Digital Communications and Networks 9 (2023) 111–124

Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

Internet of things intrusion detection model and algorithm based on cloud computing and multi-feature extraction extreme learning machine

Haifeng Lin [a], Qilin Xue [a], Jiayin Feng [c], Di Bai [b,*]

[a] Nanjing Forestry University, China
[b] Nanjing Agricultural University, China
[c] Hebei Normal University of Science and Technology, China

ARTICLE INFO

Keywords:
Internet of Things
Cloud Computing
Intrusion Prevention
Intrusion Detection
Extreme Learning Machine

ABSTRACT

With the rapid development of the Internet of Things (IoT), there are several challenges pertaining to security in IoT applications. Compared with the characteristics of the traditional Internet, the IoT has many problems, such as large assets, complex and diverse structures, and lack of computing resources. Traditional network intrusion detection systems cannot meet the security needs of IoT applications. In view of this situation, this study applies cloud computing and machine learning to the intrusion detection system of IoT to improve detection performance. Usually, traditional intrusion detection algorithms require considerable time for training, and these intrusion detection algorithms are not suitable for cloud computing due to the limited computing power and storage capacity of cloud nodes; therefore, it is necessary to study intrusion detection algorithms with low weights, short training time, and high detection accuracy for deployment and application on cloud nodes. An appropriate classification algorithm is a primary factor for deploying cloud computing intrusion prevention systems and a prerequisite for the system to respond to intrusion and reduce intrusion threats. This paper discusses the problems related to IoT intrusion prevention in cloud computing environments. Based on the analysis of cloud computing security threats, this study extensively explores IoT intrusion detection, cloud node monitoring, and intrusion response in cloud computing environments by using cloud computing, an improved extreme learning machine, and other methods. We use the Multi-Feature Extraction Extreme Learning Machine (MFE-ELM) algorithm for cloud computing, which adds a multi-feature extraction process to cloud servers, and use the deployed MFE-ELM algorithm on cloud nodes to detect and discover network intrusions to cloud nodes. In our simulation experiments, a classical dataset for intrusion detection is selected as a test, and test steps such as data preprocessing, feature engineering, model training, and result analysis are performed. The experimental results show that the proposed algorithm can effectively detect and identify most network data packets with good model performance and achieve efficient intrusion detection for heterogeneous data of the IoT from cloud nodes. Furthermore, it can enable the cloud server to discover nodes with serious security threats in the cloud cluster in real time, so that further security protection measures can be taken to obtain the optimal intrusion response strategy for the cloud cluster.

1. Introduction

With the extension and expansion of Internet technology, the Internet of Things (IoT), as an important information technology, promotes the intelligent development of modern society. Intelligent IoT applications are being gradually popularized in daily life and industrial development, such as intelligent logistics, transportation, security, medical treatment, homes, and agriculture. The centralized processing mode of cloud computing is to upload all pending transactions to the cloud for execution, and the cloud uses computing resources, storage capacity, and transmission bandwidth to process all service requests sent by end users. The IoT refers to connecting any object with the network through information sensing equipment and according to the agreed protocol. Objects exchange and communicate information through information media to realize the functions of intelligent identification, positioning, tracking, supervision, and so on. The number of global IoT device connections has increased annually, and according to forecasts, the growth rate will gradually stabilize at 15%. Furthermore, data transmission, processing, and storage will

* Corresponding author.
E-mail addresses: haifeng.lin@njfu.edu.cn (H. Lin), xueqilin@njfu.edu.cn (Q. Xue), baidi000@njau.edu.cn (D. Bai).

https://doi.org/10.1016/j.dcan.2022.09.021
Received 4 August 2021; Received in revised form 20 July 2022; Accepted 25 September 2022
Available online 5 October 2022

add more load to cloud services. Mobile and sensor devices connected to the edge of the Internet are constantly generating new data with diverse data types [1], which will lead to great challenges in the computing resources and transmission bandwidth of traditional network architecture. At the same time, for IoT applications that require low network latency, such as automatic driving, if the traditional calculation is used as the operation model, low latency cannot be achieved, and thus the vehicle cannot make real-time decisions to avoid risk in the case of emergency braking. The tasks to be processed by the terminal equipment require strong real-time performance, i.e., the tasks must complete data transmission, analysis, and processing with minimum delay, which conflicts with the processing method of the traditional computing model. In summary, in this highly information-based era, the IoT faces multiple severe challenges: a large amount of data redundancy, cloud processing capacity bottlenecks, network bandwidth limits, data security and privacy, increased cloud power load, and increased task processing delays [2]. Intrusion detection technology is an active security protection technology. According to the different methods of data analysis, IoT intrusion detection systems can be divided into anomaly intrusion detection and misuse intrusion detection. Anomaly intrusion detection is an attack or malicious behavior that occurs when the host data in the network or the entire network does not conform to normal data specifications. Misuse intrusion detection refers to modeling certain specific attack models, determining the characteristic behavior of the attack, and matching it with the current behavior. A successful match represents malicious behavior [3]. The above problems are caused by limitations in the development of the IoT data processing mode and the increasing requirements for service quality for processing operations. The cloud collaborative network architecture has brought the terminal service quality to a new development stage and promoted fast-paced technology development. The optimization of task scheduling and application module mapping is a key problem in cloud computing, which plays an important role in the performance and power optimization of network architecture. Therefore, designing an efficient scheme and algorithm for IoT intrusion detection, task scheduling, and application module mapping is the key to improving the overall performance of the IoT architecture, which is of great research significance.

The specific contributions of this study include the following:

(1) This study examines security threats to the IoT and discusses research related to IoT intrusion in a cloud computing environment from the perspective of network intrusion. Filtering and processing data through decentralized computing resources can meet the heterogeneous, low-latency, dense access, service and defense requirements of IoT when the volume of data services increases dramatically.
(2) In this study, an intrusion prevention system architecture is developed for the Internet of Things in a cloud computing environment. The architectural hierarchy, defense process and application scenarios of this architecture are discussed, and the principle that the architecture can effectively resist intrusion is described, which lays the foundation for further development of the proposed IoT intrusion prevention algorithm and model.
(3) In this study, an intrusion detection measurement model for IoT in a cloud computing environment is developed. The model is based on the characteristics of intrusion behavior in the cloud computing environment, and intrusion and response policy models are established from the perspective of system defenders and external intruders. The optimal intrusion response policy is obtained by solving the model. The simulation results show that the proposed intrusion response strategy can effectively reduce the intrusion frequency of intruders and improve the revenue of the cloud cluster system.
(4) In this paper, the MFE-ELM algorithm is proposed as an intrusion detection algorithm for cloud nodes to perform intrusion detection tasks. Simulation results show that the MFE-ELM algorithm performs well in cloud computing environments, especially in terms of detection accuracy and robustness.

The remainders of this paper are organized as follows. Section 2 discusses related work and is followed by a discussion of the IoT intrusion detection measurement model in a cloud computing environment in Section 3. The MFE-ELM algorithm is proposed in Section 4. Section 5 presents the simulation experimental results and analysis, and Section 6 concludes the paper with a summary and future research directions.

2. Related work

The Internet Engineering Task Force (IETF) divides an intrusion detection system into four components: event generator, event analyzer, corresponding unit, and event database. The function of the event generator is to obtain events from the entire computing environment and provide this event to the other parts of the system. The event analyzer obtains data through analysis and produces analysis results. The response unit responds to the analysis results and performs strong reactions such as cutting off the connection, changing the file attributes, or simply give an alarm. The event database is the general name of the location where various intermediate and final data are stored. It can be a complex database or a simple text file [4]. As for the classification of IoT intrusion detection, according to the audit data source, IoT intrusion detection systems can be divided into host data-based intrusion detection systems, network data-based intrusion detection systems, and distributed intrusion detection systems. In the Host-based Intrusion Detection System (HIDS), the function of intrusion detection is placed in the host that needs to be protected the most, and the system audit log is used as the source of data to detect whether an intrusion has occurred and produce the necessary response. The advantage of a HIDS is that it has access privileges to the host and can detect and respond to attacks in near real time. However, at the same time, an HIDS also has shortcomings. Because of the limited understanding of network topology when deployed on the tested host, attacks on other hosts without an HIDS cannot be detected. In addition, the detected attack types are limited, and the system cannot provide complete protection [5]. In a Network-based Intrusion Detection System (NIDS), network adapters deployed in key network segments of the network are used to monitor and analyze various data packets transmitted through the network in real time. The network information flow is used as the source of the input data, and the protection object is the operational status of the network. A NIDS analyzes the characteristics of each data packet and matches it with the built-in rules of the system. After an attack is detected, the NIDS creates an attack response by means of an alarm or disconnection [6]. In a Distributed-based Intrusion Detection System (DIDS), network data are obtained through boundary sensors and concentrated in supervisor sensors for correlation and analysis. Its architecture mainly includes three layers: boundary sensors, supervisor sensors and a central console. A boundary sensor is responsible for monitoring security events of traffic in the network, receiving and executing the intrusion detection response requirements from the upper layer, and sending monitoring data to a supervisor sensor. A supervisor sensor is responsible for collecting data from the boundary sensor, simplifying the information by using a local rule filter, and then transmitting it to the central console, which is responsible for managing the cooperative work of various DIDS sensors, analyzing the detection results, and responding [7]. Cloud computing builds services and applications on the server between the user layer and the datacenter. It migrates some functions of the user layer and datacenter, and provides limited distributed computing, storage, and network services. As an important support for the IoT, cloud computing can solve the problems of terminal node request delay, excessive server storage and computing burden, and excessive pressure on network transmission bandwidth. In cloud computing, cloud nodes provide network services for large-scale heterogeneous intelligent devices. A cloud network composed of cloud nodes is dynamic. In other words, user devices may join or leave the

