42}	x_{43}	x_{44}

1) COST. As shown in (3), the smaller the COST value, the better the model. Where N represents the total number of instances tested. CM represents the confusion matrix, and CM(i, j) represents the number of instances originally belonging to class i that are classified as class j. C represents the cost matrix, and C(i, j) represents the cost that the instances originally belonging to class i are classified as class j.

$$COST = \frac{1}{N}\sum_{i=0}^{4}\sum_{j=0}^{4} CM(i,\ j) * C(i,\ j) \tag{3}$$

TABLE VI
KDD CUP 99 COST MATRIX

		Predicted				
		Normal	Probe	DoS	U2R	R2L
Actual	Normal	0	1	2	2	2
	Probe	1	0	2	2	2
	DoS	2	1	0	2	2
	U2R	3	2	2	0	2
	R2L	4	2	2	2	0

As shown in Table VI, the official provided a cost matrix for evaluating entries in the KDD Cup 99 competition [19]. This is also the cost matrix used by COST. In the cost matrix, the magnitude of these cost values are proportional to the impact of the attack on the computing platform. Therefore, we think that COST is the most important evaluation indicator for analyzing the performance of a model from the perspective of detecting intrusion.

2) ACC. As shown in (4), the larger the ACC value, the better the model. Among them, x_{00}, x_{11}, x_{22}, x_{33} and x_{44} represent the number of instances belonging to Normal, Probe, DoS, U2R and R2L that are finally classified correctly.

$$\text{ACC} = \frac{x_{00} + x_{11} + x_{22} + x_{33} + x_{44}}{\sum(X)} \tag{4}$$

Although ACC is a widely used classifier performance evaluation indicator. But when the data is unbalanced (as shown in Table I, the testing dataset is extremely unbalanced), the ACC will be misleading to the researcher. Assuming that the testing dataset with 90 class A instances and 10 class B instances are classified. If the model predicts all instances as class A, the ACC is as high as 0.9. However, the model does not predict any class B instances, which is undoubtedly an unsuccessful classifier. Therefore, we think that ACC is the second most important indicator for analyzing model performance from the perspective of data classification.

3) DR. As shown in (5)-(9), the larger the DR value, the better the model. Among them, the DR is represented by the proportion of the number of correctly classified instances to the total number of instances of this type.

$$DR_{Normal} = \frac{x_{00}}{\sum x_{0i}} \tag{5}$$

$$DR_{Probe} = \frac{x_{11}}{\sum x_{1i}} \tag{6}$$

$$DR_{DoS} = \frac{x_{22}}{\sum x_{2i}} \tag{7}$$

$$DR_{U2R} = \frac{x_{33}}{\sum x_{3i}} \tag{8}$$

$$DR_{R2L} = \frac{x_{44}}{\sum x_{4i}} \tag{9}$$

DR shows the model's ability to detect a certain type of data in detail. Therefore, we think that DR can be used as a reference indicator to analyze the performance of the model.

B. Details of experimental results

1) The relationship between the importance of the information owned by the connection and the absolute value of the weight. After sorting the weights between each neuron in the DNN and all the neurons in the next layer in absolute order, in the process of reducing the same complexity of the model, we set up three pruning methods to compare, where the pruning rate P represents the proportion of pruned connections in all connections of the DNN. Method 1: According to the value P, prune the connections with a larger absolute value of the weight. Method 2: In the order of absolute values from small to large, every other connection, prune x connections (x=1, P=0.5; x=2, P=0.667; ...), reserving some connections with a smaller absolute value of the weight and some connections with a larger absolute value of the weight. Method 3: According to the value P, prune the connections with a smaller absolute value of the weight.

