IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 5, 2018, accepted May 3, 2018, date of publication May 15, 2018, date of current version July 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2836950

Machine Learning and Deep Learning Methods for Cybersecurity

YANG XIN[1,2], LINGSHUANG KONG[3], ZHI LIU[2,3], (Member, IEEE), YULING CHEN[2], YANMIAO LI[1], HONGLIANG ZHU[1], MINGCHENG GAO[1], HAIXIA HOU[1], AND CHUNHUA WANG[4]

[1]Centre of Information Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China
[3]School of Information Science and Engineering, Shandong University, Jinan 250100, China
[4]China Changfeng Science Technology Industry Group Corporation, Beijing 100854, China

Corresponding author: Zhi Liu (liuzhi@sdu.edu.cn)

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0802300, in part by the Foundation of Guizhou Provincial Key Laboratory of Public Big Data under Grant 2017BDKFJJ015, in part by the Key Research and Development Plan of Shandong Province under Grant 2017CXGC1503 and Grant 2018GSF118228, in part by the Shandong Provincial Natural Science Foundation under Grant ZR2012FZ005, and in part by the Fundamental Research Funds of Shandong University under Grant 2015JC038.

ABSTRACT With the development of the Internet, cyber-attacks are changing rapidly and the cyber security situation is not optimistic. This survey report describes key literature surveys on machine learning (ML) and deep learning (DL) methods for network analysis of intrusion detection and provides a brief tutorial description of each ML/DL method. Papers representing each method were indexed, read, and summarized based on their temporal or thermal correlations. Because data are so important in ML/DL methods, we describe some of the commonly used network datasets used in ML/DL, discuss the challenges of using ML/DL for cybersecurity and provide suggestions for research directions.

INDEX TERMS Cybersecurity, intrusion detection, deep learning, machine learning.

I. INTRODUCTION

With the increasingly in-depth integration of the Internet and social life, the Internet is changing how people learn and work, but it also exposes us to increasingly serious security threats. How to identify various network attacks, particularly not previously seen attacks, is a key issue to be solved urgently.

Cybersecurity is a set of technologies and processes designed to protect computers, networks, programs and data from attacks and unauthorized access, alteration, or destruction [1]. A network security system consists of a network security system and a computer security system. Each of these systems includes firewalls, antivirus software, and intrusion detection systems (IDS). IDSs help discover, determine and identify unauthorized system behavior such as use, copying, modification and destruction [2].

Security breaches include external intrusions and internal intrusions. There are three main types of network analysis for IDSs: misuse-based, also known as signature-based, anomaly-based, and hybrid. Misuse-based detection techniques aim to detect known attacks by using the signatures of these attacks [3]. They are used for known types of attacks without generating a large number of false alarms. However, administrators often must manually update the database rules and signatures. New (zero-day) attacks cannot be detected based on misused technologies.

Anomaly-based techniques study the normal network and system behavior and identify anomalies as deviations from normal behavior. They are appealing because of their capacity to detect zero-day attacks. Another advantage is that the profiles of normal activity are customized for every system, application, or network, therefore making it difficult for attackers to know which activities they can perform undetected. Additionally, the data on which anomaly-based techniques alert (novel attacks) can be used to define the signatures for misuse detectors. The main disadvantage of anomaly-based techniques is the potential for high false alarm rates because previously unseen system behaviors can be categorized as anomalies.

Hybrid detection combines misuse and anomaly detection [4]. It is used to increase the detection rate of known intrusions and to reduce the false positive rate of unknown attacks. Most ML/DL methods are hybrids.

This paper presents a literature review of machine learning (ML) and deep learning (DL) methods for cybersecurity applications. ML/DL methods and some applications of each method in network intrusion detection are described. It focuses on ML and DL technologies for network security, ML/DL methods and their descriptions. Our research aims on standards-compliant publications that use ''machine learning'', ''deep learning'' and cyber as keywords to search on Google Scholar. In particular, the new hot papers are used because they describe the popular techniques.

The purpose of this paper is for those who want to study network intrusion detection in ML/DL.Thus, great emphasis is placed on a thorough description of the ML/DL methods, and references to seminal works for each ML and DL method are provided. Examples are provided concerning how the techniques were used in cyber security.

This paper does not describe all of the different techniques of network anomaly detection; instead, it concentrates only on ML and DL techniques. However, in addition to anomaly detection, signature-based and hybrid methods are depicted.

Patcha and Park [5] discuss technological trends in anomaly detection and identify open problems and challenges in anomaly detection systems and hybrid intrusion detection systems. However, their survey only covers papers published from 2002 to 2006, whereas our survey includes more-recent papers. Unlike Modi *et al.* [6], this review covers the application of ML/DL in various areas of intrusion detection and is not limited to cloud security.

Revathi and Malathi [7] focus on machine-learning intrusion techniques. The authors present a comprehensive set of machine-learning algorithms on the NSL-KDD intrusion detection dataset, but their study only involves a misuse detection context. In contrast, this paper describes not only misuse detection but also anomaly detection.

Sahoo *et al.* [8] present the formal formulation of Malicious URL Detection as a machine-learning task and categorize and review the contributions of studies that address different dimensions of this problem (e.g., feature representation and algorithm design). However, unlike this paper, they do not explain the technical details of the algorithm.

Buczak and Guven [9] focus on machine-learning methods and their applications to intrusion detection. Algorithms such as Neural Networks, Support Vector Machine, Genetic Algorithms, Fuzzy Logics, Bayesian Networks and Decision Tree are described in detail. However, major ML/DL methods such as clustering, Artificial Immune Systems, and Swarm Intelligence are not included. Their paper focuses on network intrusion detection. Through a wired network, attackers must pass multiple layers of firewall and operating system defenses or gain physical access to the network. However, any node can be a target in a wireless network; thus, the network is more vulnerable to malicious attacks and is more difficult to defend than are wired networks.

The ML and DL methods covered in this paper are applicable to intrusion detection in wired and wireless networks. Readers who wish to focus on wireless network protection can refer to essays such as Soni *et al.* [10], which focuses more on architectures for intrusion detection systems that have been introduced for MANETs.

The rest of this survey is organized as follows: Section II focuses on similarities and differences in ML and DL. Section III introduces cyber security datasets used in ML and DL. Section IV describes the methods and related papers for ML and DL in cybersecurity. Section V discusses the research status quo and future direction. Section VI presents conclusions.

II. SIMILARITIES AND DIFFERENCES IN ML AND DL

There are many puzzles about the relationship among ML, DL, and artificial intelligence (AI). AI is a new technological science that studies and develops theories, methods, techniques, and applications that simulate, expand and extend human intelligence [11]. It is a branch of computer science that seeks to understand the essence of intelligence and to produce a new type of intelligent machine that responds in a manner similar to human intelligence. Research in this area includes robotics, computer vision, nature language processing and expert systems. AI can simulate the information process of human consciousness, thinking. AI is not human intelligence, but thinking like a human might also exceed human intelligence.

ML is a branch of AI and is closely related to (and often overlaps with) computational statistics, which also focuses on prediction making using computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. ML is occasionally conflated with data mining [12], but the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning. ML can also be unsupervised and be used to learn and establish baseline behavioral profiles for various entities and then used to find meaningful anomalies [13]. The pioneer of ML, Arthur Samuel, defined ML as a ''field of study that gives computers the ability to learn without being explicitly programmed.'' ML primarily focuses on classification and regression based on known features previously learned from the training data.

DL is a new field in machine-learning research. Its motivation lies in the establishment of a neural network that simulates the human brain for analytical learning. It mimics the human brain mechanism to interpret data such as images, sounds and texts [14].

The concept of DL was proposed by Hinton [15] based on the deep belief network (DBN), in which an unsupervised greedy layer-by-layer training algorithm is proposed that provides hope for solving the optimization problem of deep structure. Then the deep structure of a multi-layer automatic encoder is proposed. In addition, the convolution neural network proposed by LeCun *et al.* [16] is the first real multi-layer structure learning algorithm that uses a space relative relationship to reduce the number of parameters to improve the training performance.

DL is a machine-learning method based on characterization of data learning. An observation, such as an image, can be expressed in a variety of ways, such as a vector of each pixel intensity value, or more abstractly as a series of edges, a region of a particular shape, or the like. Using specific representations makes it easier to learn tasks from instances. Similarly to ML methods, DL methods also have supervised learning and unsupervised learning. Learning models built under different learning frameworks are quite different. The benefit of DL is the use of unsupervised or semi-supervised feature learning and hierarchical feature extraction to efficiently replace features manually [17].

The differences between ML and DL include the following:

• Data dependencies. The main difference between deep learning and traditional machine learning is its performance as the amount of data increases. Deep learning algorithms do not perform as well when the data volumes are small, because deep learning algorithms require a large amount of data to understand the data perfectly. Conversely, in this case, when the traditional machine-learning algorithm uses the established rules, the performance will be better [14].

• Hardware dependencies. The DL algorithm requires many matrix operations. The GPU is largely used to optimize matrix operations efficiently. Therefore, the GPU is the hardware necessary for the DL to work properly. DL relies more on high-performance machines with GPUs than do traditional machine-learning algorithms [18].

• Feature processing. Feature processing is the process of putting domain knowledge into a feature extractor to reduce the complexity of the data and generate patterns that make learning algorithms work better. Feature processing is time-consuming and requires specialized knowledge. In ML, most of the characteristics of an application must be determined by an expert and then encoded as a data type. Features can be pixel values, shapes, textures, locations, and orientations. The performance of most ML algorithms depends upon the accuracy of the features extracted. Trying to obtain high-level features directly from data is a major difference between DL and traditional machine-learning algorithms [17]. Thus, DL reduces the effort of designing a feature extractor for each problem.

• Problem-solving method. When applying traditional machine-learning algorithms to solve problems, traditional machine learning usually breaks down the problem into multiple sub-problems and solves the sub-problems, ultimately obtaining the final result. In contrast, deep learning advocates direct end-to-end problem solving.

