Multimedia Tools and Applications (2025) 84:16425–16453
https://doi.org/10.1007/s11042-024-19695-7

Using the ToN-IoT dataset to develop a new intrusion detection system for industrial IoT devices

Zhong Cao[1,2] · Zhicai Zhao[1,2] · Wenli Shang[1,2] · Shan Ai[3] · Shen Shen[4]

Received: 10 October 2023 / Revised: 21 February 2024 / Accepted: 7 June 2024 /
Published online: 28 June 2024

Abstract
With the rapid expansion of the Internet of Things (IoT), an increasing number of IoT devices are becoming interconnected over the Internet, consequently elevating the vulnerability to attacks targeting these devices. Consequently, conducting thorough and high-quality research in intrusion detection within the industrial IoT domain has become imperative. Machine learning is well suited to be used for intrusion detection in IoT scenarios in the face of increasing data volumes. We leveraged the ToN-IoT dataset, utilizing both the IoT device dataset and the Network dataset contained within, to propose a novel approach that integrates multiple datasets for replicating complex IoT scenarios. Pioneeringly, we introduced the use of sine and cosine component cyclic encoding for temporal attributes. Moreover, we address the challenge posed by data imbalance by employing a combination of the Synthetic Minority Over-sampling Technique (SMOTE) for oversampling and an improved version of the redundant-based Tomek link removal technique for under-sampling. We employ various machine learning and deep learning algorithms to construct binary classifiers, categorize data and distinguish between attack data and normal data. The attack data undergoes multiclass classification to identify distinct types of attack, thereby augmenting the attack database. As for the normal data can either be utilized as is or stored encrypted for future use. To comprehensively evaluate the performance of our intrusion detection systems, we used a set of evaluation metrics, including accuracy, precision, recall, and F1 score. We achieved a 100% accuracy in binary classification, while in multi-class classification, we attained accuracy rates of 98.87%, 99.36%, 99.18%, and 99.18% for precision, recall, and F1 score respectively. The evaluation results indicate that our research exhibits superior performance, consistently performing well on several key metrics.

Keywords Intrusion detection system (IDS) · Internet of things(IoT) · ToN_IoT dataset · Machine learning · Deep learning

✉ Wenli Shang
shangwl@gzhu.edu.cn

1 School of Electronics and Communication Engineering, Guangzhou University, Guangzhou 510006, China

2 Key Laboratory of On-Chip Communication and Sensor Chip of Guangdong Higher Education Institutes, Guangzhou University, Guangzhou 510006, China

3 School of Artificial Intelligence, Guangzhou University, Guangzhou 510006, China

4 Guangzhou Yunqi intelligent technology Co., LTD., Guangzhou 510006, China

1 Introduction

The Internet of Things (IoT) represents a burgeoning paradigm that enables the interconnection of physical objects and computational capabilities with Internet connectivity. In recent years, the rapid global advancement of IoT technology has resulted in a corresponding surge in the number of Internet-connected devices, with IoT devices assuming a pivotal role in this landscape. The IoT market, propelled by technological innovations and the seamless production of intelligent devices, exhibits continual growth, evidenced by the daily influx of new IoT devices joining the Internet [1] .Consequently, the proliferation of IoT technology has fostered the swift development of IoT applications across diverse industries, encompassing domains such as connected vehicles [2], environmental monitoring [3], smart homes [4], remote healthcare [5], manufacturing, and retail [6]. While the IoT offers considerable enhancements in productivity and efficiency through intelligent and remote management, it simultaneously introduces heightened vulnerabilities to network attacks due to the dearth of robust security measures within the IoT ecosystem. This exposes IoT devices to malevolent attacks originating from both internal and external sources, thereby accentuating the imperative for comprehensive security protocols [7]. As an extension and amplification of the traditional Internet infrastructure, the IoT exposes a substantial number of devices to the Internet within an intricately structured architecture [8]. Given the inherent complexity and the critical nature of security concerns across all facets of the IoT [9] , it is imperative not to underestimate the significance of addressing these issues. From an attacker's perspective, IoT sensors epitomize ideal botnet nodes due to their pervasive deployment, networking the requirements, suboptimal default settings, abundance of software vulnerabilities, and the ease with which their presence can be overlooked [10]. Consequently, network security matters in the IoT assume a central role in research endeavors, as the identification and mitigation of potential threats and the reduction of risks pertaining to IoT applications emerge as prominent topics within the field of network security [11].