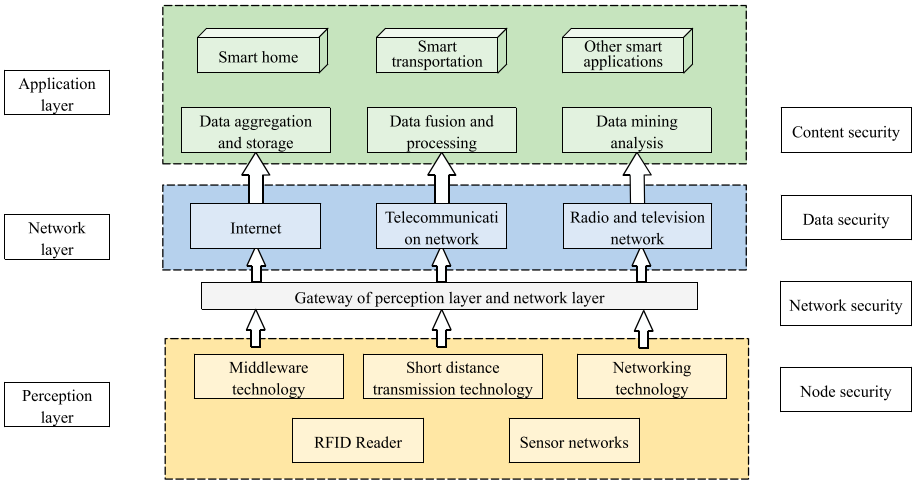


Fig. 1. Security architecture of Internet of Things.

cloud network at any time. Thus, there are dynamic network security threats in cloud networks. To deal with the complex network environment in a flexible manner, a dynamic target training set must be used for real-time training. An Extreme Learning Machine (ELM) can be used to solve the minimum norm of the least squares problem. It can finally be transformed into the Moore-Penrose generalized inverse problem of a matrix. Therefore, the algorithm has the characteristics of requiring fewer training parameters, fast training speed, and strong generalization ability, and is suitable for intrusion algorithms on cloud nodes [8]. To adapt to the complex network environment and dynamic training process of cloud nodes and reduce the training time of cloud nodes, training samples are selected according to the network characteristics and training characteristics of each cloud node.

IoT intrusion analysis involves many methods, including pattern matching, statistical analysis, and integrity analysis. Each method has its own advantages and disadvantages, as well as its own application object and scope. After the United States proposed the concept of intrusion detection in 1980, it attracted much attention in the field of security. The Stanford Research Institute led the way in designing a complete intrusion detection system model, which laid the foundation for the development of intrusion detection. Since then, there have been many advancements in intrusion detection technology, such as intrusion detection technology based on monitoring network information flow and data mining, and machine learning algorithms have been gradually integrated into intrusion detection [9]. At present, the research trend in intrusion detection is the integration of multiple technologies. These include the RNN-IDS model [10], CNN and LSTM hybrid intrusion detection models [11,12], intrusion detection method based on multi-scale convolutional neural networks [13], intrusion detection systems based on KNN super-parameter adjustment and cross verification [14], high-performance methods for DOS attacks and larger volume cores [15], the residual network model S-RESNET [16] suitable for low-dimensional and small-scale datasets, and network intrusion identification technology based on the GRU-MLP model [17]. It can be seen that machine learning has high applicability in intrusion detection and is an important technology to improve the comprehensive performance of intrusion detection systems. Liu et al. [18] combined machine learning with principal component analysis to design an effective method for monitoring the IoT intrusion. This method adopts the mode of high risk to high frequency and low risk to low frequency. Its algorithm can be adjusted according to a change in the monitoring frequency. Thamilarasu et al. [19] built a neural network model to solve data leakage and identity information leakage in the application of the IoT. Roy et al. [20] proposed a new neural network method to effectively learn and detect abnormal conditions in a network, which combines a cyclic neural network with long-term and short-term memory. The selection of the most important feature in the mass intrusion detection process is not the same as the selection of the most effective feature. To effectively solve this problem, Vijayanand et al. [21] combined a genetic algorithm with mutual information technology to design a new intrusion detection method that could identify mixed features, and successfully solved the problems of data duplication and feature independence. Larijani et al. [22] effectively improved the accuracy of model output results by eliminating zero values and meaningless information in the data, so as to make the model converge faster in the training process. Recently, some publications have reported research results on intrusion detection technology in the IoT environment, such as industrial IoT malicious behavior recognition based on an AdS deep learning model [23], an intrusion prevention system based on self-organizing incremental neural networks and SVM [24], IoT intrusion detection deep learning methods based on a BLSTM [25], DNN-DT deep learning model to address the imbalance of network data of industrial control systems [26], intrusion detection of IoT systems based on a CNN [27], DBN and deep self-encoders applied in intrusion detection of industrial IoT sensor systems [28], robust attack detection methods of industrial IoT based on integrated classifiers [29], and trusted privacy protection security frameworks based on deep blockchain in industrial IoT systems [30–32]. Nivaashini et al. [33] combined K-means clustering and vector machine classification into a group of methods and used correlation technology to select features to realize effective modeling based on hybrid machine learning. Khalvati et al. [34], who designed a hybrid model combining k-center point clustering and a vector machine, conducted modeling learning based on the selected feature data, and used naive Bayes to classify the data. Pham et al. [35] used an integration method and feature selection to increase the effectiveness of intrusion detection. With respect to feature selection, the gain rate method is used to reduce redundant data, and then the decision tree algorithm is used to classify data types. Mohammadi et al. [36] proposed a feature selection algorithm based on a supervised filter, which uses multivariate mutual information feature selection as the core and uses the least squares support vector machine as the classifier. Sun et al. [37] proposed an integration model. For a dataset with uneven data volume distribution, it is first transformed into multiple data groups with balanced distribution, and the corresponding classifier is then constructed for the newly obtained data groups through the classification algorithm, and the results are integrated using specific integration rules. In this study, we used a Multi-Feature Extraction Extreme Learning Machine (MFE-ELM) algorithm for cloud computing. In this algorithm, the process of multi-sample feature extraction is added to the cloud server,

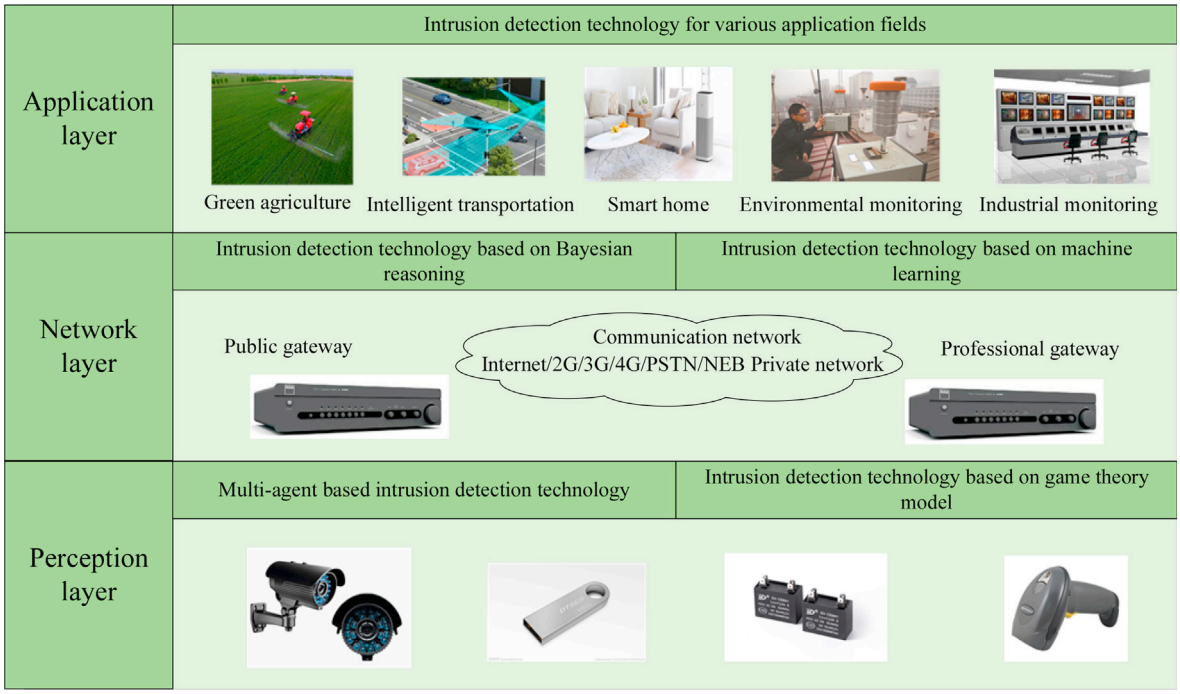


Fig. 2. Internet of Things architecture based on cloud computing.

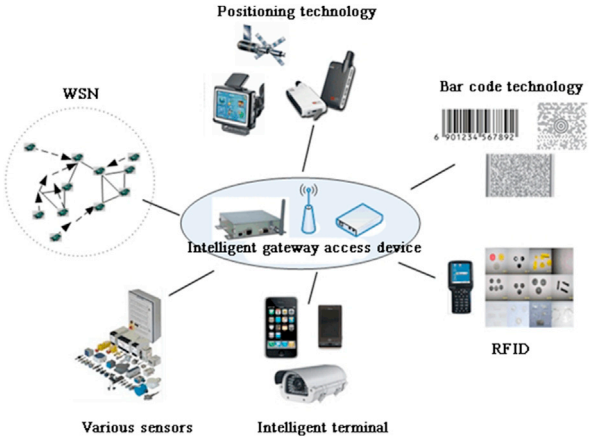


Fig. 3. Perception technology.

and the deployed MFE-ELM algorithm is used on the cloud node to detect and discover the network intrusion to the cloud node. Many threats of the overly use of IoT networked systems are not yet known, there is a very recent work, addressing exactly this issue [38].

3. Internet of things intrusion detection measurement model in cloud computing environments

3.1. Intrusion prevention system architecture of internet of things based on cloud computing

An intrusion response is the strategy and action taken against an intrusion after a cloud node finds and detects the intrusion, so the choice of policy is the most critical problem in intrusion responses. In cloud computing systems, intruders launch cyber attacks on the cloud cluster with the varying frequency with the intent of using higher privileges to initiate access to the servers for a higher level of intrusion. The ECS, as the cloud cluster management system, must respond to this intrusion process. The response policy of the ECS is to set the access prohibition rate for cloud clusters. However, in addition to dealing with illegal users, cloud clusters also need to provide services to as many legitimate users as possible. To analyze the optimal strategies for intruders and systems, the hierarchical structure of the IoT is composed of a perception layer, a network layer, and an application layer, and each layer cooperates with each other to complete the collection, transmission, and processing of information, as shown in Fig. 1. The architecture of the IoT based on cloud computing is shown in Fig. 2. From bottom to top, there are perception, network, and application layers.