As shown in Figure 5, in contrast, the performance of the model is best under the effect of the pruning method 3 represented by the red line. The model achieves a lower COST, which is more stable and reduces the possibility of losing important information of the model caused by pruning and can

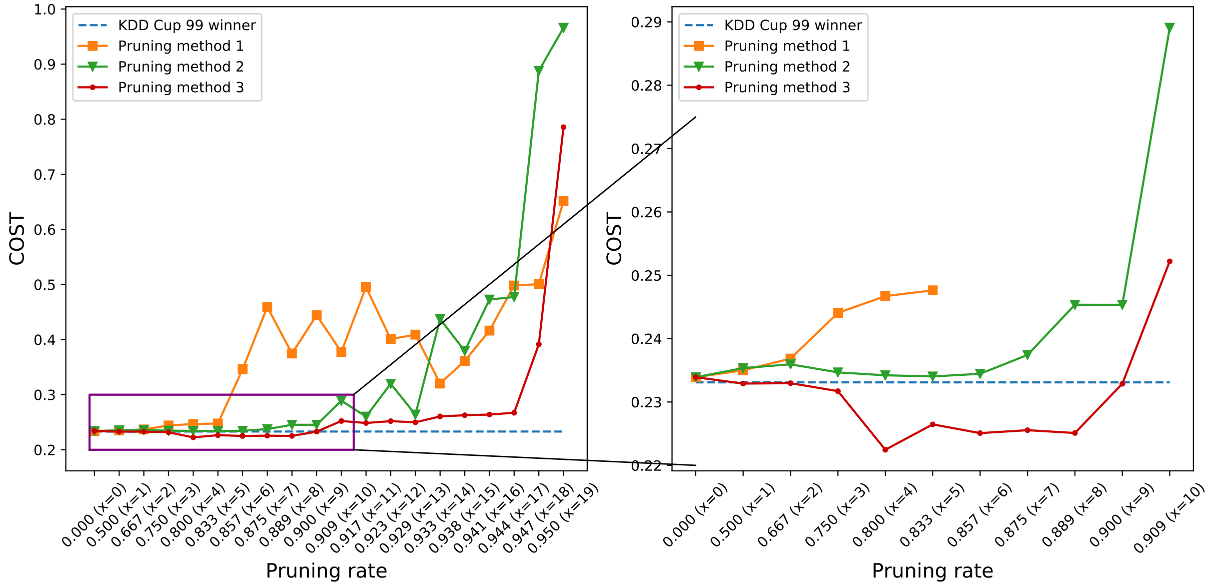

Fig. 5. Comparison of three pruning methods

not be retrained to recover. The second is pruning method 2, and the last is pruning method 3. This proves that in the DNN under the intrusion detection environment, the connection with a larger absolute weight value has more important information than the connection with a smaller absolute weight value. By adopting the P-DNN combined with the pruning method 3 which only the connections with more important information in the weight were reserved and reduce the complexity of the model, we found the best intrusion detection classifier model.

2) Best model. As shown in Figure 6, during the change of the pruning rate P in steps of 0.001, we found the best model when P equals 0.885. The confusion matrix and performance evaluation indicators of the model are shown in Table VII. From the perspective of detecting intrusion, the COST of the model is 0.1875. From the perspective of data classification, ACC of the model is 0.9317. From the perspective of evaluation reference indicator DR, DR_{normal}, DR_{probe}, DR_{DoS}, DR_{U2R} and DR_{R2L} are 0.964, 0.886, 0.968, 0.272 and 0.313 respectively.