• Execution time. In general, it takes a long time to train a DL algorithm because there are many parameters in the DL algorithm; therefore, the training step takes longer. The most advanced DL algorithm, such as ResNet, takes exactly two weeks to complete a training session, whereas ML training takes relatively little time, only seconds to hours. However, the test time is exactly the opposite. Deep learning algorithms require very little time to run during testing. Compared with some ML algorithms, the test time increases as the amount of data increases. However, this point does not apply to all ML algorithms, because some ML algorithms have short test times.

• Interpretability. Crucially, interpretability is an important factor in comparing ML with DL. DL recognition of handwritten numbers can approach the standards of people, a quite amazing performance. However, a DL algorithm will not tell you why it provides this result [14]. Of course, from a mathematical point of view, a node of a deep neural network is activated. However, how should neurons be modeled and how do these layers of neurons work together? Thus, it is difficult to explain how the result was generated. Conversely, the machine-learning algorithm provides explicit rules for why the algorithm chooses so; therefore, it is easy to explain the reasoning behind the decision.

An ML method primarily includes the following four steps [12]:

• Feature Engineering. Choice as a basis for prediction (attributes, features).

• Choose the appropriate machine learning algorithm. (Such as classification algorithm or regression algorithm, high complexity or fast)

• Train and evaluate model performance. (For different algorithms, evaluate and select the best performing model.)

• Use the trained model to classify or predict the unknown data.

The steps of a DL approach are similar to ML, but as mentioned above, unlike machine-learning methods, its feature extraction is automated rather than manual. Model selection is a constant trial and error process that requires a suitable ML/DL algorithm for different mission types. There are three types of ML/DL approaches: supervised, unsupervised and semi-supervised. In supervised learning, each instance consists of an input sample and a label. The supervised learning algorithm analyzes the training data and uses the results of the analysis to map new instances. Unsupervised learning is a machine-learning task that deduces the description of hidden structures from unlabeled data. Because the sample is unlabeled, the accuracy of the algorithm's output cannot be evaluated, and only the key features of the data can be summarized and explained. Semi-supervised learning is a means of combining supervised learning with unsupervised learning. Semi-supervised learning uses a large amount of unlabeled data when using labeled data for pattern recognition. Using semi-supervised learning can reduce label efforts while achieving high accuracy.

Commonly used ML algorithms include for example KNN, SVM, Decision Tree, and Bayes. The DL model includes for example DBM, CNN, and LSTM. There are many parameters such as the number of layers and nodes to choose, but also to improve the model and integration. After the training is complete, there are alternative models that must be evaluated on different aspects.

The evaluation model is a very important part of the machine-learning mission. Different machine-learning

missions have various evaluation indicators, whereas the same type of machine-learning missions also have different evaluation indicators, each with a different emphasis such as classification, regression, clustering and the like [19]. The confusion matrix is a table that describes the classification results in detail, whether they are correctly or incorrectly classified and different classes are distinguished, for a binary classification, a $2 * 2$ matrix, and for n classification, an $n * n$ matrix [20].

For a binary classification as shown in Table 1, the results can be divided into four categories:

TABLE 1. Confusion matrix.

	Predicted as Positive	Predicted as Negative
Labeled as Positive	True Positive(TP)	False Negative(FN)
Labeled as Negative	False Positive(FP)	True Negative(TN)

- True Positive (TP): Positive samples correctly classified by the model;
- False Negative (FN): A positive sample that is mis- classified by the model;
- False Positive (FP): A negative samples that is misclassified by the model;
- True Negative (TN): Negative samples correctly classified by the model;

Further, the following metrics can be calculated from the confusion matrix:

- Accuracy: $(TP + TN)/ (TP + TN + FP + FN)$. Ratio of the number of correctly classified samples to the total number of samples for a given test data set. When classes are balanced, this is a good measure; if not, this metric is not very useful.
- Precision: $TP/ (TP + FP)$. It calculates the ratio of all ''correctly detected items'' to all ''actually detected items''.
- Sensitivity or Recall or True Positive Rate (TPR): $TP/ (TP + FN)$. It calculates the ratio of all ''correctly detected items'' to all ''items that should be detected''.
- False Negative Rate (FNR): $FN/(TP + FN)$. The ratio of the number of misclassified positive samples to the number of positive samples.
- False Positive Rate (FPR): $FP/(FP + TN)$. The ratio of the number of misclassified negative samples to the total number of negative samples.
- True Negative Rate (TNR): $TN/(TN + FN)$. The ratio of the number of correctly classified negative samples to the number of negative samples.
- F1-score: $2 * TP/(2 * TP + FN + FP)$. It calculates the harmonic mean of the precision and the recall.
- ROC: In ROC space, the abscissa for each point is FPR and the ordinate is TPR, which also describes the trade-off of the classifier between TP and FP. ROC's main analysis tool is a curve drawn in ROC space - the ROC curve.
- AUC: The value of AUC is the size of the area under the ROC curve. In general, AUC values range from 0.5 to 1.0, and larger AUCs represent better performance.

In the area of cybersecurity, the metrics commonly used in assessment models are precision, recall, and F1-score. The higher and better the precision and recall of model tests are, the better, but in fact these two are in some cases contradictory and can only be emphatically balanced according to the task needs. The F1-score is the harmonic average of precision and recall, considering their results. In general, the higher the F1-score, the better the model will perform.

III. NETWORK SECURITY DATA SET

Data constitute the basis of computer network security research. The correct choice and reasonable use of data are the prerequisites for conducting relevant security research. The size of the dataset also affects the training effects of the ML and DL models. Computer network security data can usually be obtained in two ways: 1) directly and 2) using an existing public dataset. Direct access is the use of various means of direct collection of the required cyber data, such as through Win Dump or Wireshark software tools to capture network packets. This approach is highly targeted and suitable for collecting short-term or small amounts of data, but for long-term or large amounts of data, acquisition time and storage costs will escalate. The use of existing network security datasets can save data collection time and increase the efficiency of research by quickly obtaining the various data required for research. This section will introduce some of the Security datasets that are accessible on the Internet and facilitate section IV of the research results based on a more comprehensive understanding.

A. DARPA INTRUSION DETECTION DATA SETS

DARPA Intrusion Detection Data Sets [21], which are under the direction of DARPA and AFRL/SNHS, are collected and published by The Cyber Systems and Technology Group (formerly the DARPA Intrusion Detection Evaluation Group) of MIT Lincoln Laboratory for evaluating computer network intrusion detection systems.

The first standard dataset provides a large amount of background traffic data and attack data. It can be downloaded directly from the website. Currently, the dataset primarily includes the following three data subsets:

· 1998 DARPA Intrusion Detection Assessment Dataset: Includes 7 weeks of training data and 2 weeks of test data.

· 1999 DARPA Intrusion Detection Assessment Dataset: Includes 3 weeks of training data and 2 weeks of test data.

· 2000 DARPA Intrusion Detection Scenario-Specific Dataset: Includes LLDOS 1.0 Attack Scenario Data, LLDOS 2.0.2 Attack scenario data, Windows NT attack data.

B. KDD CUP 99 DATASET

The KDD Cup 99 dataset [22] is one of the most widely used training sets; it is based on the DARPA 1998 dataset. This dataset contains 4 900 000 replicated attacks on record.

TABLE 2. Features of KDD Cup 99 dataset.

No.	Features	Types	No.	Features	Types
1	duration	Continuous	22	is_guest_login	Symbolic
2	protocol_type	Symbolic	23	count	Continuous
3	service	Symbolic	24	srv_count	Continuous
4	flag	Symbolic	25	serror_rate	Continuous
5	src_bytes	Continuous	26	srv_serror_rate	Continuous
6	dst_bytes	Continuous	27	rerror_rate	Continuous
7	Land	Symbolic	28	srv_rerror_rate	Continuous
8	wrong_fragment	Continuous	29	same_srv_rate	Continuous
9	urgent	Continuous	30	diff_srv_rate	Continuous
10	hot	Continuous	31	drv_diff_host_rate	Continuous
11	num_failed_logins	Continuous	32	dst_host_count	Continuous
12	logged_in	Symbolic	33	dst_host_srv_count	Continuous
13	num_compromised	Continuous	34	dst_host_same_srv_rate	Continuous
14	root_shell	Continuous	35	dst_host_diff_srv_rate	Continuous
15	su_attempted	Continuous	36	dst_host_same_src_port_rate	Continuous
16	num_root	Continuous	37	dst_host_srv_diff_host_rate	Continuous
17	num_file_creations	Continuous	38	dst_host_serror_rate	Continuous
18	num_shells	Continuous	39	dst_host_srv_serror_rate	Continuous
19	num_access_files	Continuous	40	dst_host_rerror_rate	Continuous
20	num_outbound_cmds	Continuous	41	dst_host_srv_rerror_rate	Continuous
21	is_host_login	Symbolic			

There is one type of the normal type with the identity of normal and 22 attack types, which are divided into five major categories: DoS (Denial of Service attacks), R2L (Root to Local attacks), U2R (User to Root attack), Probe (Probing attacks) and Normal. For each record, the KDD Cup 99 training dataset contains 41 fixed feature attributes and a class identifier. Of the 41 fixed feature attributes, seven characteristic properties are the symbolic type; the others are continuous.

In addition, the features include basic features (No.1–No.10), content features (No.11–No.22), and traffic features (No.23–No.41) as shown in Table 2. The testing set has specific attack types that disappear in the training set, which allows it to provide a more realistic theoretical basis for intrusion detection.

To date, the KDD Cup '99 dataset remains the most thoroughly observed and freely available dataset, with fully labeled connection records spanning several weeks of network traffic and a large number of different attacks [23]. Each connection record contains 41 input features grouped into basic features and higher-level features. The basic features are directly extracted or derived from the header information of IP packets and TCP/UDP segments in the tcpdump files of each session. The listfiles for tcpdump from the DARPA training data were used to label the connection records. The so-called content-based higher-level features use domain knowledge to look specifically for attacks in the actual data of the segments recorded in the tcpdump files. These address 'r2l' and 'u2r' attacks, which occasionally either require only a single connection or are without any prominent sequential patterns. Typical features include the number of failed login attempts and whether root access was obtained during the session. Furthermore, there are time-based and connection-based derived features to address 'DoS' and 'probe' attacks. Time-based features examine connections within a time window of two seconds and provide statistics about these. To provide statistical information

about attacks exceeding a two-second time-window, such as slow probing attacks, connection-based features use a connection window of 100 connections. Both are further split into same-host features, which provide statistics about connections with the same destination host, and same-service features, which examine only connections with the same service [24].