3.1.1. Perception layer

The sensing layer of the IoT is mainly composed of sensors, RFID readers, and other sensing terminals. It is not only the information source of the IoT, but also the basis for various expanded applications of it. It is one of the important differences between the IoT and the traditional Internet, as shown in Fig. 3.

Generally, the perception layer of the IoT has the following characteristics:

(a) A large number of nodes. The IoT has a variety of sensing objects and a large demand for monitoring data. Sensing nodes are often deployed in environments with less human contact, such as air, underwater or underground, and the application scenarios are complex and dynamic. Therefore, it is generally necessary to deploy a large number of perception layer nodes to meet the omni-directional and three-dimensional perception requirements.
(b) Various terminal types. The perception layer deploys different types of perception terminals on the same perception node. These terminals have different functions, interfaces, and control modes, resulting in various types and structures of perception layer terminals.
(c) Low safety performance. From the perspective of hardware, due to the poor deployment environment, the sensing layer nodes often face natural or man-made damage. From the perspective of software, due to performance and cost constraints, the sensing nodes do not have a high computing and storage capacity, so they are unable to utilize a security mechanism with high requirements for

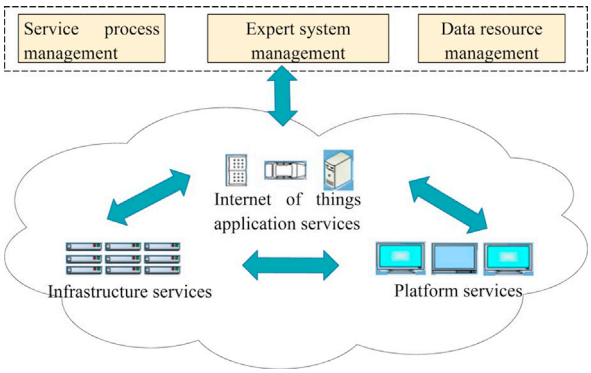


Fig. 4. Dynamic organization and management of data.

computing capacity, resulting in the low security performance of the nodes.

3.1.2. Network layer

The network layer of the IoT is mainly connected with wide-area networks such as mobile communication networks and the Internet through various network access devices to quickly, reliably, and safely transmit the information collected by the perception layer to the information processing layer. It then processes, classifies, and aggregates the information according to different application requirements. The network layer is mainly composed of network infrastructure, network management, and processing systems.

The network layer integrates all existing network forms to build a more extensive "interconnection". Each network has its own characteristics and application scenarios, which can play the greatest role only when combined. Therefore, in practical applications, information can be transmitted through any network or network combination. Fig. 4 shows the dynamic organization and management of the data.

3.1.3. Application layer

The application layer of the IoT is located at the top of the three-tier structure and processes information through the cloud computing platform. Together with the lowest sensing layer, the application layer is the most significant feature and core of the IoT. The application layer can calculate, process, and mine the data collected by the perception layer to realize real-time control, accurate management, and scientific decision-making of the physical world.

Structurally, the IoT application layer includes the following three parts.

(a) IoT middleware. IoT middleware is an independent system software or service program that encapsulates various common functions and provides them to IoT applications.
(b) IoT applications. IoT applications are those directly used by users, such as intelligent control, security, power meter reading, telemedicine, and intelligent agriculture.
(c) Cloud computing. Cloud computing can facilitate the storage and analysis of massive amounts of data in the IoT. According to the service definition of cloud computing, the cloud can be divided into Infrastructure-as-a-Service (IaaS), platform-as-a-Service (PaaS), service, and Software-as-a-Service (SaaS).

3.2. Security analysis of internet of things architecture

3.2.1. Perception layer security

The security of the perception layer is the primary concern in the security of the entire IoT. Compared with traditional communication networks, most of the sensing nodes of the IoT are deployed in an unmanned environment, and their nodes show multi-source heterogeneity. Because the energy and intelligence held by each node are limited, it is impossible to obtain complex security protection capabilities.

The biggest feature of the security technology of the perception layer is that it is "lightweight". Neither cryptographic algorithms nor protocols need to be complex. As a result, the security level of the perception layer is weaker than that of the network and application layers. Therefore, during application, it is necessary to deploy security aggregation devices between the network and perception layers. After the security aggregation devices enhance the security of information, they then exchange with the network layer to compensate for the lack of security capability of the perception layer.

The main security threats to the perceptual layer come from physical capture, denial of service, Trojans, viruses and data disclosure.

(a) Physical capture: The presence of sensing devices outdoors and decentralized installation makes them vulnerable to physical attacks and easy to tamper with information, resulting in a loss of security. With the embedding of RFID (non-contact automatic identification technology) tags and QR codes, user access to the IoT is scanned, tracked and located without control. Malicious users read RFID tag data through legitimate readers, which leads to serious security threats in the process of acquiring and transmitting RFID tag data, resulting in the disclosure of the user's private information. In addition, the rewritability of RFID tags may not guarantee the security, effectiveness, and integrity of the data in the tag.
(b) Denial of service: IoT nodes refuse to provide the service of forwarding data packets in order to save their own energy or to prevent being controlled by trojans, resulting in a sharp decline in network performance. The perception layer is bound to be attacked by the external network (e.g., the Internet) when it is accessed. At present, in addition to illegal access, the main threat is denial-of-service attacks. Because of limited resources and low computing and communication capabilities, the ability to withstand denial of service of the sensing node is relatively weak, which may cause the paralysis of the sensing network.
(c) Trojans and viruses: Because of the cost and inconvenience of security protection measures, some sensing nodes may not take these measures, or take very basic information security protection measures, which may lead to counterfeiting and unauthorized service access problems. For example, when the operating system or application software of the sensing node of the IoT is outdated and the system vulnerabilities cannot be patched in time, problems are prone to occur in object identification, authentication, and control.
(d) Data disclosure: The IoT collects a wide variety and richness of data through a large number of sensing devices. Without proper protection, there can be privacy breaches, fraudulent use or theft of data. If the information received by sensing nodes is unprotected or insufficiently secured, it can be illegally accessed by third parties, causing great harm.

The main security mechanisms adopted by the perception layer include physical security, authentication and authorization, access control, encryption, and secure routing mechanisms as well as key management.

(a) Physical security mechanism: Commonly used RFID tags have low cost and low security. This security mechanism mainly realizes security control by sacrificing the functions of some tags.
(b) Authentication and authorization mechanism: This is mainly used to verify identity and the effectiveness and authenticity of the exchanged data, which mainly includes the authentication and

authorization management between internal nodes and that of nodes to users. In the perception layer, RFID tags need to realize identity authentication through authentication and authorization mechanisms.

(c) Access control mechanism: This is embodied in the user's access control over the node's own information and the data collected by the node, so as to prevent unauthorized users from accessing the perception layer. Common access control mechanisms include mandatory access control, autonomous access control, role-based access control, and attribute-based access control. Encryption mechanisms and key management are the basis of all security mechanisms and are an important means of protection of received information. Key management is the generation, distribution, update, and dissemination of keys. An encryption mechanism is required for the successful operation of the RFID tag authentication mechanism.

(d) Secure routing mechanism: This ensures that when the network is attacked, it can still correctly discover and build routes. It includes data confidentiality and authentication, data integrity and freshness verification, equipment and identity authentication, and routing message broadcast authentication mechanisms.

3.2.2. Network layer security

The IoT network layer also faces major security threats.

(a) Attacks against IoT terminals: With the enhancement of the computing and storage capacity of IoT terminals, the chance of attack from viruses and trojans is also greatly increased. Viruses and Trojan horses are more propagative, more destructive, more concealed and more threatening in IoT. The operating system of the network terminal lacks integrity protection and verification mechanisms, and the software and hardware modules of the device are easily tampered with by attackers. The communication interface inside the terminal lacks confidentiality and integrity protection, and the transmitted information can be easily stolen or tampered.

(b) Attacks on the IoT network information: The basic means to illegally obtain unauthorized data is to disguise as a network entity to steal, tamper or delete data on the link to intercept business data and analyze network traffic.

(c) Attacks on data integrity: The attacker tampers with the service, signaling and control information transmitted in the system's wireless link, including insertion, modification, and deletion of data [39].

(d) Denial of service attacks are divided into physical level interference, protocol-level interference, denial of service disguised as a network entity, etc. Physica-level interference refers to the interference of wireless links through physical means to block normal communication; protocol-level interference refers to interfering with normal communication by inducing failures in specific protocol processes; denial of service masquerading as a network entity refers to an attacker masquerading as a legitimate network entity and refusing to answer a user's service request. For the illegal access attack on a business, the attacker disguises itself as other legitimate users, illegally accesses the network, or cuts between users and the network to carry out intermediate attacks.

(e) Attacks on the core network of the IoT involve the illegal acquisition of data, eavesdropping on user services, tampering with signaling and control data, disguising as a network entity to intercept user information, and active and passive analysis of user traffic, i.e., illegal access to the system data storage.

(f) Attacks on data integrity, including tampering with user services and signaling messages, tampering with applications and data downloaded to user terminals, tampering with user terminals by pretending to be applications and data initiators, and tampering with user data stored in system storage entities.

(g) Denial-of-service attacks, including physical interference, protocol level interference, refusal to answer user requests masquerading as network entities, abuse of emergency services. Denial attacks include denial of cost, denial of sending data, denial of receiving data, etc. Basic ways of illegal access to unauthorized services include pretending to be a user, service network or home network, and abusing privileges to illegally access unauthorized services.

IoT network layer security solutions include building a network security architecture integrating the IoT, Internet, and mobile networks; building a unified protection platform for IoT network security; improving the security application and guaranteeing measures between the application layers of the IoT system; establishing a comprehensive IoT network security access and application access control mechanism.

3.2.3. Security of application layer

The major security threats facing the application layer of IoT include privacy threats, unauthorized access, identity impersonation, information eavesdropping, tampering, repudiation and denial, and replay threats.

(a) Privacy threats: The widespread use of wireless communication, electronic tags and unattended devices in IoT exposes the application layer of IoT to the disclosure of privacy leakage and malicious tracking.

(b) Unauthorized access: Illegal users may use unauthorized services or legitimate users use unauthorized services on the IoT.

(c) Identity impersonation: An attacker may capture or hijack an unattended device, and then impersonate its identity to send data information and perform operations to the client or application server.

(d) Information eavesdropping and tampering: In the heterogeneous and multi-domain network environment of the IoT, the security mechanisms among networks are independent of each other and the application layer data is likely to be eavesdropped, injected, and tampered with.