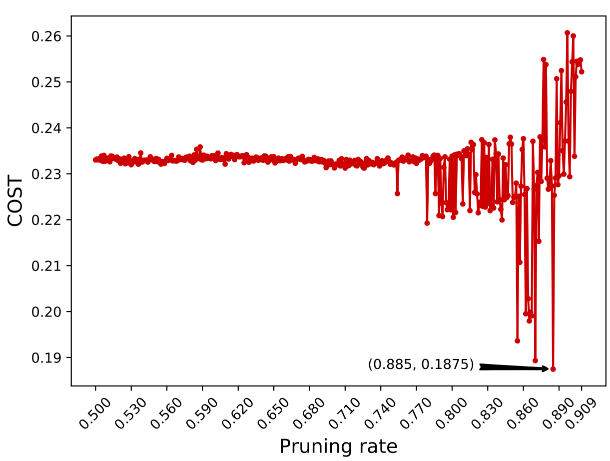

Fig. 6. Find the best model

TABLE VII
BEST MODEL

		Predicted					
		Normal	Probe	DoS	U2R	R2L	DR
Actual	Normal	58385	992	96	409	711	0.964
	Probe	93	3689	366	3	15	0.886
	DoS	5162	403	222595	1064	629	0.968
	U2R	21	44	91	62	10	0.272
	R2L	8650	127	7	2344	5061	0.313
ACC		0.9317					
COST		0.1875					

As shown in Table IX, the model performs well on known attacks (the attack instances exist in the training dataset and testing dataset) with a detection rate as high as 0.9904. The model performed poorly on unknown attacks (the attack instances only exist in the testing dataset) with a detection rate of only 0.1050. But the model exhibits high value for detecting the 0-day attack and variants of attacks. We think that the main reasons for poor performance include the following two points: First, it is difficult to detect application layer attacks, such as mailbomb, through the connection features of the network layer. Second, the features of the dataset are not sufficient to distinguish between the normal connections and the connections of certain attack; as shown in Table VIII, in the testing dataset, there is case where the snmpgetattack instance and the normal instance have exactly the same features.

TABLE VIII
DIFFERENT CONNECTION INSTANCES WITH THE SAME FEATURES

0, udp, private, SF, 105, 146, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 255, 254, 1.00, 0.01, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, normal.
0, udp, private, SF, 105, 146, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0.00, 0.00, 0.00, 0.00,1.00, 0.00, 0.00, 255, 254, 1.00, 0.01, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, snmpgetattack.

TABLE IX
PERFORMANCE ON KNOWN ATTACKS AND UNKNOWN ATTACKS

	Known attacks						Unknown attacks					
	Attack	Detected	Total	DR			Attack	Detected	Total	DR		
Probe	ipsweep	303	306	0.9902	0.9945	**0.9904**	mscan	591	1053	0.5613	0.7406	**0.1050**
	nmap	84	84	1			saint	734	736	0.9973		
	portsweep	344	354	0.9718								
	satan	1633	1633	1								
DoS	back	0	1098	0	0.9942		apache2	169	794	0.2129	0.0911	
	land	3	9	0.3333			mailbomb	0	5000	0		
	neptune	57943	58001	0.9990			processtable	427	759	0.5626		
	pod	44	87	0.5058			udpstorm	1	2	0.5000		
	smurf	163996	164091	0.9994								
	teardrop	12	12	1								
U2R	buffer_overflow	22	22	1	0.8205		httptunnel	5	158	0.0317	0.1587	
	loadmodule	2	2	1			ps	14	16	0.8750		
	perl	2	2	1			sqlattack	2	2	1		
	rootkit	6	13	0.4615			xterm	9	13	0.6923		
R2L	ftp_write	1	3	0.3333	0.848		named	7	17	0.4118	0.0015	
	guess_passwd	3726	4367	0.8532			sendmail	6	17	0.3529		
	imap	0	1	0			snmpgetattack	0	7741	0		
	multihop	4	18	0.2222			snmpguess	0	2406	0		
	phf	0	2	0			worm	0	2	0		
	warezmaster	1315	1602	0.8209			xlock	2	9	0.2222		
							xsnoop	0	4	0		

C. Comparison

In order to locate our research, we have made a comparison as shown in table X.The comparison included recent research using the KDD Cup 99 dataset and achieving satisfactory results. Evaluation indicators include COST, ACC, and DR. Regrettably, due to the different indicators used by different researchers, only incomplete evaluation data can be collected in some research. However, we still include these research in the scope of comparison because of their excellent research results.