The KDD Cup '99 competition provides the training and testing datasets in a full set and also provides a so-called '10%' subset version. The '10%' subset was created due to the huge amount of connection records present in the full set; some 'DoS' attacks have millions of records. Therefore, not all of these connection records were selected. Furthermore, only connections within a time-window of five minutes before and after the entire duration of an attack were added into the '10%' datasets [22]. To achieve approximately the same distribution of intrusions and normal traffic as the original DARPA dataset, a selected set of sequences with 'nor-mal' connections were also left in the '10%' dataset. Training and test sets have different probability distributions. The full training dataset contains nearly five million records. The full training dataset and the corresponding '10%' both contain 22 different attack types in the order that they were used in the 1998 DARPA experiments. The full test set, with nearly three million records, is only available unlabeled; however, a '10%' subset is provided both as unlabeled and labeled test data. It is specified as the 'corrected' subset, with a different distribution and additional attacks not part of the training set. For the KDD Cup '99 competition, the '10%' subset was intended for training. The 'corrected' subset can be used for performance testing; it has over 300,000 records containing 37 different attacks.

C. NSL-KDD DATASET

The NSL-KDD dataset [7] is a new version of the KDD Cup 99 dataset. The NSL-KDD dataset improves some of the limitations of the KDD Cup 99 dataset. The KDD 1999 Cup Dataset Intrusion Detection Dataset was applied to the 3rd International Knowledge Discovery and Data Mining Tools Contest. This model identifies features between intrusive and normal connections for building network intrusion detectors. In the NSL-KDD dataset, each instance has the characteristics of a type of network data. It contains 22 different attack types grouped into 4 major attack types, as shown in Table 3.

The dataset covers the KDDTrain+ dataset as the training set and KDDTest+ and KDDTest−21 datasets as the testing set, which has different normal records and four different types of attack records, as shown in Table 4. The KDDTest−21 dataset is a subset of the KDDTest+ and is more difficult to classify [25].

D. ADFA DATASET

The ADFA data set is a set of data sets of host level intrusion detection system issued by the Australian defence academy (ADFA) [26], which is widely used in the testing of intrusion detection products. In the dataset, various system calls have been characterized and marked for the type of attack. The data set includes two OS platforms, Linux (ADFA-LD) and Windows(ADFA-WD), which record the order of system calls. In the case of ADFA-LD, it records the invocation of operating system for a period of time. Kernel provides the user space program and the kernel space interact with a set of standard interface, the interface to the user program can be restricted access hardware devices, such as the application of system resources, operating equipment, speaking, reading and writing, to create a new process, etc. User space requests, kernel space is responsible for execution, and these interfaces are the bridge between user space and kernel space. ADFA-LD is marked for the attack type, as shown in the figure. Linux system, user space by making system calls to kernel space to produce soft interrupts, so that the program into the kernel state, perform corresponding operations. There is a corresponding system call number for each system call. It contains 5 different attack types and 2 normal types, as shown in Table 5.

TABLE 3. Type of attck in NSL-KDD.

Types of Attack	Attacks in NSL-KDD Training Set
Dos	back, Neptune, smurf, teardrop, land, pod
Probe	Satan, portsweep, ipsweep, nmap.
R2L	Warezmaster, warezclient, ftpwrite, guesspassword, imap, multihop, phf, spy
U2R	Rootkit, butteroverflow, loadmodule, perl.

TABLE 4. Different classifications in the NSL-KDD.

	Total	Normal	Dos	Probe	R2L	U2L
KDD Train^{+}	125973	67343	45927	11656	995	52
KDD Test^{+}	22544	9711	7458	2421	2754	200
KDD Test^{-21}	11850	2152	4342	2402	2754	200

TABLE 5. Type of attck in ADFA-LD.

Attack Type	Data Size	Note Type
Training	833	normal
Validation	4373	normal
Hydra-FTP	162	attack
Hydra-SSH	148	attack
Adduser	91	attack
Java-Meterperter	75	attack
Webshell	118	attack

IV. ML AND DL ALGORITHM FOR CYBERSECURITY

This section is divided into two parts. The first part introduces the application of traditional machine-learning algorithms in network security. The second part introduces the

application of deep learning in the field of cybersecurity. It not only describes the research results but also compares similar studies.

A. SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is one of the most robust and accurate methods in all machine-learning algorithms. It primarily includes Support Vector Classification (SVC) and Support Vector Regression (SVR). The SVC is based on the concept of decision boundaries. A decision boundary separates a set of instances having different class values between two groups. The SVC supports both binary and multi-class classifications. The support vector is the closest point to the separation hyperplane, which determines the optimal separation hyperplane. In the classification process, the mapping input vectors located on the separation hyperplane side of the feature space fall into one class, and the positions fall into the other class on the other side of the plane. In the case of data points that are not linearly separable, the SVM uses appropriate kernel functions to map them into higher dimensional spaces so that they become separable in those spaces [27].

Kotpalliwar and Wajgi [28] choose two representative datasets ''Mixed'' and ''10% KDD Cup 99'' datasets. The RBF is used as a kernel function for SVM to classify DoS, Probe, U2R, and R2L datasets. The study calculates parameter values related to intrusion-detector performance evaluation. The validation accuracy of the ''mixed'' dataset and the classification accuracy of the ''10% KDD'' dataset were estimated to be 89.85% and 99.9%, respectively. Unfortunately, the study did not assess accuracy or recall except for accuracy.

Saxena and Richariya [29] proposed a Hybrid PSO-SVM approach for building IDS. The study used two feature reduction techniques: Information Gain and BPSO. The 41 attributes reduced to 18 attributes. The classification performance was reported as 99.4% on the DoS, 99.3% on Probe or Scan, 98.7% on R2L, and 98.5% on the U2R. The method provides a good detection rate in the case of a Denial of Service (DoS) attack and achieves a good detection rate in the case of U2R and R2L attacks. However, the precision of Probe, U2R and R2L is 84.2%, 25.0% and 89.4%, respectively. In other words, the method provided by the essay leads to a higher false alarm rate.

Pervez and Farid [30] proposes a filtering algorithm based on a Support Vector Machine (SVM) classifier to select multiple intrusion classification tasks on the NSL-KDD intrusion detection dataset. The method achieves 91% classification accuracy using only three input features and 99% classification accuracy using 36 input features, whereas all 41 input features achieve 99% classification accuracy. The method performed well on the training set with an F1-score of 0.99. However, in the test set, the performance is worse; the F1-score is only 0.77. With poor generalization, it cannot effectively detect unknown network intrusions.

Chandrasekhar and Raghuveer [31] integrates fuzzy C-means clustering, an artificial neural network and support vector machine-intrusion detection technology. With the help of the fuzzy C-means clustering technique, the heterogeneous training data are collected into homogeneous subsets, reducing the complexity of each subset, which helps to improve detection accuracy. After the initial clustering, ANNs are trained on the corresponding homogeneous subsets and use the linear SVM classifier to perform the final classification. The experimental results obtained with the calibrated KDD CUP 1999 dataset show the effectiveness of this method.

In the same work, the KDD Cup 99 dataset is divided into 4 subsets according to different intrusion types and trained separately; DoS and PROBE attacks have a higher frequency and can be effortlessly separated from normal activity. In contrast, U2R and R2L attacks are embedded in the data portion of the packet, making it difficult to achieve detection accuracy on both types of attacks. The technique has attained a consistent peak scores for all types of intrusions. Overall accuracy of the DoS, Probe, R2L and U2R categories was 99.66%, 98.55%, 98.99% and 98.81%, respectively. Compared with other reported intrusion detection approaches, this method is better in classification effect, but the trained classifier cannot effectively detect the abnormal in the actual network.

Yan and Liu [32] attempts to use a direct support vector classifier to create a transductive method and introduces the simulated annealing method to degenerate the optimization model. The study used a subset of the DARPA 1998 dataset. For DoS-type attacks, 200 normal samples and 200 attack samples are selected; the feature dimension is 18, and samples are randomly divided into a training set and a test set according to the ratio 6:4. The experimental results show that the accuracy, FPR and precision are 80.1%, 0.47% and 81.2%, respectively. The dataset used in this study is too small, and the classification results are not very satisfactory.

Using the same dataset, Kokila *et al.* [33] focus on DDoS attacks on the SDN controller. A variety of machine-learning algorithms were compared and analyzed. The SVM classifier has a lower false alarm rate and a higher classification accuracy of 0.8% and 95.11%, respectively.

On the basis of a short sequence model, Xie *et al.* [34] applied a class SVM algorithm to ADFA-LD. Due to the short sequence removes duplicate entries, and between the normal and abnormal performed better separability, so the technology can reduce the cost of computing at the same time to achieve an acceptable performance limits, but individual type of attack mode recognition rate is low.

B. K-NEARESTNEIGHBOR

The kNN classifier is based on a distance function that measures the difference or similarity between two instances. The standard Euclidean distance d(x, y) between two instances x and y is defined as:

$$d(x, y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

where, $\mathbf{x_k}$ is the k^{th} featured element of instance $\mathbf{x}$, $\mathbf{y_k}$ is the k^{th} featured element of the instance $\mathbf{y}$ and $\mathbf{n}$ is the total number of features in the dataset.

Assume that the design set for kNN classifier is U. The total number of samples in the design set is S. Let $C = \{C_1, C_2, \ldots C_L\}$ are the L distinct class labels that are available in S. Let $\mathbf{x}$ be an input vector for which the class label must be predicted. Let $\mathbf{y_k}$ denote the k^{th} vector in the design set S. The kNN algorithm is to find the k closest vectors in design set S to input vector $\mathbf{x}$. Then the input vector $\mathbf{x}$ is classified to class C_j if the majority of the k closest vectors have their class as C_j [34].