(e) Repudiation and denial: All participants in the communication may accept or deny the previously sent data and completed operations; an attacker can send the information received by a destination node to cheat the application system.

In the face of multiple security threats, the IoT application layer uses several approaches to secure data and privacy.

(a) Data security limits the legal object of browsing data, prevents the harm caused by unauthorized information disclosure, and ensures the confidentiality of data; to ensure the authenticity of the data, the integrity of the data is verified with the help of a hash function, which provides a tampering proof environment against hardware attacks for the system. The common methods of data integrity detection are digital signature, MAC, and digital watermarking.

(b) Privacy security: Data privacy is protected using probabilistic or statistical methods while ensuring that the statistical or categorical characteristics of the final data remain unchanged; some sensitive and specific information is replaced with non-specific information for privacy protection purposes based on anonymization techniques.

(c) Cloud computing is secure. Cloud computing has the advantages of ubiquitous network access, supercomputing power, and large-capacity information storage capacity. The resource-limited IoT requires an intelligent processing platform in the cloud environment.

4. MFE-ELM algorithm description

In cloud computing, cloud nodes are held responsible for providing network services to large-scale heterogeneous intelligent devices, but the local-area cloud network composed of cloud nodes is dynamically changing. In other words, user devices may join or disconnect a cloud network at any time. Thus, dynamic network safety threats are posed in the cloud network. In order to flexibly respond to the complex network environment, real-time training using dynamic target training sets is required. Many intrusion algorithms take a long time in training. These intrusion detection algorithms are not suitable for cloud computing models due to the limited computing power and storage capacity of cloud nodes. Therefore, this study needs to investigate the deployment and application of intrusion detection algorithms with lightweight, short training time, and high detection accuracy in cloud nodes.

The cloud server carries out real-time automatic response to intrusion behavior, uniformly monitors and manages the security status of cloud nodes, calculates security status measurement, and sets tracking paths; The decision-making and corresponding after the intrusion is detected by the system; Store and manage the data and logs of intrusion cloud nodes to facilitate forensic applications. Intrusion Forensics is the tracing and evidence recording of intrusion behavior. The cloud node monitors the security status of the cloud node by submitting a security log to the cloud server. Its purpose is to enable system managers to grasp the security status of cloud clusters in real time, so as to participate in the decision-making to deal with intrusion. However, when the number of cloud nodes deployed is too large, cloud services need to select the cloud nodes that are seriously threatened by intrusion in the cloud cluster. A proper classification algorithm is a primary factor in deploying cloud computing intrusion prevention systems and a prerequisite for the system to enter intrusion responses and reduce intrusion threats. An ELM provides solutions by seeking the minimum norm of the least square problem, which can eventually be converted to the Moore-Penrose generalized inverse problem of a matrix. Therefore, the algorithm is suitable for cloud node intrusion algorithms with few training parameters, fast training speed, and high generalization capability. To adapt to the complicated network environment and dynamic training process and reduce the training time of cloud nodes, training samples are selected according to the network properties and training characteristics of every cloud node. In this study, a Multi-Feature Extraction Extreme Learning Machine (MFE-ELM) algorithm is used for cloud computing. This algorithm increases the process of sample selection in the cloud server and uses the MFE-ELM algorithm deployed in the cloud nodes to detect and spot the network intrusion into cloud nodes.

4.1. Extreme learning machine (ELM)

ELM does not require iterations to adjust the neural network weights and biases of the nodes in the hidden layer; instead, it can be achieved by directly learning to use the least squares method. Without iteration in the basic training steps, the neural network has a significantly faster learning speed and a reduced possibility of becoming trapped in overfitting.

It is assumed that there are N training data $[\mathbf{x}_i, \mathbf{t}_i] i = 1, 2, \ldots, N$. Where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbf{R}^n$ is the sample, $\mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbf{R}^m$ is the expected output vector of the sample, $\mathbf{x}_j$ and n and m represent the dimensions of the corresponding input and output, respectively. For N training data and l nodes in the hidden layer $l \leq N$, $g(x)$ is the activation function, which is usually the sigmoid type. Therefore, the output mathematical model of the ELM neural network is shown in Equation (1)

$$\sum_{i=1}^{l} \beta_i g(\mathbf{a}_i \cdot \mathbf{x}_j + b_i) = \mathbf{t}_j,\ j = 1, 2, \ldots, N \tag{1}$$

where $\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{in}]^{\mathrm{T}}$ represents the input weight between the nodes in the input layer and the node in the ith hidden layer, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^{\mathrm{T}}$ is the output weight, and b_i is the bias of the hidden layer. Equation (1) can be simplified as

$$\mathbf{H}\beta = \mathbf{T} \tag{2}$$

In Equation (2), the output matrix of the hidden layer is represented by $\mathbf{H}$; the output weight matrix (between the hidden layer and the output layer) is shown as β, and the expected output matrix is indicated as $\mathbf{T}$.

$$\mathbf{H}(\mathbf{a}_i, b_i, \mathbf{x}_j) = \begin{bmatrix} g(\mathbf{a}_1\mathbf{x}_1 + b_1) & \cdots & g(\mathbf{a}_l\mathbf{x}_1 + b_l) \\ \vdots & \ddots & \vdots \\ g(\mathbf{a}_1\mathbf{x}_N + b_1) & \cdots & g(\mathbf{a}_l\mathbf{x}_N + b_l) \end{bmatrix}_{N\times l} \tag{3}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_l^T \end{bmatrix}_{l\times m} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^{\mathbf{T}} \\ \vdots \\ \mathbf{t}_N^{\mathbf{T}} \end{bmatrix}_{N\times m} \tag{4}$$

When $l = N$ is met, i.e., the number of training samples is the same as the number of nodes in the hidden layer, the inverse of matrix $\mathbf{H}$ can be obtained directly from Equation (2), as shown in Equation (4), which has nothing to do with the number of output nodes, and the optimal solution to the output weight matrix β can also be calculated. However, normally $l < N$, meaning that the number of nodes in the hidden layer is always smaller than that of the training samples, the matrix $\mathbf{H}$ is a singular matrix, and it needs to solve Equation (2) with the least squares method.

$$\beta = \mathbf{H}^{\dagger}\mathbf{T} \tag{5}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inversion of the output matrix of the hidden layer.

The above is the solution process of the ELM algorithm, in which the parameters of the hidden layer nodes are determined randomly during the parameter training process. In practical applications, as experimental samples need to be standardized, the values of the parameters of nodes in hidden layers are usually randomly selected within the scope of [−1,1].

4.2. MFE-ELM algorithm description

We divide the total sample set Z_n saved in the cloud server into the sample set distributed by the cloud nodes $Z_{nf}^f = \left\{ z_J^f = \left(x_J^f, y_J^f \right) \right\}_{J=1}^{nf}$ and the alternative sample set $Z_{nc}^c = \left\{ z_J^c = \left(x_J^c, y_J^c \right) \right\}_{J=1}^{nc}$, where $Z_{nf}^f \cup Z_{nc}^c = Z_n Z_{nf}^f \cap Z_{nc}^c = \varnothing$. The purpose of sample selection is to select Z_{nf}^f from Z_n and make the network obtained by Z_{nf}^f learning and based on the ELM algorithm meet $J(a_i, b_i, \beta) \leq \sigma$, where $\sigma \in (0, \min J(a_i, b_i, \beta))$ is the upper limit of the pre-determined performance index.

We assume that $U^c = \left\{ \left| u_j^c \right| u_j^c = \left| y_j^c - f\left(x_j^c, a_i, b_i, \beta \right) \right. \right\}_{j=1}^{nc}$ is the set constituted by the absolute values of the differences in sample output, network input, and network output in Z_{nc}^c. We define $Z_s^c = \left(x_m^c, y_m^c \right)$ as any element of the biggest element Z_{nc}^c corresponding to U^c. Initialize the sample sets $Z_{nf}^f = \varnothing$ and $Z_{nc}^c = Z_n$ and the network structure parameters $a_i = \varnothing$, $b_i = \varnothing$, $\beta = \varnothing$ and adopt the following learning rules

$$\text{Rule1}, Z_{nf}^f = Z_{nf}^f \cup \left\{ z_s^c \right\} \tag{6}$$

$$\text{Rule2}, Z_{nc}^c = Z_{nc}^c - \left\{ z_s^c \right\} \tag{7}$$

$$\text{Rule3}, a_i^{'} = a_i + \frac{\left\{ x_s^c \right\} - \left\{ x_s^c \right\}_{\min}}{\left\{ x_s^c \right\}_{\max} - \left\{ x_s^c \right\}_{\min}} \tag{8}$$

$$b_i^{'} = b_i + \frac{\left\{ x_s^c \right\} - \left\{ x_s^c \right\}_{\min}}{\left\{ x_s^c \right\}_{\max} - \left\{ x_s^c \right\}_{\min}} \tag{9}$$

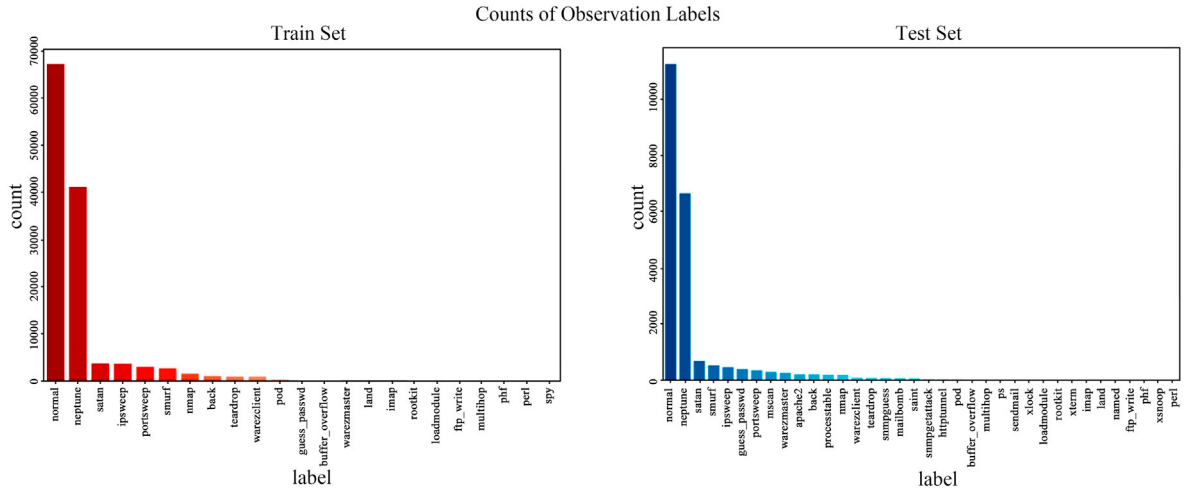


Fig. 5. Dataset label feature distribution.