As can be seen from Table X, the model constructed by P-DNN is superior in performance to all other related work in the table. From the perspective of detecting intrusion, COST is greatly reduced to 0.1875, and from the perspective of data classification, ACC is slightly increased to 0.9317. These reflect the excellent intrusion detection performance of the model. Maheshkumar Sabhnani et al. stated that all the classic machine learning algorithms tested on the KDD Cup 99 dataset offered an acceptable level of detection performance only for DoS and PROBE attacks and demonstrated poor performance on the U2R and R2L [43]. However, from the perspective of the reference indicator detection rate, compared with other research works, the method we proposed can not only get satisfactory results on DR_{Normal}, DR_{Probe}, and DR_{DoS} but also make DR_{U2R} and DR_{R2L} achieve some improvement.

V. CONCLUSIONS

We first propose an effective intrusion detection method based on pruning deep neural network: P-DNN. Firstly we train a deep neural network with complex structure and good detection performance through extending feature's dimension. Then, through comparative experiments of three pruning methods, it is proved that in the deep neural network under the intrusion detection environment, the connections with a larger absolute value of the weight have more important information than the connections with a smaller absolute value of the weight. On this basis, through the pruning operation, the weights of the deep neural network with a smaller absolute value are assigned to 0, which only reserve the connections with more important information in the weight, reducing the complexity of the model. Finally, retrain the remaining connections with a larger absolute value of the weight to find the best model. We use the KDD Cup 99 dataset to evaluate the effectiveness of the method and achieve exciting results. The model constructed by P-DNN achieves a detection rate of 0.9904 for known attacks and a detection rate of 0.1050 for unknown attacks. At the same time, we explain two main reasons why the model performs poorly on unknown attacks: First, it is difficult to detect application layer attacks through the connection features of the network layer. Second, the features of the dataset are not sufficient to distinguish between

TABLE X
COMPARISON WITH RELATED WORK

	DR_{Normal}	DR_{Probe}	DR_{DoS}	DR_{U2R}	DR_{R2L}	ACC	COST
KDD Cup 99 winner [7]	**0.995**	0.833	0.971	0.132	0.084	0.9272	0.2331
KDD Cup 99 runner-up [8]	0.994	0.845	**0.975**	0.118	0.073	0.9292	0.2356
PNrule [10]	0.995	0.732	0.969	0.066	0.107	0.9259	0.2381
Multi-Classifier [11]	/	**0.887**	0.973	**0.298**	0.096	/	0.2285
Decision Trees [12]	0.994	0.779	0.966	0.136	0.005	0.9280	0.2371
Naive Bayes [12]	0.977	0.883	0.967	0.110	0.087	0.9210	0.2441
MOGFIDS [13]	0.984	0.886	0.972	0.158	0.111	0.9277	0.2317
Random Forests [14]	/	/	/	/	/	0.9293	0.2280
Multi-Objective Genetic Programming [15]	0.995	0.780	0.970	0.114	0.056	0.9240	0.2431
Tree Classifier + Fuzzy Ensemble [16]	/	/	/	/	/	0.9300	0.2179
DNN [17]	0.995	0.764	0.942	0.089	0.243	0.9129	/
P-DNN	0.964	0.886	0.968	0.272	**0.313**	**0.9317**	**0.1875**

the normal connections and the connections of certain attack. By comparing with related work, the model built by P-DNN achieves the best intrusion detection performance: From the perspective of detecting intrusion, COST is greatly reduced to 0.1875; from the perspective of data classification, ACC is slightly increased to 0.9317; from the perspective of the reference indicator detection rate, in addition to the satisfactory results in DR_{Normal}, DR_{Probe}, and DR_{DoS}, it also achieves some improvement in DR_{U2R} and DR_{R2L}.

ACKNOWLEDGMENT

This work was supported by NSFC-General Technology Fundamental Research Joint Fund (No. U1836215), and the National Key Research and Development Program of China (No. 2016QY03D0605).