Rao and Swathi [36] used Indexed Partial Distance Search k-Nearest Neighbor (IKPDS) to experiment with various attack types and different k values (i.e., 3, 5, and 10). They randomly selected 12,597 samples from the NSl-KDD dataset to test the classification results, resulting in 99.6% accuracy and faster classification time. Experimental results show that IKPDS, and in a short time Network Intrusion Detection Systrms(NIDS), have better classification results. However, the study of the test indicators of the experiment is not perfect; it did not consider the precision and recall rate.

Sharifi *et al.* [37] presents the K-Means and kNN combination intrusion detection system. First, the input invasion data (NSL-KDD) are preprocessed by principal component analysis to select the best 10 important features. Then, these preprocessed data are divided into three parts and fed into the k-means algorithm to obtain the clustering centers and labels. This process is completed 20 times to select the best clustering scheme. These cluster centers and labels are then used to classify the input KDD data using simple kNNs. In the experiment, two methods were used to compare the proposed method and the results of kNN. These measures are based on the accuracy of the true detection of attack and attack type or normal mode. Implement two programs to investigate the results. In the first case, the test data are separated from the train data, whereas in the second case, some test data are not substituted from the training data. However, in any case, the average accuracy of the experiment is only approximately 90%, and it did not consider the precision and recall rate.

Shapoorifard and Shamsinejad [38] studied some new techniques to improve the classification performance of KNN in intrusion detection and evaluate their performance on NSL-KDD datasets. The farthest neighbor (k-FN) and nearest neighbor (KNN) are introduced to classify the data. When the farthest neighbor and the nearest neighbor have the same category label, the second nearest neighbor of the data is used for discrimination. Experimental results show that this method has been improved in terms of accuracy, detection rate and reduction of failure alarm rate. Because the experimental results in this paper only provide a histogram, the accuracy of this method can only be roughly estimated. The detection rate and false alarm rate are 99%, 98% and 4%, respectively. However, the study did not identify specific types of attacks in abnormal situations.

To reduce the false alarm rate, Meng *et al.* [39] developed a knowledge-based alarm verification method, designed an intelligent alarm filter based on multi-level k-nearest neighbor classifier and filtered out unwanted alarms. Expert knowledge is a key factor in evaluating alerts and determining rating thresholds. Alert filters classify incoming alerts into appropriate clusters for tagging through expert knowledge rating mechanisms. Experts will further analyze the effect of different classifier settings on classification accuracy in the evaluation of the performance of alarm filters in real datasets and network environments, respectively. Experimental results show that when $K = 5$, the alarm filter can effectively filter out multiple NIDS alarms.

In the same work, the study initially trained the filter using the DARPA 1999 dataset and evaluated it in a network environment built by Snort and Wireshark. Before Snort was deployed on the internal network, Snort was designed to detect various types of network attacks. Wireshark was implemented before Snort and was responsible for recording network packets and providing statistics. KNN-based smart alarm filters are deployed behind Snort to filter Snort alarms. Real-world web traffic will pass through Wireshark and reach Snort. Snort checks network packets and generates alerts. Thereafter, all generated Snort alarms will be forwarded to intelligent kNN-based alarm filters for alarm filtering. Experiments use the accuracy and F-score as indicators; the average of results were 85.2% and 0.824, respectively.

Vishwakarma *et al.* [40] propose a kNN intrusion detection method based on the ant colony optimization algorithm (ACO), pre-training the KDD Cup 99 dataset using ACO, and studies on the performance of kNN-ACO, BP Neural network and support vector machine for comparative analysis with common performance measurement parameters (accuracy and false alarm rate). The overall accuracy reported for the proposed method is 94.17%, and the overall FAR is 5.82%.Unfortunately, the dataset used for this study was small, with only 26,167 samples participating in the training.

Another study [41] used KNN for intrusion detection on the same KDD Cup 99 dataset in an approach similar to that of Vishwakarma *et al.* [40]. The main difference is that the k-NN, SVM, and pdAPSO algorithms are mixed to detect intrusions. The experimental results show that mixing different classifiers can improve classification accuracy. The statistical results show that the classification accuracy is 98.55%. Other than accuracy, the study did not count other indicators.

C. DECISION TREE

A decision tree is a tree structure in which each internal node represents a test on one property and each branch represents a test output, with each leaf node representing a category. In machine learning, the decision tree is a predictive model; it represents a mapping between object attributes and object

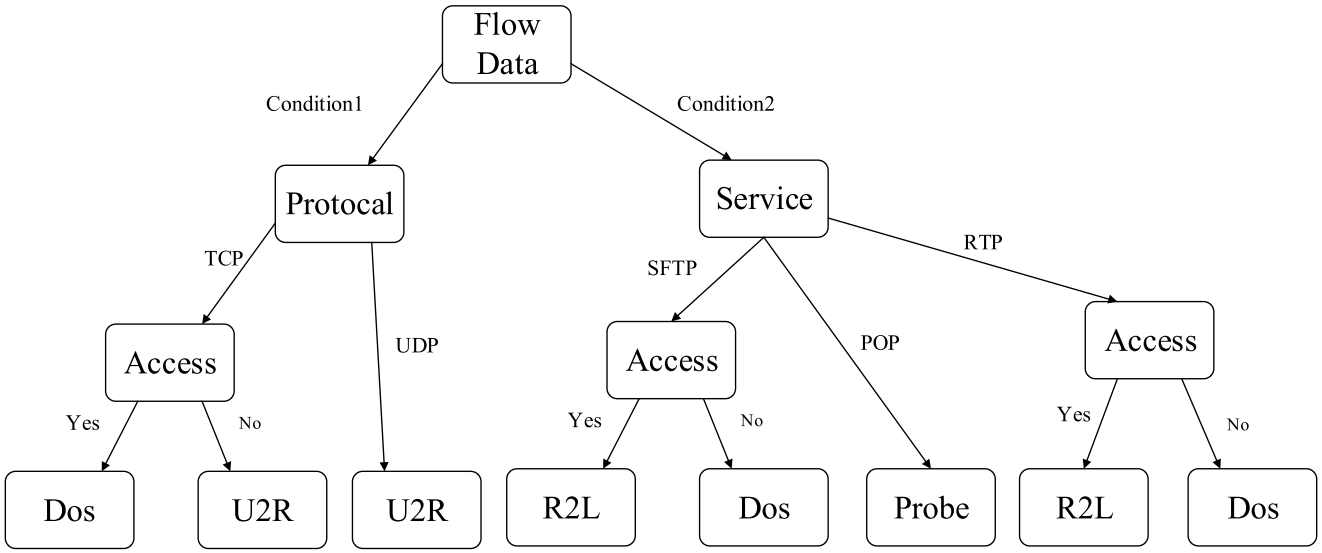

FIGURE 1. An example decision tree.

values. Each node in the tree represents an object, each divergence path represents a possible attribute value, and each leaf node corresponds to the value of the object represented by the path from the root node to the leaf node. The decision tree only has a single output; if you want complex output, you can establish an independent decision tree to handle different outputs. Commonly used decision tree models are ID3, C4.5 and CART.

As shown in Fig.1, the decision tree classifies the samples through the conditions of training, and has better detection accuracy for known intrusion methods, but is not suitable for detection of unknown intrusion.

Ingre *et al.* [42] propose a decision tree-based IDS for the NSL-KDD dataset. Feature selection using a correlation feature selection (CFS) approach, selecting 14 features from each data sample, improves the prediction performance of DTS-based IDS. The performance was evaluated separately for five categories and two categories; overall accuracy was 83.7% and 90.3%, respectively. FAR was 2.5% and 9.7%. The method provided by the paper from the experimental results was not prominent; there is room for improvement.

Malik and Khan [43] attempt to prune the decision tree using a particle swarm optimization (PSO) algorithm and apply it to network intrusion detection problems. Single-objective optimization decision tree pruning (SO-DTP) and multi-objective optimization decision tree pruning (MO-DTP) methods were used for experiments. The experiments were performed on a KDD99 dataset.

From the experimental results, the multi-objective optimization decision tree pruning (MO-DTP) method is most suitable to minimize the size of the entire tree. On average, its tree size is tripled compared with any other tree classifier used. Single Objective Optimization Decision Tree Trimming (SO-DTP) avoids overfitting, resulting in a more generalized tree. By using these generalized trees to classify attacks, significant performance improvement can be observed. The false alarm rate, accuracy and precision of MO-DTP are 13.6%, 96.65 and 99.98, respectively. The false alarm rate, accuracy and precision of SO-DTP were 0.81%, 91.94% and 99.89%. The research considers the binary classification and multi-classification and various parameters and is highly representative.

Relan and Patil [44] propose two techniques for using feature selection, the C4.5 decision tree algorithm and the C4.5 decision tree (with pruning). Train and test classifiers use the KDD Cup 99 and NSL-KDD datasets. Only the discrete value attributes of protocol_type, Service, flag, land, logged_in, is_host_login, is_guest_login and class are considered in the classification process. The experimental results show that C4.5 (with pruning) has higher precision and lower FAR of 98.45% and 1.55% than does the C4.5 decision tree.

Another research [45] used C4.5 for intrusion detection on the NSL-KDD dataset. In this work, feature selection and segmentation values are important issues in building decision trees; the algorithm is designed to solve both of these problems. The information gain is used to select the most relevant features, and the segmentation values are chosen such that the classifier has no bias on the most frequent values. Sixteen attributes were selected as features on the NSL-KDD dataset. The proposed decision tree splitting (DTS) algorithm can be used for signature-based intrusion detection. However, the accuracy of this method is only 79.52%.

Modinat *et al.* [46] is similar to Ingre *et al.* [42]. The difference is the feature selection; they use GainRatio for feature selection. The experiment on the KDD99 dataset

showed an improvement in the performance of the decision tree classifier in some categories of attack, that is, Remote to Local: R2L (98.31% for reduced dataset over 98% for full dataset) and User to Root: U2R (76.92% for reduced dataset over 75% for full dataset)). In the case of Denial of Service (DoS and Normal categories), both methods yielded the same result (100% for both full and reduced datasets). In addition, there were some demeaning results in the category of Probing attack (97.78% for reduced dataset and over 99.49% for full dataset).