Rule 4: Calculate and update the optimal external weight $\beta(J(a_i, b_i, \beta) = \sigma)$ according to Equation (6) with Z_{nf}^{f} and a_i and b_i and update Z_{nf}^{f}, Z_{nc}^{c}, a_i, b_i and β. After several iterations, we obtain $J(a_i, b_i, \beta) \leq \sigma$.

The algorithm includes the following steps:

1) Initialize sample sets $Z_{nf}^{f} = \varnothing$ and $Z_{nc}^{c} = Z_n$ in the cloud server and the network structural parameters $a_i = \varnothing$, $b_i = \varnothing$, $\beta = \varnothing$;
2) Randomly generate the parameters in the nodes of the hidden layer $(a_i, b_i), i = 1, \ldots, L$;
3) Calculate the output matrix H of the hidden layer (make sure H color full rank matrix);
4) Calculate $J(a_i, b_i, \beta)$ and U^c;
5) If $J(a_i, b_i, \beta) \leq \sigma$, turn to Step 10), otherwise, continue;
6) Select Z_s^c from Z_n^c, $Z_{nf}^{f} = Z_{nf}^{f} \cup \{z_s^c\}$, $Z_{nc}^{c} = Z_{nc}^{c} - \{z_s^c\}$;
7) Update a_i and b_i;

$$a_i^{'} = a_i + \frac{\{x_s^c\} - \{x_s^c\}_{\min}}{\{x_s^c\}_{\max} - \{x_s^c\}_{\min}} \tag{10}$$

$$b_i^{'} = b_i + \frac{\{x_s^c\} - \{x_s^c\}_{\min}}{\{x_s^c\}_{\max} - \{x_s^c\}_{\min}} \tag{11}$$

8) Use Z_{nf}^{f} as well as a_i and b_i. Calculate and update the optimal external weight $\beta(\text{make} J(a_i, b_i, \beta) = \sigma)$;
9) If $Z_{nc}^{c} \neq \varnothing$, implement Step 4), otherwise, continue;
10) Use Z_n as well as a_i and b_i, calculate and update the optimal external weight β, the algorithm ends.

In this study, the activation function for the hidden layer of the MFE-ELM model uses the sigmoid transformation function

$$G(a_i, b_i, x) = 1/\left[1 + e^{(-a_i x + b_i)}\right] \tag{12}$$

After analyzing the MFE-ELM, it can be learned that when selecting samples for cloud nodes, the most time-intensive process is the calculation of $J(a_i, b_i, \beta)$. Assume that $t(J)$ is the average time to calculate $J(a_i, b_i, \beta)$and that the delay t' of the data transmission between cloud computing and cloud nodes and $t' \ll t(J)$, then the learning time of the cloud node is $t \approx n_L \cdot t(J)$.

5. Simulation experiment

5.1. Environmental preparation

In this study, the experiment was conducted in a Python 3.6 environment under the Windows 10 operating system of Intel Core i7-6500U,

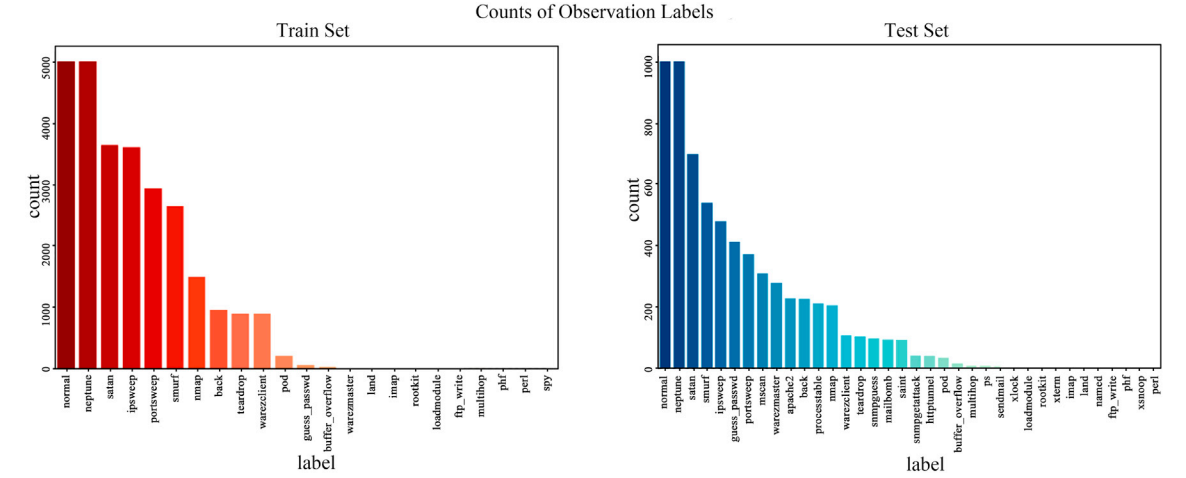


Fig. 6. Dataset label feature distribution.

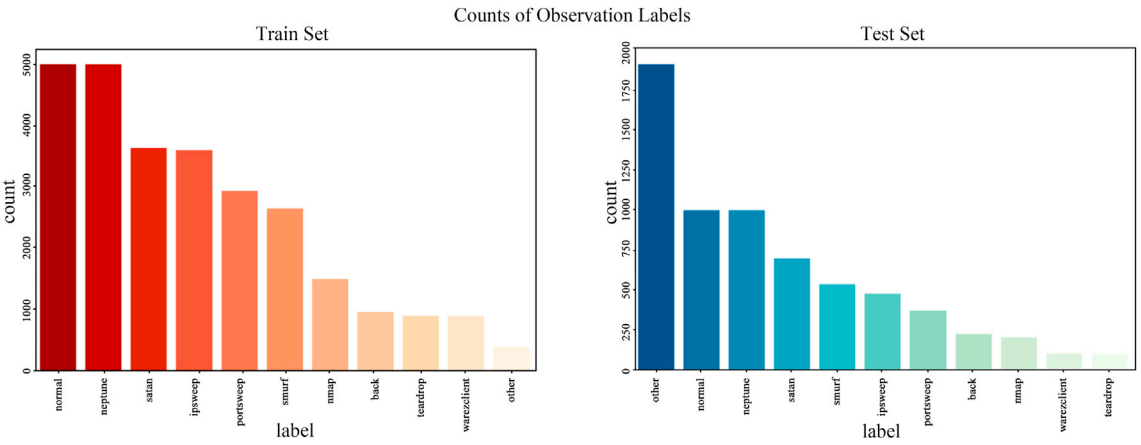


Fig. 7. Dataset label feature distribution.

2.50 GHz CPU and 8.00 GB RAM. The NSL-KDD dataset was used as the experimental dataset due to the complexity of collecting real network attack data by the sensing nodes. The NSL-KDD dataset was formerly provided by the Advanced Research and Design Bureau under the U.S. Department of Defense. It was organized by the design bureau and collected during the intrusion detection project in the Lincoln Laboratory. It records data such as network connections and system audit logs by simulating the attack mode in a real situation, forming a dataset specially used to judge the behavior of network data. At present, the NSL-KDD dataset has become the most authoritative dataset for network intrusion detection, which provides a solid foundation for the development of intrusion detection technology intelligence.

The objective of this experiment is to classify and predict the collected datasets of an intrusion detection system. To better evaluate the experimental performance, this study mainly uses indicators such as accuracy rate, false positive rate, false negative rate, and so on. The characteristics of the NSL-KDD dataset can be roughly divided into three types: category, digital count, and digital rate. They are analyzed as follows:

(1) Category. Some features identify the named value of some content in the feature, such as the protocol. The type feature column indicates that the protocol is being observed, or the flag identifies the flag that occurs during this record. The values are TCP, UDP, and other protocol names. These feature columns must be uniquely hot coded during data preprocessing.
(2) Digital count type. Duration, src bytes, dst, the data in the characteristic columns such as bytes represent the count of their respective tracking contents, and the data type is an integer. For the data of these characteristic columns, the value range of the data cannot be determined at present. If the value distribution range is large, it needs to be standardized to eliminate the impact of subsequent analysis.
(3) Digital rate type. The values of the feature columns named rate are floating-point values from 0 to 1.0, indicating the rates of the things they represent. These features do not require temporary processing.

In order to observe the data distribution of label columns more intuitively, the histogram is used for data visualization, and the results are shown in Fig. 5.

It can be seen from the results in Fig. 5 that there is an obvious quantitative contrast in the number of tag column features in the data, and the number of normal behavior and Neptune attacks is significantly higher than that of other tags. This phenomenon is mainly caused by the attacker continuously sending a large number of forged syn packets by taking advantage of the defects in the TCP protocol's three handshakes.

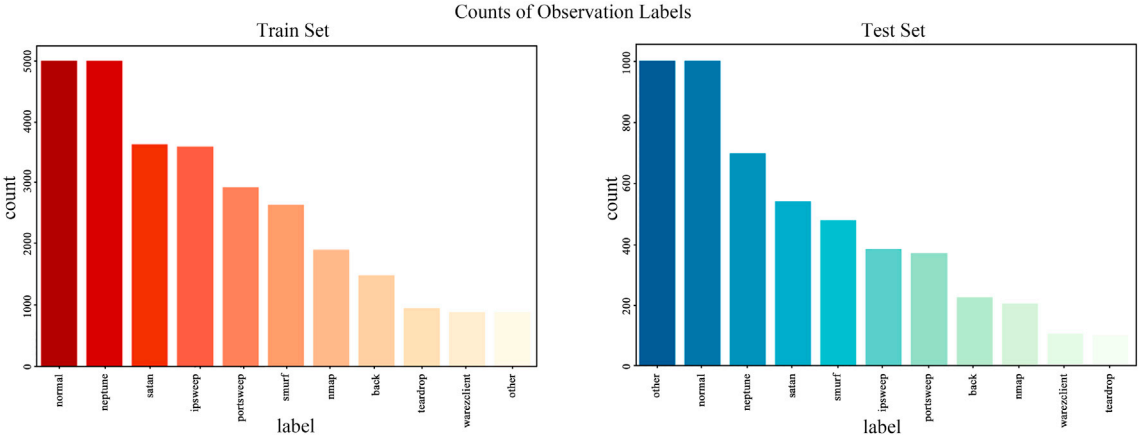


Fig. 8. Dataset label feature distribution.