AVAILABILITY

Codes are available at: https://github.com/BydRay/P-DNN

REFERENCES

[1] "Internet Trends 2019," https://www.bondcap.com/.
[2] "China Internet of Things Application Research Report 2018," http://www.clic.org.cn/xdwlgyl/296885.jhtml.
[3] "A Summary of China's Internet Network Security Situation in 2018," http://www.cac.gov.cn/2019-04/17/c_1124379080.htm.
[4] L. T. Heberlein, B. Mukherjee, K. Levitt, G. Dias, and D. Mansur, "Towards detecting intrusions in a networked environment," Ph.D. dissertation, U. of Calif., Davis, 1991.
[5] "Snort," https://www.snort.org/.
[6] "Suricata," https://suricata-ids.org/.
[7] B. Pfahringer, "Winning the kdd99 classification cup: bagged boosting," *SIGKDD explorations*, vol. 1, no. 2, pp. 65–66, 2000.
[8] I. Levin, "Kdd-99 classifier learning contest: Llsoft's results overview," *SIGKDD explorations*, vol. 1, no. 2, pp. 67–75, 2000.
[9] M. Vladimir, V. Alexei, and S. Ivan, "The mp13 approach to the kdd'99 classifier learning contest," *SIGKDD Explorations. ACM SIGKDD*, vol. 1, no. 2, pp. 76–77, 2000.
[10] R. Agarwal and M. V. Joshi, "Pnrule: a new framework for learning classifier models in data mining (a case-study in network intrusion detection)," in *Proceedings of the 2001 SIAM International Conference on Data Mining*. SIAM, 2001, pp. 1–17.
[11] M. Sabhnani and G. Serpen, "Application of machine learning algorithms to kdd intrusion detection dataset within misuse detection context." in *MLMTA*, 2003, pp. 209–215.
[12] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 420–424.
[13] C.-H. Tsang, S. Kwong, and H. Wang, "Anomaly intrusion detection using multi-objective genetic fuzzy system and agent-based evolutionary computation framework," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005, pp. 4–pp.
[14] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.
[15] K. Badran and P. Rockett, "Multi-class pattern classification using single, multi-dimensional feature-space feature extraction evolved by multi-objective genetic programming and its application to network intrusion detection," *Genetic Programming and Evolvable Machines*, vol. 13, no. 1, pp. 33–63, 2012.
[16] S. Masarat, H. Taheri, and S. Sharifian, "A novel framework, based on fuzzy ensemble of classifiers for intrusion detection systems," in *2014 4th international conference on computer and knowledge engineering (ICCKE)*. IEEE, 2014, pp. 165–170.
[17] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.
[18] "DARPA 1998 dataset," https://www.ll.mit.edu/r-d/datasets.
[19] "Kdd Cup 99 dataset," http://kdd.ics.uci.edu/databases/kddcup99/.
[20] V. Engen, J. Vincent, and K. Phalp, "Exploring discrepancies in findings obtained with the kdd cup'99 data set," *Intelligent Data Analysis*, vol. 15, no. 2, pp. 251–276, 2011.
[21] M. U. ÖNEY and S. PEKER, "The use of artificial neural networks in network intrusion detection: A systematic review," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE, 2018, pp. 1–6.
[22] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *arXiv preprint arXiv:1806.03517*, 2018.
[23] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.
[24] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6.
[25] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.
[26] "Deep learning pruning," https://blog.csdn.net/jacke121/article/details/79450321.
[27] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8609–8613.
[28] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.
[29] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
[30] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *University of Toronto, Technical Report*, 2012.
[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[32] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.
[33] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[34] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.
[35] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.
[36] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 32, 2017.
[37] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[40] M. Denil, B. Shakibi, L. Dinh, N. De Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in neural information processing systems*, 2013, pp. 2148–2156.
[41] B. Reagen, U. Gupta, R. Adolf, M. M. Mitzenmacher, A. M. Rush, G.-Y. Wei, and D. Brooks, "Weightless: Lossy weight encoding for deep neural network compression," *arXiv preprint arXiv:1711.04686*, 2017.
[42] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
[43] M. Sabhnani and G. Serpen, "Why machine learning algorithms fail in misuse detection on kdd intrusion detection data set," *Intelligent data analysis*, vol. 8, no. 4, pp. 403–415, 2004.