Azad and Jha [47] proposes an intrusion detection system based on a C4.5 decision tree and the genetic algorithm. The proposed system solves the problem of small separation in the decision tree, improves the accuracy of classification and reduces the false positive rate. The proposed system is compared with some well-known systems, such as Random Tree [48], Naïve Bayes and Reptree. The results show that the proposed system has better results than the existing system. Training with the KDD Cup 99 dataset yielded the best results, with a 99.89% accuracy rate and a 0.11% FAR.

Balogun and Jimoh [49] propose a hybrid intrusion detection algorithm based on decision tree and k nearest neighbor. The dataset initially passes through the decision tree and generates node information. Node information is based on the rules generated by the decision tree. The node information (as an additional attribute) is sorted by kNN along with the original attribute set to obtain the final output. The KDD Cup 1999 dataset was used on the WEKA tool to perform a performance assessment using a single 10-fold cross-validation technique [50] on a single base classifier (decision tree and kNN) and the proposed hybrid classifier (DT-kNN). Experimental results show that the hybrid classifier (DT-kNN) offers the best results in terms of accuracy and efficiency compared with a single classifier (decision tree and kNN).

In the same study, the proposed hybrid classifier can reach an accuracy of 100% with a false positive rate of 0%. Compared with other NIDSs that also applied KDD Cup 1999 as a dataset, this hybrid classifier showed superior performance in U2R, R2L, DoS and Probe attacks, although it was not the best for U2R and R2L attacks. However, in terms of accuracy, the proposed classifier could obtain the best performance at 100%. This experiment was performed on the 10% KDD Cup 1999 dataset without sampling. Some new attack instances in the test dataset, which never appeared in training, could also be detected by this system. From the results, this method works best but might be an overfit situation.

Snort rule checking is one of the most popular forms of network intrusion detection systems. Ammar [51] presents that Snort priority over real network traffic (real attack) can be optimized in real time through a decision tree classifier in the case of high-speed networks using only three features (protocol, source port and destination port) with an accuracy of 99%. Snort [52] usually sends alert priorities for the common attack classes (34 classes) by default. The decision tree model can adjust the priority. The obtained annotator can provide a useful supplement for an anomaly detection intrusion detection system.

An advanced persistence threat (APT) attack is bringing out large social issues. The APT attack uses social engineering methods to target various systems for intrusion. Moon *et al.* [53] propose a decision tree-based intrusion detection system that can detect the malicious code's behavior information by running malicious code on the virtual machine and analyze the behavior information [54] to detect the intrusion. In addition, it detects the possibility of initial intrusion and minimizes damage by responding quickly to APT attacks. The accuracy of the experiment was 84.7%. The proposed system appears to have less accuracy than do other detection systems. However, considering the detection of APT attacks related to malicious code, the detection accuracy of the system is high.

D. DEEP BELIEF NETWORK

Deep Belief Network (DBN) is a probabilistic generative model consisting of multiple layers of stochastic and hidden variables. The Restricted Boltzmann Machine (RBM) and DBN are interrelated because composing and stacking a number of RBMs enables many hidden layers to train data efficiently through activations of one RBM for further training stages [55]. RBM is a special topological structure of a Boltzmann machine (BM). The principle of BM originated from statistical physics as a modeling method based on an energy function that can describe the high-order interactions between variables. BM is a symmetric coupled random feedback binary unit neural network composed of a visible layer and a plurality of hidden layers. The network node is divided into a visible unit and a hidden unit, and the visible unit and the hidden unit are used to express a random network and a random environment. The learning model expresses the correlation between units by weighting.

In the study, Ding *et al.* [56] apply Deep Belief Nets (DBNs) to detect malware. They use PE files from the internet as samples. DBNs use unsupervised learning to discover multiple layers of features that are then used in a feed-forward neural network and fine-tuned to optimize discrimination. The unsupervised pre-training algorithm makes DBNs less prone to overfitting than feedforward neural networks initialized with random weights. It also makes it easier to train neural networks with many hidden layers.

Because the DBNs can learn from additional unlabeled data, in the experiments, the DBNs produce better classification results than several other widely used learning techniques, outperforming SVM, KNN, and decision tree. The accuracy of the method is approximately 96.1%, but other specifications are not mentioned.

Nadeem *et al.* [57] combine neural networks with semi-supervised learning to achieve good accuracy with only a small number of labeled samples. Experiments using KDD Cup 99 dataset, tracing the non-labeled data through the Ladder Network and then using the DBN to classify the data of

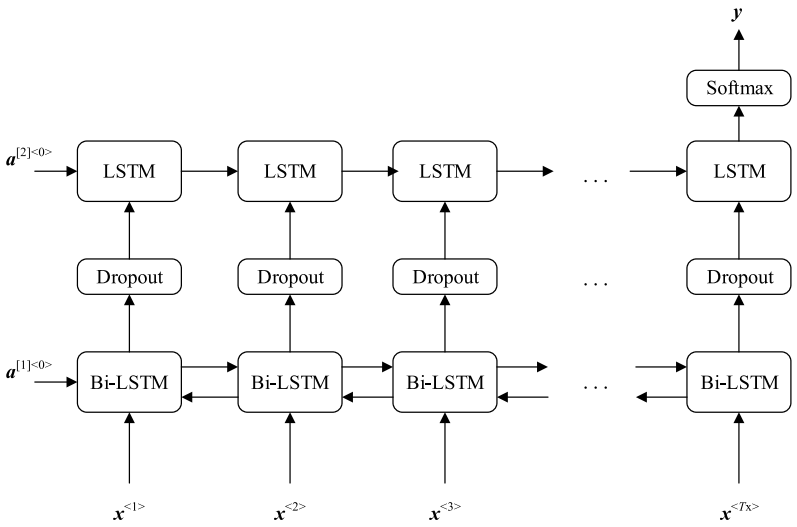

FIGURE 2. An example RNN model structure.

the label obtained detection accuracy similar to the supervised learning, which was 99.18%. However, there was the same problem as with Ding [56]–no calculation of indicators in addition to the accuracy rate.

Gao *et al.* [58] compared different DBN structures, adjusted the number of layers and number of hidden units in the network model, and obtained a four-layer DBN model. The KDD Cup 99 dataset was used for testing. The accuracy, precision and FAR of the model were 93.49%, 92.33% and 0.76%.

Zhao *et al.* [59] aim at the problems of a large amount of redundant information, large amount of data, long training time, and ease of falling into a local optimum in intrusion detection. An intrusion detection method based on deep belief network (DBN) and probabilistic neural network (PNN) is proposed. First, the original data are converted to low-dimensional data, using the DBN nonlinear learning ability to retain the basic attributes of the original data. Second, to obtain the best learning performance, the particle-swarm optimization algorithm is used to optimize the number of hidden nodes in each layer. Next, use the PNN to classify the low-dimensional data. Finally, the KDD CUP 99 dataset was used to test the performance of the above approach. The accuracy, precision and FAR of the experimental results were 99.14%, 93.25% and 0.615%, respectively.

The method Alrawashdeh and Purdy [60] implement is based on a deep belief network using Logistic Regression soft-max for fine-tuning the deep network. The multi-class Logistic Regression layer was trained with 10 epochs on the improved pre-trained data to improve the overall performance of the network. The method achieved a detection rate of 97.9% on the total 10% KDD Cup 99 test dataset and produced a low false negative rate of 2.47%.

After training with 40% NSL-KDD datasets, Alom *et al.* [61] explored the intrusion detection capabilities of DBN through a series of experiments. The trained DBN network can effectively identify unknown attacks provided to it, and the proposed system achieves 97.5% accuracy after 50 iterations.

According to the behavioral characteristics of ad hoc networks, Tan *et al.* [62] design a DBN-based ad hoc network intrusion detection model and conduct a simulation experiment on the NS2 platform. Experimental results show that the DBN detection method can obtain better accuracy and can be applied to Ad hoc network intrusion detection technology. Accuracy and FAR were 97.6% and 0.9%, respectively.

E. RECURRENT NEURAL NETWORKS

The recursive neural network (RNN) is used to process sequence data. In the traditional neural network model, data from the input layer to the hidden layer to the output layer; The layers are fully connected and there is no connection between the nodes between each layer. Many problems exist that this conventional neural network cannot solve.

The reason that RNN is a recurrent neural network is that the current output of a sequence is also related to the output before it. The concrete manifestation is that the network can remember the information of the previous moment and apply it to the calculation of the current output; that is, the nodes between the hidden layers become connected, and the input of the hidden layer includes both the output of the input layer and the last moment hidden layer output. Theoretically, any length of sequence data RNN can be processed. However, in practice, to reduce the complexity, it is often assumed that the current state is only related to the previous states.

Fig 2 shows the RNN timing properties, expanded into a whole network structure here in a multi-layer network. The improved model based on RNN has Long Short Term Memory (LSTM) and a Gated Recurrent Unit (GRU).

Yin *et al.* [63] propose intrusion detection (RNN-IDS) based on a cyclic neural network. The NSL-KDD dataset was used to evaluate the performance of the model in binary

classification and multi-class classification, and the influence of the number of neurons and different learning rates on the performance of the model. The training accuracy and the test accuracy obtained by the model in binary classification are respectively 99.81% and 83.28%. The training accuracy and test accuracy of the multi-classification model are 99.53% and 81.29%, respectively.

In a similar work, Krishnan and Raajan [64] also used RNN for intrusion detection, but the dataset used was the KDD 99 Cup. Experiment accuracy, recall and precision of Probe was 96.6%, 97.8% and 88.3%, respectively; DoS was 97.4%, 97.05% and 99.9%, respectively; U2R was 86.5%, 62.7% and 56.1%, respectively; and R2L are 29.73%, 28.81% and 94.1%.

Staudemeyer [65] implement the LSTM recurrent neural network classifier for intrusion detection data. The results show that the LSTM classifier has certain advantages over the other strong static classifiers in the 10% KDD Cup 99 dataset. These advantages lie in the detection of DoS attacks and Probe, both of which produce unique time series events on the attack categories that generate fewer events. The model-classification accuracy rate reached 93.85%; the FAR was 1.62%.