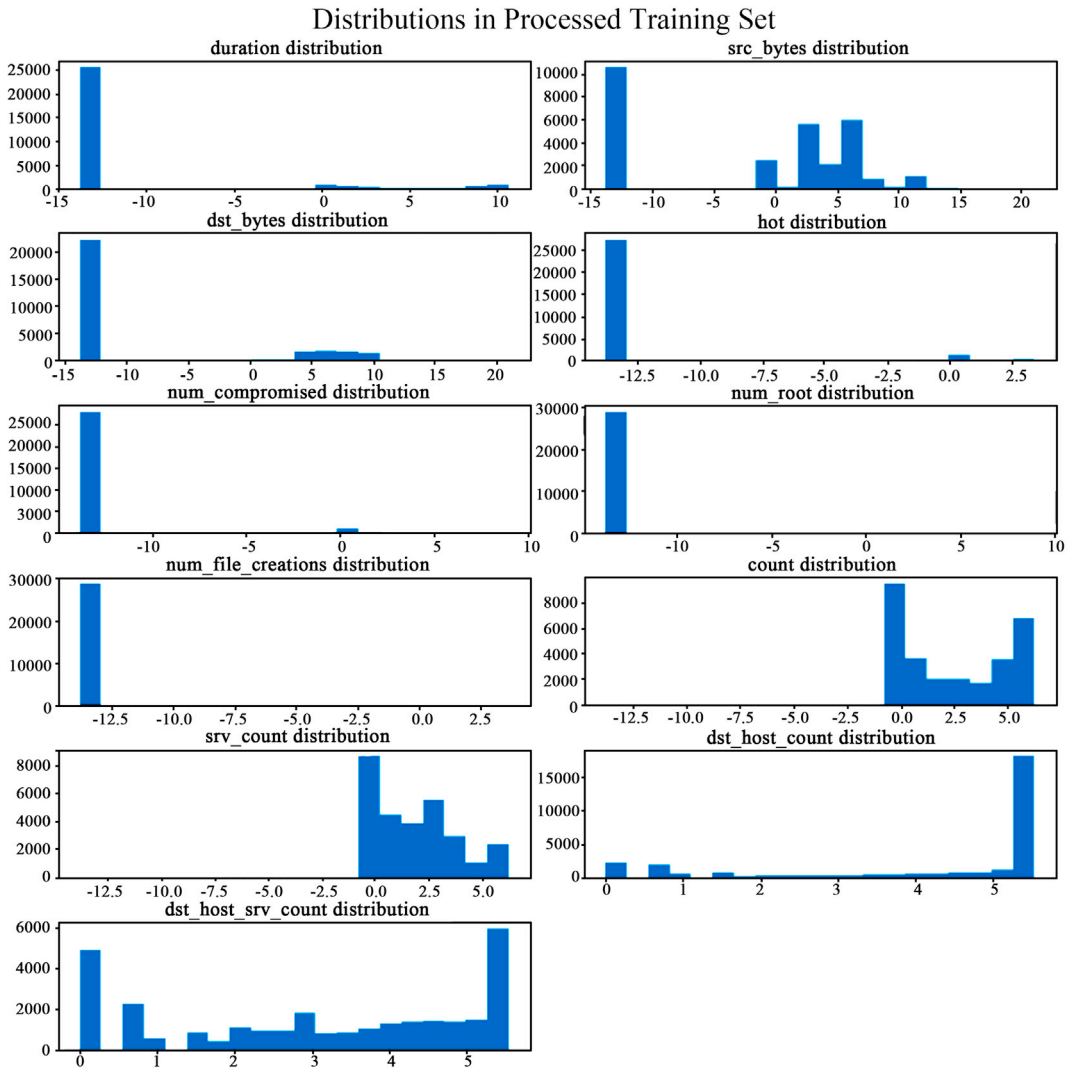


Fig. 9. Training set feature standardization results.

5.2. Feature engineering of dataset

5.2.1. Label column processing

In view of the large deviation of this data volume, the data volume of normal and Neptune in the training set and test set was reduced to 5000 and 1000, respectively, and then histogram analysis was conducted on the label column again. The results are presented in Fig. 6.

After data reduction, it can be seen that the data scale between different tags is closer than that before processing, but further observations found that some target tag data were rarely observed, resulting in an insufficient amount of tag data, and some tag data only appeared in the test set, which would have a great impact on the follow-up training results. At this time, the attack tags (normal, Neptune, Satan, ipsweep, portsweep, Smurf, nmap, back, teardrop, and warezclient) with sufficient data were retained, the tag data with small remaining data was merged, and the merged class was uniformly named another class. The number of label categories was visualized again, and the results are shown in Fig. 7.

The above results indicate that the discretization degree of the processed label category data was more standardized, which greatly reduces the huge data volume difference between various categories and facilitated the subsequent data use. It is further observed that the number of other label categories just merged shows an obvious difference between the training set and the test set, i.e., the number of other tag categories in the training set is the lowest, but the number of other tag categories in the test set is the highest, which is significantly better than the amount of data of the same type in the training set. Such contrast greatly interferes with subsequent training results, affecting the accuracy of the model. In view of the large difference in the amount of data, some data items in the other tag category in the training set (80% selected in this experiment) were transferred to the test set to balance the data scale of the same tag

Table 1
Training set processing results.

	duration	src_bytes	dst_bytes	land	…	flag_S2	flag_S3	flag_SF	flag_SH
13	−13.815511	5.811141	−13.815511	0	…	0	0	1	0
17	−13.815511	2.890372	−13.815511	0	…	0	0	1	0
30	−13.815511	2.079442	−13.815511	0	…	0	0	1	0
33	−13.815511	−13.815511	−13.815511	0	…	0	0	0	0
46	−13.815511	3.332205	−13.815511	0	…	0	0	1	0

Table 2
Comparison of accuracy and training time of different algorithms.

ALGORITHM	ACCURACY (%)	TRAINING TIME (s)
MFE-ELM	96.53 ± 0.34	4.64 ± 0.20
ELM	94.05 ± 0.18	4.25 ± 0.07
SVM	91.41 ± 0.13	19.04 ± 0.15
BP	86.72 ± 0.27	82.93 ± 0.36

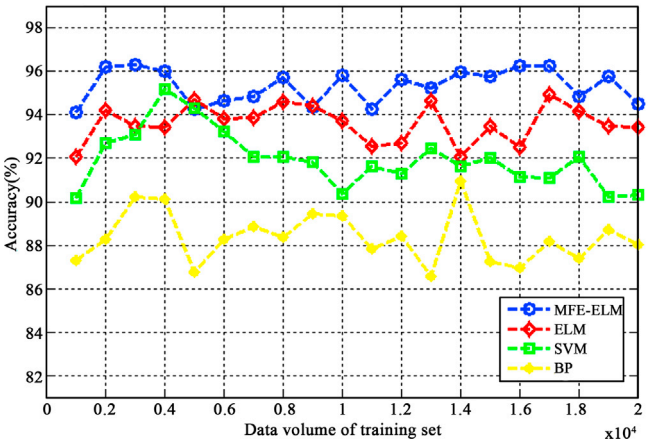


Fig. 10. Comparison of accuracy of MFE-ELM, ELM, SVM and BP.

category in the training and test sets. A comparison of the number of label categories after processing is shown in Fig. 8.

After the above multi-step processing, we reduced the 23 tag values in the original training set and 36 tag values in the test set to 11 tag values, and reduced the number of normal and Neptune tag values in the training set and test set to 5000, and neutralized the number of other tag values in the training set and test set to approximately 2000 and 400, respectively, and the specific gravity was in the middle of the same dataset. Through the label category data scale comparison chart at this time, it can be seen that the label feature columns of the training set and the test set were processed.

5.2.2. Number type column processing

Integer-type features with more than 10 eigenvalues are standardized below. Taking the training set data as an example, the processed results are shown in Fig. 9.

In addition, because this dataset is the benchmark dataset, to avoid destroying the integrity of information or causing unnecessary information loss, the integer-type features representing the rate will not be standardized for the time being.

To facilitate subsequent data training, the feature labels of category classes were transformed into the feature labels of integer values, and the discontinuous features were processed by unique heat coding. Here, we first reconciled the training and test sets, encoded the overall classification features, and then split the connected dataset back into separate training and test sets. After processing, the processed training and test sets were viewed. Consider the training set as an example, and the results are listed in Table 1.

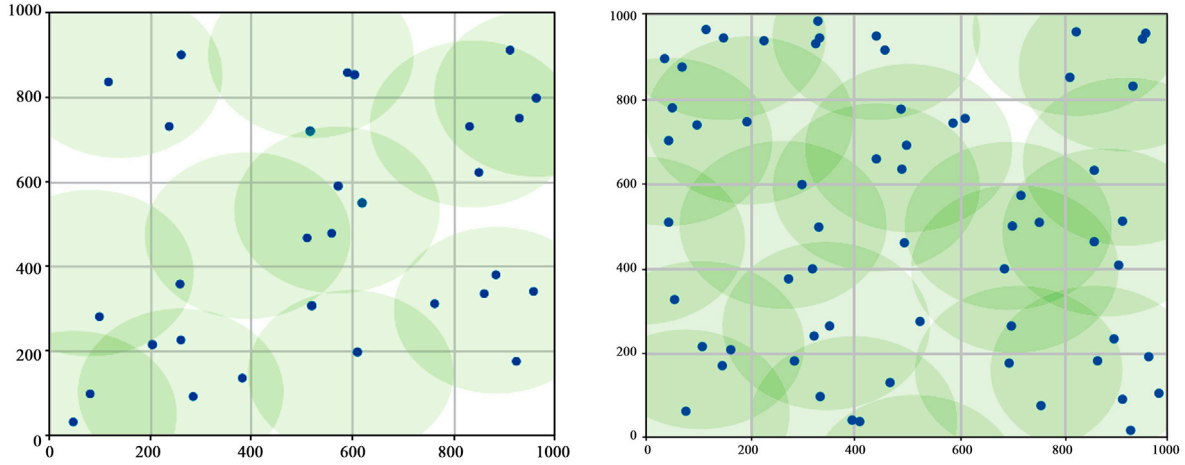
Thus far, data preprocessing and feature engineering are completed.