Kim *et al.* [66] also use LSTM as their model and the 10% KDD Cup 99 as their dataset. They set 80 for the hidden layer size and 0.01 for the learning rate. The paper reported the results as 96.93% accuracy, the average precision as 98.8%, and the average FAR as 10%. Compared with the experimental results of [65], the method obtained a higher false detection rate while obtaining a higher detection rate.

To remedy the issue of high false-alarm rates, Kim *et al.* [67] propose a system-call language-modeling approach for designing LSTM-based host intrusion detection systems. The method consists of two parts: the front-end is for LSTM modeling of system calls in various settings, and the back-end is for anomaly prediction based on an ensemble of thresholding classifiers derived from the front-end. Models were evaluated using the KDD Cup 99 dataset and achieve 5.5% FAR and 99.8% precision.

Le *et al.* [68] compares the effect of six commonly used optimizers on the LSTM intrusion detection model. Experimenting with the KDD Cup 99 dataset, the LSTM RNN model with Nadam optimizer [69] outperforms previous works. Intrusion detection with accuracy is 97.54%, the precision is 98.95%, and the false alarm rate is reasonable at 9.98%.

Gated Recurrent Unit (GRU) and long short-term memory (LSTM) unit are both variants of a recurrent neural network (RNN). Conventionally, like most neural networks, each of the aforementioned RNN variants employs the Softmax function as its final output layer for its prediction and the cross-entropy function for computing its loss. Agarap *et al.* [70] present an amendment to this norm by introducing a linear support vector machine (SVM) as the replacement for Softmax in the final output layer of a GRU model. This proposal is primarily intended for binary classification of intrusion detection using the 2013 network traffic data from the honeypot systems [71] of Kyoto University. Results show that the GRU-SVM model performs relatively higher than does the conventional GRU-Softmax model.

In the paper, the reported training accuracy of 81.54% and testing accuracy of 84.15% posits that the proposed GRU-SVM model has a relatively stronger predictive performance than does the conventional GRU-Softmax model (with training accuracy of 63.07% and testing accuracy of 70.75%).

F. COVOLUTIONAL NEURAL NETWORKS

The recursive neural network (RNN) is used to process sequence data. In the traditional neural network model, data from the input layer to the hidden layer to the output layer; The layers are fully connected and there is no connection between the nodes between each layer. Many problems exist that this conventional neural network cannot solve.

Convolutional Neural Networks (CNN) is a type of artificial neural network that has become a hotspot in the field of speech analysis and image recognition. Its weight-sharing network structure makes it more similar to a biological neural network, thus reducing the complexity of the network model and reducing the number of weights. This advantage is more obvious when the network input is a multi-dimensional image, and the image can be directly used as the input of the network to avoid the complex feature extraction and data reconstruction in the traditional recognition algorithm. The Convolutional Network is a multi-layered sensor specifically designed to recognize two-dimensional shapes that are highly invariant to translation, scaling, tilting, or other forms of deformation [72].

CNN is the first truly successful learning algorithm for training multi-layer network structures, that is, the structure shown in Fig 3. It reduces the number of parameters that must be learned to improve the training performance of the BP algorithm through spatial relationships. As a deep learning architecture, CNN is proposed to minimize the data preprocessing requirements. There are three main means for CNN to reduce network-training parameters: local receptivity, weight sharing and pooling. The most powerful part of CNN is the learning feature hierarchies from large amounts of unlabeled data. Therefore, CNN are quite promising for application in the network intrusion detection field.

To learn useful feature representations automatically and efficiently from large amounts of unlabeled raw network traffic data by using deep learning approaches, Yu *et al.* [73] propose a deep learning approach, called dilated convolutional autoencoders (DCAEs), for the network intrusion detection model, which combines the advantages of stacked autoencoders and CNNs. In essence, the model can automatically learn essential features from large-scale and more-varied unlabeled raw network traffic data consisting of real-world traffic from botnets, web-based malware, exploits, APTs (Advanced Persistent Threats), scans, and normal traffic streams.

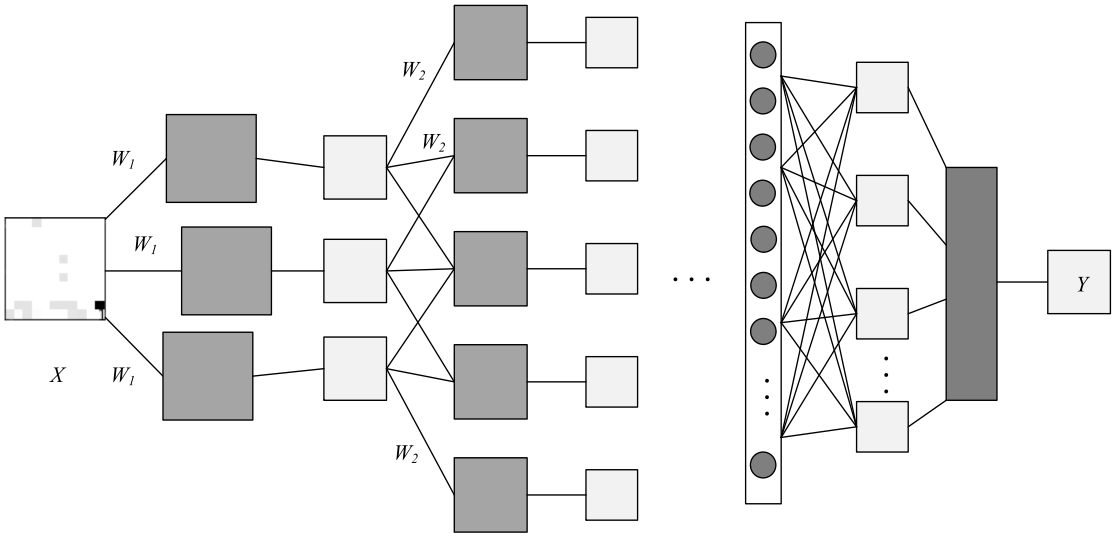

FIGURE 3. An example CNN model structure.

In the same work, Contagio-CTU-UNB datasets and CTU- UNB datasets are created based on various malware traffic data. The classification task is performed to evaluate the performance of the proposed model. The precision, recall and *F*-score of classification tasks were 98.44%, 98.40% and 0.984, respectively.

Kolosnjaji *et al.* [74] shifts performance improvements made in the area of neural networks to modeling the execution sequence of disassembled malicious binary files. A neural network consisting of convolution and feedforward neural structures is implemented. This architecture embodies a hierarchical feature extraction method that combines the features of the n-gram [75] instruction with the simple vectorization of convolution.

In the paper, features are extracted from header files in Portable Executable (PE) files for evaluation only. The results show that the proposed method outperforms the benchmark methods, such as simple feedforward neural network and support vector machine. The F1 score of 0.92 is reached along with a precision and recall of 0.93.

Saxe and Berlin [76] propose an eXpose neural network that uses the depth learning method we have developed to take common raw short strings as input (a common case for security inputs, which include artifacts such as potentially malicious URLs, file paths, named pipes, named mutexes, and registry keys) and learn to extract features and classifications simultaneously using character-level embedding and convolutional neural networks. In addition to fully automating the feature design and extraction process, eXpose also outperformed the baseline based on manual feature extraction for all intrusion detection issues tested, the detection rate was 92% and a decrease in false alarm rate was 0.1%.

Wang *et al.* [77] propose a one-dimensional convolutional neural network end-to-end encrypted traffic classification method. The method integrates feature extraction, feature selection and classifier into a unified end-to-end framework and automatically learns the nonlinear relationship between the original input and the expected output. This method uses a public ISCX VPN-nonVPN traffic dataset for verification.

1D-CNN performed well in 2-class classification with 100% and 99% precision for non-VPN and VPN traffic, respectively. Recall rates for non-VPN and VPN traffic are 99% and 100%, respectively. VPN traffic of 1D-CNN in 6-class and 12-class networks also showed performance of 94.9% and 92.0% precision and recalls of 97.3% and 95.2%, respectively. However, the 1D-CNN performance of non-VPN services is not very good. The precision is only 85.5% and 85.8%; the recall rate is only 85.8% and 85.9%.

Wang *et al.* [78] proposed a malware traffic classification method using a convolutional neural network by taking traffic data as images. This method needed no hand-designed features but directly took raw traffic as input data of the classifier.

In this study, the USTC-TRC2016 flow dataset was established, and the data preprocessing kit USTCTK2016 was developed. Based on the dataset and the toolkit, we found the best type of traffic characterization by analyzing the eight experimental results. Experimental results show the average accuracy of classifiers is 99.41%.

V. DISCUSSION AND FUTURE DIRECTION

Our work examines a large number of academic intrusion detection studies based on machine learning and deep learning as shown in Table 5. In these studies, many imbalances appear and expose some of the problems in this area of research, largely in the following areas: (i) the benchmark datasets are few, although the same dataset is used, and the methods of sample extraction used by each institute vary. (ii) The evaluation metrics are not uniform, many studies only assess the accuracy of the test, and the result is one-sided. However, studies using multi-criteria evaluation often adopt

TABLE 6. ML and DL methods and data use.