5.3. Experimental results and analysis

In the simulation experiment, we compared the performance of MFE-ELM, traditional ELM, SVM and BP neural network as intrusion detection algorithms in a cloud computing environment, in which ELM, SVM and BP are deployed on cloud nodes. The kernel function of SVM is sigmoid. The learning rate of BP is $lr = 0.07$, the momentum term coefficient $mc = 0.9$, the maximum number of iterations $epochs = 6000$, $goal = 0.0001$. 30000 pieces of data are taken as the total training samples and 10000 pieces of data are taken as the test samples. The parameters of comparison are training time and detection accuracy. The experimental data is the average of the data obtained by running the experimental program 20 times. According to the results in Table 2, the MFE-ELM proposed in this study performs well in terms of accuracy. In terms of training time, MFE-ELM needs to perform the process of sample screening, so it is slightly longer than that of the traditional ELM. In terms of training time and accuracy, BP performs worse than MFE-ELM and ELM, while SVM also requires a longer training time. Therefore, it can be concluded that in terms of accuracy, MFE-ELM, as an intrusion detection algorithm, has a better performance in a cloud computing environment.

In the experiment, we analyzed the training time and accuracy of the MFE-ELM algorithm under different training sets. Based on the above experiments, only MFE-ELM, ELM, SVM and BP were selected to compare the training time and accuracy. The number of training sets was extracted from 1000, 2000 , … , 20000, respectively. The experimental results are shown in Fig. 10.

As shown in Fig. 10, MFE-ELM still exhibits higher intrusion detection accuracy than ELM and SVM as the training set size increases. From the perspective of the algorithm, the main reason is that traditional classical



(a) 30 Terminal flow generators (b) 60 Terminal flow generators

Fig. 11. Terminal flow generator.

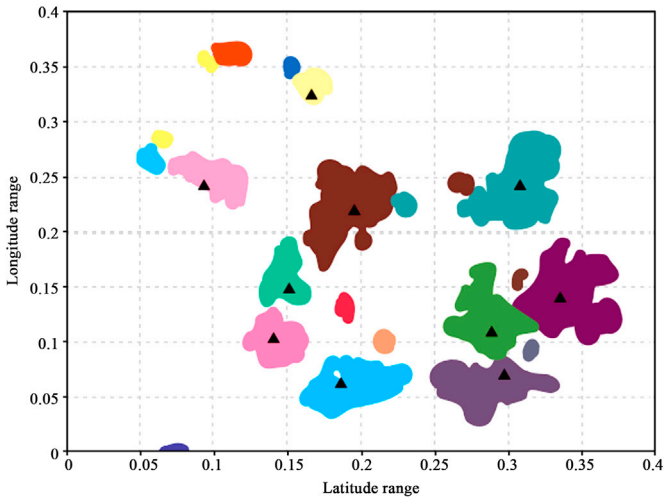


Fig. 12. Clustering results of terminal test points (range 1).

machine learning algorithms are based on gradient descent. Their main purpose in the training process is to achieve the minimum training error without considering the size of the output weight. In contrast to these learning algorithms, MFE-ELM achieves not only the minimum training error, but also the minimum weight norm. There is no overfitting phenomenon in the training process of MFE-ELM, and there is no problem with local minima in the solution process. Therefore, compared with the traditional learning algorithm, the learning repetition rate will not be too high to improve the accuracy of intrusion classification. In the training process of the MFE-ELM, the square loss function on each cloud node is optimized. The optimization is mainly reflected in the sample screening process. According to this algorithm, each cloud node obtains the best external weight that is more suitable for training. As shown in Table 2, for the training time, the simulation results show that the training time of the MFE-ELM proposed in this paper is slightly slower than that of ELM. The main reason is that MFE-ELM has a new sample screening process compared with ELM. Although MFE-ELM performs more slowly than ELM, the difference is small and within an acceptable range. It can be concluded that the MFE-ELM algorithm performs well in the intrusion detection of cloud computing. Fig. 11 shows the simulation experiment terminal traffic generator, in which the red dot represents the randomly distributed terminal traffic generator, the blue circle represents the coverage of all deployed edge gateways, and its center is the deployment location of edge gateways. There are 30 terminal traffic generators in Figs. 11(a) and 60 terminal traffic generators in Fig. 11(b). The way to cover each endpoint traffic generator is shown in Fig. 11.

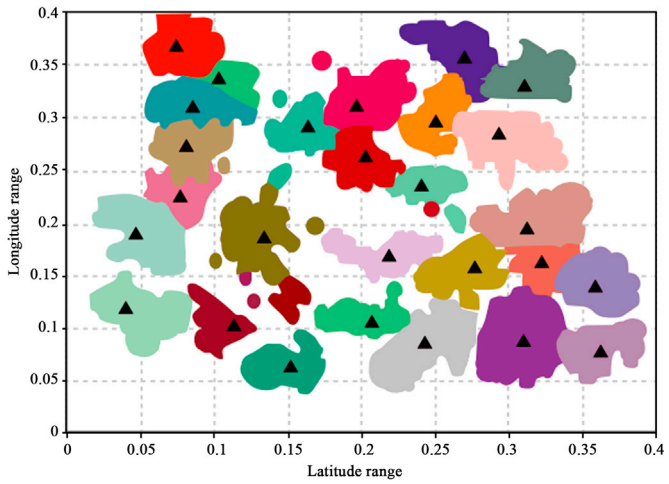


Fig. 13. Clustering results of terminal test points (range 2).

Visualize the clustering of all terminal test points. The first range is the clustering of all terminal test points with horizontal coordinates within [0, 0.4] and vertical coordinates within [0, 0.4], as shown in Fig. 12. The second range is the clustering of all terminal test points with horizontal coordinates in the range of [0, 0.4] and vertical coordinates in the range of [0, 0.4], as shown in Fig. 13, in which the black triangle is the final base station position.

In cloud computing networks, the network environment of cloud nodes is highly dynamic. Therefore, this study needs to analyze the robustness of the MFE-ELM algorithm, and specifically, it needs to use the NSL-KDD dataset to simulate the dynamic network environment. In other words, we analyze the dependence of the MFE-ELM algorithm on time statistics. In the NSL-KDD dataset, some fields are related to time-based traffic statistics, which are statistical parameters of time that form the relationship between the current connection record and the previous connection record during the training process. However, in an actual cloud computing network environment, there is a large amount of raw data without manual statistical processing. In other words, it is difficult to obtain data according to the time statistics. To ensure that the intrusion detection algorithm could be effectively implemented in a real-time and highly dynamic network environment, a robustness experiment was carried out by eliminating the characteristics of some fields.

How to effectively use unlabeled data, for discrete data, it is impossible to obtain statistical characteristics of traffic; the short length of data makes it difficult to calculate statistical characteristics of payload. Because the source domain data and target domain data in a real network often contain a variety of protocol data and noise, the distributions of the source and target domains are quite different. At this time, the direct use of all data cannot improve the algorithm performance, or even reduce the classification effect, especially the existence of a large number of samples with different distributions from the data to be tested. For encrypted data, because of its strong randomness, the encrypted data of different protocols have little impact on the learning and training of the classifier. For non-encrypted data, the distribution of the different protocols varies significantly. Data with different distributions will affect the selection of the classifier classification hyperplane and result in a reduction in the classification performance. In this study, the non-encrypted data of some datasets were analyzed, as shown in Fig. 14.

As can be seen in Fig. 14 above that there are great differences in the statistical distribution of data values between ACARS and AIS, which shows that there are also great differences in the data distribution between ACARS and AIS. If ACARS unlabeled data are used as auxiliary data, the mined data distribution information cannot be used to improve the classification effect of the AIS. Also, Since the target dataset is composed of public protocols, this study also extracted the data payload for processing and filtered non-encrypted data with a similar distribution. As the filtered data were not pure non-encrypted data, they are called semi-labeled non-encrypted data, and the encrypted data generated by the encryption algorithm were used as unlabeled data to assist in classification and recognition.

6. Conclusion and future work

The IoT is a complex, dynamic environment, which poses new challenges to the deployment and algorithm performance of IoT intrusion detection. Based on the research of cloud-based IoT defense architecture, this study focused on cloud computing intrusion detection algorithms and measurement models. Considering the resource constraints of cloud nodes and the high complexity of the network environment, a measurement model of IoT intrusion detection in a cloud computing environment is proposed, and an MFE-ELM algorithm of a limit learning machine with multi-feature extraction is proposed. Experimental simulations verified the training time and detection accuracy of the proposed algorithm. In the experiment, the MFE-ELM algorithm showed good performance in cloud computing, especially in terms of accuracy and

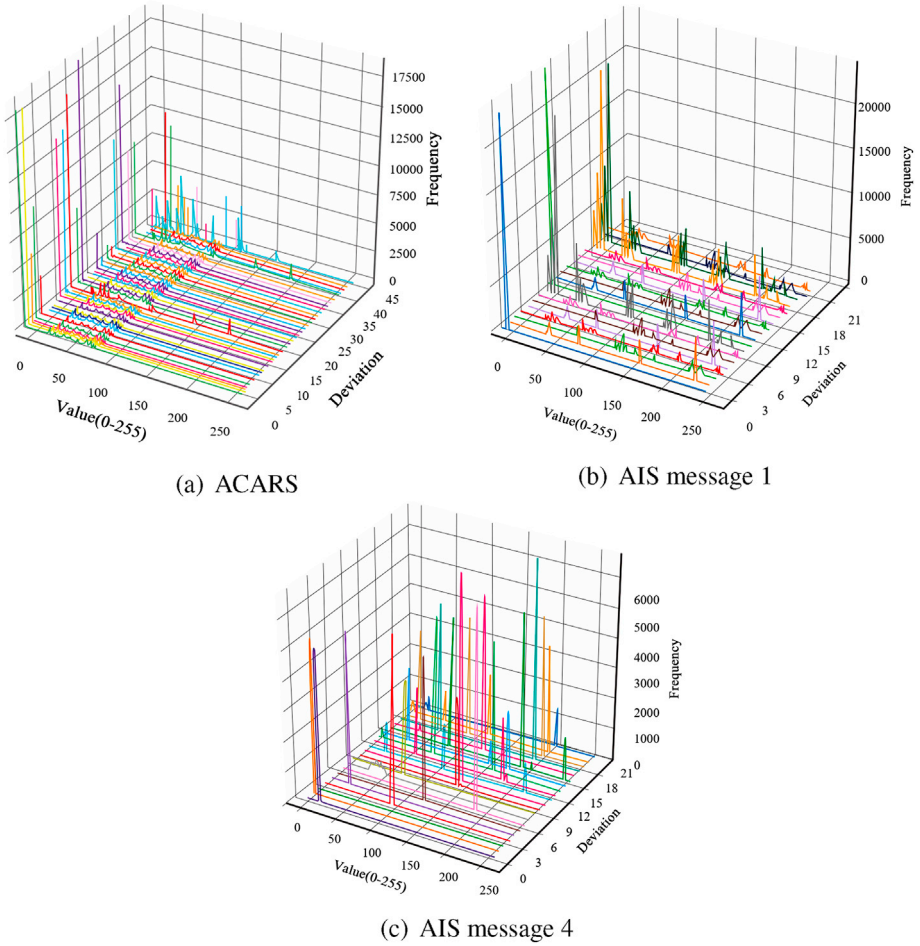


Fig. 14. Statistical distribution of non-encrypted data in ACARS and AIS protocols.

time efficiency. Through several experiments, it was proved that the MFE-ELM is an intrusion detection algorithm suitable for cloud computing environments.

The future work of this study is as follows:

(1) Cloud node host exception detection. There is no complete and reliable detection method for abnormal data that may be generated locally by the cloud node, such as operation records, registry data, and cloud node hardware parameters. Because of the heterogeneous characteristics of cloud nodes, constructing a general IoT intrusion detection method is one of the problems to be solved.
(2) When studying problems in this field, we should focus on a specific application scenario, such as intelligent gateway host anomaly detection in the IoT. Due to the resource-constrained characteristics of cloud nodes, the lightweight host anomaly detection method is worthy of research.
(3) Analysis of intrusion behavior. The intrusion behavior of the IoT often has the characteristics of multi-objective, multi-channel, and multi-source. Intruders hide their intrusion intentions through various technical methods. It is difficult to detect these intrusions with only lightweight detection methods. Therefore, it is necessary to further analyze and process the collected data to analyze the intruder's attack intention or correctly predict the intruder's intrusion behavior.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was funded by the Key Research and Development plan of Jiangsu Province (Social Development) No.BE20217162 and Jiangsu Modern Agricultural Machinery Equipment and Technology Demonstration and Promotion Project No. NJ2021-19.

References

[1] Z. Chiba, N. Abghour, K. Moussaid, A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection, Comput. Secur. 75 (6) (2018) 36–58.
[2] I.S. Thaseen, C.A. Kumar, A. Ahmad, Integrated intrusion detection model using chi-square feature selection and ensemble of classifiers, Arabian J. Sci. Eng. 44 (4) (2019) 3357–3368.
[3] T. Qian, Y. Wang, M. Zhang, J. Liu, Intrusion detection method based on deep neural network, J. Huazhong Univ. Sci. Technol. 46 (1) (2018) 6–10.
[4] S.M.H. Bamakan, H. Wang, Y. Shi, Ramp loss k-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem, Knowl. Base Syst. 126 (2017) 113–126.
[5] W.L. Al-Yaseen, Z.A. Othman, M.Z.A. Nazri, Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system, Expert Syst. Appl. 67 (2017) 296–303.
[6] B. Baranidharan, Internet of Things (IoT) technologies, architecture, protocols, security, and applications: a survey, in: Handbook of Research on Cloud and Fog Computing Infrastructures for Data Science, IGI Global, 2018, pp. 149–174.

[7] M.M.A. Al, M.R. Yuce, Sensors and systems for wearable environmental monitoring toward IoT-enabled applications: a review, IEEE Sensor. J. 19 (18) (2019) 7771–7788.
[8] A.A. Zaidan, B.B. Zaidan, A review on intelligent process for smart home applications based on IoT: coherent taxonomy, motivation, open challenges, and recommendations, Artif. Intell. Rev. 53 (1) (2020) 141–165.
[9] S. Balaji, K. Nathani, R. Santhakumar, IoT technology, applications and challenges: a contemporary survey, Wireless Pers. Commun. 108 (1) (2019) 363–388.
[10] Y. Chuan-Long, Z. Yue-Fei, F. Jin-Long, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access (5) (2017) 21954–21961.
[11] M.H. Mohammad, G. Abdu, A. Ahmed, A hybrid deep learning model for efficient intrusion detection in big data environment, Inf. Sci. 513 (2020) 386–396.
[12] C.M. Hsu, M.Z. Azhari, H.Y. Hsieh, Robust network intrusion detection scheme using long-short term memory based convolutional neural networks, Mobile Network. Appl. (2020) 1–8.
[13] X. Wang, S. Yin, H. Li, A network intrusion detection method based on deep multi-scale convolutional neural network, Int. J. Wireless Inf. Network 27 (4) (2020) 503–517.
[14] R. Wazirali, An improved intrusion detection system based on knn hyperparameter tuning and cross-validation, Arabian J. Sci. Eng. 45 (12) (2020) 10859–10873.
[15] J. Kim, J. Kim, H. Kim, Cnn-based network intrusion detection against denial-of-service attacks, Electronics 9 (6) (2019) 916–929.
[16] Y. Xiao, X. Xiao, An intrusion detection system based on a simplified residual network, Information 10 (11) (2019) 356.
[17] X. Congyuan, S. Jizhong, D. Xin, An intrusion detection system using a deep neural network with gated recurrent units, IEEE Access 6 (2018) 48697–48707.
[18] L.Q. Liu, B. Xu, X.P. Zhang, An intrusion detection method for internet of things based on suppressed fuzzy clustering, EURASIP J. Wirel. Commun. Netw. 113 (1) (2018) 1–7.
[19] G. Thamilarasu, S. Chawla, Towards deep-learning-driven intrusion detection for the internet ofthings, Sensors 19 (9) (2019) 1977–1996.
[20] B. Roy, H. Cheung, A deep learning approach for intrusion detection in internet of things usingbi-directional long short-term memory recurrent neural network, in: 2018 28th International Telecommunication Networks and Applications Conference, IEEE, Sydney, 2018, pp. 1–6.
[21] R. Vijayanand, D. Devaraj, B. Kannapiran, A novel intrusion detection system for wireless mesh network with hybrid feature selection technique based on ga and mi, J. Intell. Fuzzy Syst. 34 (3) (2018) 1243–1250.
[22] H. Larijani, J. Ahmad, N. Mtetwa, A heuristic intrusion detection system for internet-of-things, in: Intelligent Computing-Proceedings of the Computing Conference, Springer, London, 2019, pp. 86–98.
[23] M. Al-Hawawreh, N. Moustafa, E. Sitnikova, Identification of malicious activities in industrial internet of things based on deep learning models, J. Inf. Secur. Appl. 41 (8) (2018) 1–11.
[24] C. Constantinides, S. Shiaeles, B. Ghita, A novel online incremental learning intrusion prevention system, in: 2019 10th IFIP International Conference on New Technologies, Mobility and Security, (NTMS), IEEE, 2019.
[25] B. Roy, H. Cheung, A deep learning approach for intrusion detection in internet of things using bi-directional long short-term memory recurrent neural network, in: 28th International Telecommunication Networks and Applications Conference (ITNAC), IEEE, 2019, pp. 57–62.
[26] A. Al-Abassi, H. Karimipour, An ensemble deep learning-based cyber-attack detection in industrial control system, IEEE Access 99 (8) (2020) 83965–83973.
[27] P.V. Huong, D.T. Le, T. Le, Intrusion detection in IoT systems based on deep learning using convolutional neural network, in: 2019 6th NAFOSTED Conference on Information and Computer Science, NICS), 2020, pp. 448–453.
[28] G. Altan, Secure DeepNet-IoT: a deep learning application for invasion detection in industrial Internet of Things sensing systems, Trans. Emerge. Telecommun. Technol. 32 (4) (2021) e4228–e4243.
[29] V. Priya, I.S. Thaseen, T.R. Gadekallu, Robust attack detection approach for IoT using ensemble classifier, Comput. Mater. Continua (CMC) 66 (3) (2021) 2457–2470.
[30] R. Kumar, R. Tripathi, DBTP2SF: a deep blockchain-based trustworthy privacy reserving secured framework in industrial internet of things systems, Trans. Emerge. Telecommun. Technol. 32 (4) (2021).
[31] I. Psychoula, E. Merdivan, D. Singh, L. Chen, M. Geist, A deep learning approach for privacy preservation in assisted living, in: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops), 2018, pp. 710–715.
[32] I. Psychoula, et al., Users' privacy concerns in IoT based applications, in: 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing,Internet of People and Smart City, Innovation, IEEE, 2018.
[33] M. Nivaashini, P. Thangaraj, A framework of novel feature set extraction based intrusion detection system for internet of things using hybrid machine learning algorithms, in: 2018 International Conference on Computing, Power and Communication Technologies, IEEE, Greater Noida, 2018, pp. 44–49.
[34] L. Khalvati, M. Keshtgary, N. Rikhtegar, Intrusion detection based on a novel hybrid learning approach, J. AI Data Mining. 6 (1) (2018) 157–162.
[35] T.N. Pham, E. Foo, S. Suriadi, Improving performance of intrusion detection system using ensemble methods and feature selection, in: Australasian Computer Science Week Multi-Conference, ACM, New York, 2018, pp. 1–6.
[36] S. Mohammadi, V. Desai, H. Karimipour, Multi-variate mutual information-based feature selection for cyber intrusion detection, in: 2018 IEEE Electrical Power and Energy Conference, IEEE, Toronto, 2018, pp. 1–6.
[37] Z.B. Sun, Q.B. Song, X.Y. Zhu, A novel ensemble method for classifying imbalanced data, Pattern Recogn. 48 (5) (2015) 1623–1637.
[38] A. Holzinger, et al., Digital transformation for sustainable development goals (sdgs) - a security, safety and privacy perspective on ai, Springer Lecture Note Comput. Sci. 48 (5) (2021) 1–20.
[39] P. Kieseberg, et al., A tamper-proof audit and control system for the doctor in the loop, Brain Informatics 3 (4) (2016) 269–279.