Methods	Paper	Data used	Accuracy	Precision	FAR	F1-score
RBF-SVM	Kotpalliwar and Wajgi. [28]	10% KDD Cup 99	99.9%	-	-	-
PSO-SVM	Saxena and Richariya. [29]	KDD Cup 99	99.0%	84.2%	-	-
SVM	Pervez and Farid. [30]	NSL-KDD	82.37%	74%	-	0.77
C-SVM	Chandrasekhar and Raghuveer. [31]	10% KDD Cup 99	98.9%	99.5%	-	0.98
SVM	Yan and Liu. [32]	Part of DARPA 1998	80.1%	81.2%	0.47%	-
SVM	Kokila et al. [33]	DARPA 1998	95.11%	-	0.8%	-
IPDS-KNN	Rao and Swathi. [36]	Part of NSL-KDD	99.6%	-	-	-
Kmeans-KNN	Sharifi et al. [37]	NSL-KDD	90%	-	-	-
kFN-KNN	Shapoorifard and Shamsinejad. [38]	NSL-KDD	99%	98%	4%	-
KNN	Meng et al.[39]	DARPA 1999	85.2%	-	-	0.824
ACO-KNN	Vishwakarma et al.[40]	Part of KDD Cup 99	94.7%	-	5.82%	-
Mix-KNN	Dada [41]	KDD Cup 99	98.55%	-	-	-
CFS-DT	Ingre et al. [42]	NSL-KDD	90.3%	-	9.7%	-
Muti-DTs	Malik and Khan [43]	KDD Cup 99	91.94%	99.89%	0.81%	-
C4.5 DT	Relan and Patil [44]	KDD Cup 99	-	98.45%	1.55%	-
CFS-DT	Modinat et al. [46]	KDD Cup 99	94.6%	-	-	-
GA-C4.5	Azad and Jha [47]	KDD Cup 99	99.89%	-	0.11%	-
DT-KNN	Balogun and Jimoh [49]	KDD Cup 99	100%	-	0%	-
DT	Ammar [51]	Netflow	99%	-	-	-
DT	Moo et al. [53]	Netflow	84.7%	-	-	-
DBN	Ding et al. [56]	Netflow	96.1%	-	-	-
DBN	Nadeem et al. [57]	KDD Cup 99	99.18%	-	-	-
DBN	Gao et al. [58]	KDD Cup 99	93.49%	92.33%	0.76%	-
DBN-PNN	Zhao et al.[59]	KDD Cup 99	99.14%	93.25%	0.62%	-
LR-DBN	Alrawashdeh and Purdy [60]	10% KDD CUP 99	-	97.9%	2.47%	-
DBN	Alom et al.[61]	40% NSL-KDD	97.5%	-	-	-
DBN	Tan et al. [62]	Netflow	97.6%	-	0.9%	-
RNN	Yin et al. [63]	NSL-KDD	83.28%	-	-	-
RNN	Krishnan and Raajan [64]	KDD CUP 99	77.55%	84.6%	-	0.73
LSTM	Staudemeyer [65]	10% KDD CUP 99	93.85%	-	1.62%	-
LSTM	Kim and Jihyun [66]	10% KDD CUP 99	96.93%	98.8%	10%	-
LSTM	Kim et al. [67]	KDD Cup 99	99.8%	-	5.5%	-
LSTM	Le et al. [68]	KDD Cup 99	97.54%	98.95%	9.98%	-
GRU	Agarap [70]	Netflow	84.15%	-	-	-
CNN	Yu et al. [73]	CTU-UNB datasets	-	98.44%	-	0.98
CNN	Kolosnjaji et al. [75]	Netflow	-	93%	-	0.92
CNN	Saxe and Berlin [76]	Netflow	92%	-	0.1%	-
1D-CNN	Wang et al. [77]	ISCX dataset	-	97.3%	-	0.96
CNN	Wang et al. [78]	Netflow	99.41%	-	-	-

ML and DL methods and data use.

different metric combinations such that the research results cannot be compared with one another. (iii) Less consideration is given to deployment efficiency, and most of the research stays in the lab irrespective of the time complexity of the algorithm and the efficiency of detection in the actual network.

In addition to the problem, trends in intrusion detection are also reflected in Table 5. (i) The study of hybrid models has been becoming hot in recent years, and better data metrics are obtained by reasonably combining different algorithms. (ii) The advent of deep learning has made end-to-end learning possible, including handling large amounts of data without human involvement. However, the fine-tuning requires many trials and experience; interpretability is poor. (iii) Papers comparing the performance of different algorithms over time are increasing year by year, and increasing numbers of researchers are beginning to value the practical significance of algorithms and models. (iv) A number of new datasets are in the school's charge, enriching the existing research on cybersecurity issues, and the best of them is likely to be the benchmark dataset in this area.

The problems and trends described above also provide a future for intrusion detection research:

A. DATA SETS

Existing datasets have the defects of old data, redundant information and unbalanced numbers of categories. Although the data can be improved after processing, there is a problem of insufficient data volume. Therefore, establishing network intrusion detection datasets with large amounts of data, wide-type coverage and balanced sample numbers of attack categories becomes a top priority in the field of intrusion detection.

B. HYBRID METHOD

Hybrid detection methods mostly combine machine-learning methods such as those described by [30], [33], [41], whereas intrusion detection with a combination of deep learning and machine-learning methods is less studied. AlphaGo has validated the validity of this idea, which is an exciting research direction.

C. DETECTION SPEED

By reducing the detection time and improving the detection speed from the algorithm and hardware aspects, the algorithm can be used less time given the complexity of the machine-learning algorithm and deep learning algorithm. Hardware can use multiple computers for parallel computing. Combining the two approaches is also an interesting topic.

D. ONLINE LEARNING

The means of network intrusion is increasing day by day. How to fit the new data better with the trained model is also an exciting research direction. At present, transfer learning is a viable means to fine-tune the model with a small amount of labeled data, which should be able to achieve better results in actual network detection.

VI. CONCLUSION

This paper presents a literature review of ML and DL methods for network security. The paper, which has mostly focused on the last three years, introduces the latest applications of ML and DL in the field of intrusion detection. Unfortunately, the most effective method of intrusion detection has not yet been established. Each approach to implementing an intrusion detection system has its own advantages and disadvantages, a point apparent from the discussion of comparisons among the various methods. Thus, it is difficult to choose a particular method to implement an intrusion detection system over the others.

Datasets for network intrusion detection are very important for training and testing systems. The ML and DL methods do not work without representative data, and obtaining such a dataset is difficult and time-consuming. However, there are many problems with the existing public dataset, such as uneven data, outdated content and the like. These problems have largely limited the development of research in this area.

Network information update very fast, which brings to the DL and ML model training and use with difficulty, model needs to be retrained long-term and quickly. So incremental learning and lifelong learning will be the focus in the study of this field in the future.

ACKNOWLEDGMENT

(Yang Xin and Lingshuang Kong contributed equally to this work.)

REFERENCES

[1] S. Aftergood, "Cybersecurity: The cold war online," *Nature*, vol. 547, no. 7661, pp. 30–31, Jul. 2017.

[2] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, "Evaluating computer intrusion detection systems: A survey of common practices," *ACM Comput. Surv.*, vol. 48, no. 1, pp. 1–41, 2015.

[3] C. N. Modi and K. Acha, "Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: A comprehensive review," *J. Supercomput.*, vol. 73, no. 3, pp. 1192–1234, 2017.

[4] E. Viegas, A. O. Santin, A. França, R. Jasinski, V. A. Pedroni, and L. S. Oliveira, "Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 163–177, Jan. 2017.

[5] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Netw.*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007.

[6] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in Cloud," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, 2013.

[7] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," in *Proc. Int. J. Eng. Res. Technol.*, 2013, pp. 1848–1853.

[8] D. Sahoo, C. Liu, and S. C. H. Hoi. (2017). "Malicious URL detection using machine learning: A survey." [Online]. Available: https://arxiv.org/abs/1701.07179

[9] 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.

[10] M. Soni, M. Ahirwa, and S. Agrawal, "A survey on intrusion detection techniques in MANET," in *Proc. Int. Conf. Comput. Intell. Commun. Netw.*, 2016, pp. 1027–1032.

[11] R. G. Smith and J. Eckroth, "Building AI applications: Yesterday, today, and tomorrow," *AI Mag.*, vol. 38, no. 1, pp. 6–22, 2017.

[12] P. Louridas and C. Ebert, "Machine learning," *IEEE Softw.*, vol. 33, no. 5, pp. 110–115, Sep./Oct. 2016.

[13] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[15] G. E. Hinton, "Deep belief networks," *Scholarpedia*, vol. 4, no. 5, p. 5947, 2009.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[17] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, Jun. 2014.

[18] I. M. Coelho, V. N. Coelho, E. J. da S. Luz, L. S. Ochi, F. G. Guimarães, and E. Rios, "A GPU deep learning metaheuristic based model for time series forecasting," *Appl. Energy*, vol. 201, no. 1, pp. 412–418, 2017.

[19] I. Žliobaitė, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Mach. Learn.*, vol. 98, no. 3, pp. 455–482, 2015.

[20] J. N. Goetz, A. Brenning, H. Petschko, and P. Leopold, "Evaluating machine learning and statistical prediction techniques for landslide susceptibility modeling," *Comput. Geosci.*, vol. 81, no. 3, pp. 1–11, 2015.

[21] R. P. Lippmann *et al.*, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proc. DARPA Inf. Survivability Conf. Expo. (DISCEX)*, vol. 2, 2000, pp. 12–26 .

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

[23] G. Meena and R. R. Choudhary, "A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA," in *Proc. Int. Conf. Comput., Commun. Electron.*, 2017, pp. 553–558.

[24] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to KDD CUP 99 dataset," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5947–5957, 2011.

[25] S. Lakhina, S. Joseph, and B. Verma, "Feature reduction using principal component analysis for effective anomaly–based intrusion detection on NSL-KDD," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 6, pp. 3175–3180, 2010.

[26] M. Xie, J. Hu, X. Yu, and E. Chang, "Evaluating host-based anomaly detection systems: Application of the frequency-based algorithms to ADFA-LD," in *Proc. Int. Conf. Netw. Syst. Secur.*, 2014, pp. 542–549.

[27] R. K. Sharma, H. K. Kalita, and P. Borah, "Analysis of machine learning techniques based intrusion detection systems," in *Proc. Int. Conf. Adv. Comput., Netw., Inform.*, 2016, pp. 485–493.
[28] M. V. Kotpalliwar and R. Wajgi, "Classification of attacks using support vector machine (SVM) on KDDCUP'99 IDS database," in *Proc. Int. Conf. Commun. Syst. Netw. Technol.*, 2015, pp. 987–990.
[29] H. Saxena and V. Richariya, "Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information gain," *Int. J. Comput. Appl.*, vol. 98, no. 6, pp. 25–29, 2014.
[30] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD CUP 99 dataset employing SVMs," in *Proc. 8th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, 2014, pp. 1–6.
[31] A. M. Chandrasekhar and K. Raghuveer, "Confederation of FCM clustering, ANN and SVM techniques to implement hybrid NIDS using corrected KDD CUP 99 dataset," in *Proc. Int. Conf. Commun. Signal Process.*, 2014, pp. 672–676.
[32] M. Yan and Z. Liu, "A new method of transductive SVM-based network intrusion detection," in *Proc. IFIP TC Conf.*, Nanchang, China, Oct. 2010, pp. 87–95.
[33] R. T. Kokila, S. T. Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *Proc. 6th Int. Conf. Adv. Comput.*, 2015, pp. 205–210.
[34] M. Xie, J. Hu, and J. Slay, "Evaluating host-based anomaly detection systems: Application of the one-class SVM algorithm to ADFA-LD," in *Proc. 11th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, 2014, pp. 978–982.
[35] X. U. Peng and F. Jiang, "Network intrusion detection model based on particle swarm optimization and k-nearest neighbor," *Comput. Eng. Appl.*, vol. 4, no. 5, pp. 31–38, 2014.
[36] B. B. Rao and K. Swathi, "Fast kNN classifiers for network intrusion detection system," *Indian J. Sci. Technol.*, vol. 10, no. 14, pp. 1–10, 2017.
[37] A. M. Sharifi, S. A. Kasmani, and A. Pourebrahimi, "Intrusion detection based on joint of K-means and KNN," *J. Converg. Inf. Technol.*, vol. 10, no. 5, pp. 42–51, 2015.
[38] H. Shapoorifard and P. Shamsinejad, "Intrusion detection using a novel hybrid method incorporating an improved KNN," *Int. J. Comput. Appl.*, vol. 173, no. 1, pp. 5–9, 2017.
[39] W. Meng, W. Li, and L.-F. Kwok, "Design of intelligent KNN-based alarm filter using knowledge-based alert verification in intrusion detection," *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 3883–3895, 2015.
[40] S. Vishwakarma, V. Sharma, and A. Tiwari, "An intrusion detection system using KNN-ACO algorithm," *Int. J. Comput. Appl.*, vol. 171, no. 10, pp. 18–23, 2017.
[41] E. G. Dada, "A hybridized SVM-kNN-pdAPSO approach to intrusion detection system," in *Proc. Fac. Seminar Ser.*, 2017, pp. 14–21.
[42] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for NSL-KDD dataset," in *Proc. Int. Conf. Inf. Commun. Technol. Intell. Syst.*, 2017, pp. 207–218.
[43] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection," *Clust. Comput.*, vol. 2, no. 3, pp. 1–14, Jul. 2017.
[44] N. G. Relan and D. R. Patil, "Implementation of network intrusion detection system using variant of decision tree algorithm," in *Proc. Int. Conf. Nascent Technol. Eng. Field*, 2015, pp. 1–5.
[45] K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," *Int. J. Adv. Netw. Appl.*, vol. 7, no. 4, pp. 2828–2834, 2016.
[46] A. M. Modinat, G. A. Abimbola, O. B. Abdullateef, and A. Opeyemi, "Gain ratio and decision tree classifier for intrusion detection," *Int. J. Comput. Appl.*, vol. 126, no. 1, pp. 8887–8975, 2015.
[47] C. Azad and V. K. Jha, "Genetic algorithm to solve the problem of small disjunct in the decision tree based intrusion detection system," *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 8, pp. 56–71, 2015.
[48] S. Puthran and K. Shah, "Intrusion detection using improved decision tree algorithm with binary and quad split," in *Proc. Int. Symp. Secur. Comput. Commun.*, 2016, pp. 427–438.
[49] A. O. Balogun and R. G. Jimoh, "Anomaly intrusion detection using an hybrid of decision tree and K-nearest neighbor," *J. Adv. Sci. Res. Appl.*, vol. 2, no. 1, pp. 67–74, 2015.
[50] M. A. Iniesta-Bonillo, R. Sánchez-Fernández, and D. Jiménez-Castillo, "Sustainability, value, and satisfaction: Model testing and cross-validation in tourist destinations," *J. Bus. Res.*, vol. 69, no. 11, pp. 5002–5007, 2016.
[51] A. Ammar, "A decision tree classifier for intrusion detection priority tagging," *J. Comput. Commun.*, vol. 3, no. 4, pp. 52–58, 2015.
[52] R. Selvi, S. S. Kumar, and A. Suresh, "An intelligent intrusion detection system using average manhattan distance-based decision tree," *Adv. Intell. Syst. Comput.*, vol. 324, pp. 205–212, 2015.
[53] D. Moon, H. Im, I. Kim, and J. H. Park, "DTB-IDS: An intrusion detection system based on decision tree using behavior analysis for preventing APT attacks," *J. Supercomput.*, vol. 73, no. 7, pp. 2881–2895, 2017.
[54] S. Jo, H. Sung, and B. Ahn, "A comparative study on the performance of intrusion detection using decision tree and artificial neural network models," *J. Korea Soc. Digit. Ind. Inf. Manage.*, vol. 11, no. 4, pp. 33–45, 2015.
[55] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Clust. Comput.*, vol. 4, no. 3, pp. 1–13, Sep. 2017.
[56] Y. Ding, S. Chen, and J. Xu, "Application of deep belief networks for opcode based malware detection," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 3901–3908.
[57] M. Nadeem, O. Marshall, S. Singh, X. Fang, and X. Yuan, "Semi-supervised deep neural network for network intrusion detection," in *Proc. KSU Conf. Cybersecur. Educ. Res. Pract.*, Oct. 2016, pp. 1–13.
[58] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, 2014, pp. 247–252.
[59] G. Zhao, C. Zhang, and L. Zheng, "Intrusion detection using deep belief network and probabilistic neural network," in *Proc. IEEE Int. Conf. Comput. Sci. Eng.*, vol. 1, Jul. 2017, pp. 639–642.
[60] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. IEEE Int. Conf. Mach. Learn. Appl.*, Dec. 2017, pp. 195–200.
[61] M. Z. Alom, V. R. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proc. Aerosp. Electron. Conf.*, 2016, pp. 339–344.
[62] Q. Tan, W. Huang, and Q. Li, "An intrusion detection method based on DBN in ad hoc networks," in *Proc. Int. Conf. Wireless Commun. Sensor Netw.*, 2016, pp. 477–485.
[63] C. L. Yin, Y. F. Zhu, J. L. Fei, and X. Z. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[64] R. B. Krishnan and N. R. Raajan, "An intellectual intrusion detection system model for attacks classification using RNN," *Int. J. Pharm. Technol.*, vol. 8, no. 4, pp. 23157–23164, 2016.
[65] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South Afr. Comput. J.*, vol. 56, no. 1, pp. 136–154, 2015.
[66] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service*, 2016, pp. 1–5.
[67] G. Kim, H. Yi, J. Lee, Y. Paek, and S. Yoon. (2016). "LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems." [Online]. Available: https://arxiv.org/abs/1611.01726
[68] T.-T.-H. Le, J. Kim, and H. Kim, "An effective intrusion detection classifier using long short-term memory with gradient descent optimization," in *Proc. Int. Conf. Platform Technol. Service*, 2017, pp. 1–6.
[69] L. Bontemps, V. L. Cao, J. Mcdermott, and N. A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," in *Proc. Int. Conf. Future Data Secur. Eng.*, 2017, pp. 141–152.
[70] A. F. Agarap. (2017). "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data." [Online]. Available: https://arxiv.org/abs/1709.03082
[71] T. Ergen and S. S. Kozat, "Efficient online learning algorithms based on LSTM neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 4, no. 2, pp. 1–12, 2017.
[72] S.-J. Bu and S.-B. Cho, "A hybrid system of deep learning and learning classifier system for database intrusion detection," in *Hybrid Artificial Intelligent Systems*, 2017, pp. 615–625.
[73] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Secur. Commun. Netw.*, vol. 2, no. 3, pp. 1–10, 2017.

[74] B. Kolosnjaji, G. Eraisha, G. Webster, A. Zarras, and C. Eckert, "Empowering convolutional networks for malware classification and analysis," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 3838–3845.
[75] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *AI 2016: Advances in Artificial Intelligence*, 2016, pp. 137–149.
[76] J. Saxe and K. Berlin. (2017). "eXpose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys." [Online]. Available: https://arxiv.org/abs/1702.08568
[77] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Inform. (ISI)*, Jul. 2017, pp. 43–48.
[78] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw.*, 2017, pp. 712–717.

YANG XIN received the Ph.D. degree in information security from the Beijing University of Posts and Telecommunications (BUPT). He is currently an Associate Professor with BUPT, where he serves as the Vice Director of the Beijing Engineering Lab for Cloud Security, Information Security Center. His current research interests include network security and big data security.

LINGSHUANG KONG is currently pursuing the master's degree with the School of Information Science and Engineering, Shandong University. His research field includes pattern recognition, machine learning, and deep learning.

ZHI LIU received the Ph.D. degrees from the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, in 2008. He is currently an Associate Professor with the School of Information Science and Engineering, Shandong University. He is also the Head of the Intelligent Information Processing Group. His current research interests are in applications of computational intelligence to linked multicomponent big data systems, medical image in the neurosciences, multimodal human computer interaction, remote sensing image processing, content based image retrieval, semantic modeling, data processing, classification, and data mining.

YULING CHEN is currently an Associate Professor with the Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guiyang, China. Her recent research interests include cryptography and information safety.

YANMIAO LI is currently pursuing the Ph.D. degree in telecommunications. His main research interests include information security, user cross-domain behavior analysis, and network security.

HONGLIANG ZHU received the Ph.D. degree in information security from the Beijing University of Posts and Telecommunications. He is currently a Lecturer with BUPT, where he serves as the Vice Director of the Beijing Engineering Lab for Cloud Security, Information Security Center. His current research interests include network security and big data security.

MINGCHENG GAO received the master's degree in electronics and communication engineering from Shangdong University in 2011. He is currently pursuing the Ph.D. degree in information security with the Beijing University of Posts and Telecommunications. His main research interests include information security, user cross-domain behavior analysis, and network security.

HAIXIA HOU is currently pursuing the Ph.D. degree in telecommunications. His main research interests include information security, user cross-domain behavior analysis, network, and blockchain.

CHUNHUA WANG is currently a Senior Engineer with the China Changfeng Science Technology Industry Group Corporation.