Industrial IoT applications, integral to the industry 4.0 revolution, are confronted with heightened security risks, particularly within industrial control and infrastructure systems that demand stringent security measures [12]. A noteworthy incident occurred when multiple Ukrainian substations fell victim to a recent IoT attack, resulting in substantial power outages affecting approximately 225,000 customers [13]. Another prominent case involves the Mirai botnet, which emerged in late 2016. Comprising compromised smart cameras, this botnet instigated widespread disruption to Internet services by launching massive distributed denial-of-service (DDoS) attacks against well-established companies [14]. Given the gravity of these threats, it is imperative to establish effective security measures to safeguard IoT applications. While traditional IoT systems have relied on security tools such as firewalls, encryption, and intrusion detection systems [15], the unique characteristics of IoT applications necessitate tailored security solutions. Lightweight communication protocols and resource-constrained devices with limited computational and storage capacities typify IoT deployments. Consequently, security applications requiring extensive computational resources and cryptographic solutions are unsuitable for IoT environments. As a result, existing security measures prove insufficient in fully protecting IoT applications against potential threats. To effectively detect network threats targeting IoT services, the development of network security applications such as intrusion detection systems (IDS) assumes paramount importance. In recent years, network security researchers have focused significant attention on the advancement of IDS technology [16]. By leveraging IDS capabilities, it becomes possible to identify and mitigate

potential security breaches in IoT networks, thereby fortifying the overall security posture of these applications.

IDS is a vital component in network security that aims to detect potential intrusion activities within a network. By collecting and analyzing critical data such as historical information and network logs, IDS establishes specific security policies to govern the network [17]. It operates in real-time, continuously monitoring the network to identify and flag any unauthorized intrusions that violate the defined security policies, thereby safeguarding the confidentiality and integrity of the network. In essence, IDS serves as a secondary defense mechanism, complementing other security measures like firewalls. If firewalls can be likened to locks on the doors of a building, IDS functions as an internal surveillance system within that building. In the event of a trespasser breaching the building or an insider crossing the established security boundaries, the IDS promptly raises an alarm. This provides an additional layer of protection, capable of detecting malicious activities that may have bypassed other security mechanisms. With the increasing recognition of IoT threats, there has been a growing focus on advancing intrusion detection technology specifically tailored for IoT environments [12]. Various methodologies have been proposed, including novel signature-based, anomaly-based, and specification-based models. These advancements aim to enhance the effectiveness and efficiency of intrusion detection in IoT networks, aligning with the evolving security requirements of this interconnected ecosystem [18].

In recent years, machine learning have flourished, and an increasing number of machine learning and deep learning algorithms have been applied to the field of intrusion detection [19–21]. As shown in [22], most current research is based on relatively old datasets such as KDD-99 [23], UNSWNB15 [24], or CICIDS [25]. These datasets were once classic network intrusion detection datasets, but with the passage of time, the heterogeneity of current IoT networks, as well as many different protocols, standards, and technologies, has become significantly different from the past. In 2020, [26] proposed a new intrusion detection dataset called ToN_IoT, which includes various normal and attack events for different IoT services, as well as heterogeneous data sources, including telemetry data from IoT devices, Linux operating system (OS) logs, Windows OS logs, and network traffic from IoT systems. While the ToN_IoT dataset has attracted attention and spurred numerous intrusion detection algorithm studies in recent years, it is worth noting that much of the existing research in this domain is confined to smaller IoT datasets, such as the Network dataset, or datasets specific to Linux or Windows OS environments. Consequently, these limited datasets fail to adequately simulate the complexities and challenges encountered in real-life IoT system scenarios. To address this limitation, further exploration and investigation are necessary to fully utilize the ToN_IoT dataset and to develop intrusion detection algorithms that can effectively handle the intricacies of complex IoT environments. Moreover, the current research does not make enough use of temporal features in the dataset, which can cause indistinguishability between normal data as well as attack data in some scenarios. In this paper we will use the ToN_IoT dataset to process the data to simulate a complex IoT system scene in a real scenario, and use the sine and cosine components to encode the temporal features in a circular manner so that the temporal features can be well utilized. To mitigate the issue of data imbalance, we utilize a combination of the Synthetic Minority Over-sampling Technique (SMOTE) oversampling technique and an improved redundant-based Tomek link removal under-sampling technique. Subsequently, we employ various machine learning and deep learning algorithms to construct binary classifiers to categorize the data, differentiating between attack data and normal data. Following that, we will undertake multiclass classification on the attack data to discern the types of attacks, thereby enriching the attack database. As for the normal data, it can either be used as is or stored in encrypted form for future utilization.

The key contributions of this study are listed as follows:

1. Combining the datasets of seven IoT devices from the ToN_IoT dataset together, we utilized the median method to fill in missing values and form a large-scale dataset, which includes data flow information from seven types of IoT devices. This is to simulate complex IoT system scenarios in real life. Additionally, we combined the IoT dataset with the network dataset, which includes data flow information from IoT devices as well as data flow information after interconnection. Our dataset is closer to the IoT system scenario under Internet conditions.

2. For the problem of low utilization of temporal features, cyclic encoding of periodic features is used to convert temporal features into sine and cosine components for subsequent use.

3. To mitigate the issue of data imbalance, we utilize a combination of the SMOTE oversampling technique and an improved redundant-based Tomek link removal under-sampling technique.

4. The generated processed dataset is evaluated using several different machine learning algorithms and the best classification method with high evaluation metrics for distinguishing between normal and attack instances is selected. This contributes to the subsequent utilization of normal data and the enrichment of the attack database. The intrusion detection system serves as a valuable reference for the construction of intrusion detection systems in other datasets or scenarios.

This paper is organized as follows. Section 2 discusses the current research and related works. Section 3 introduces the proposed approach, system framework, and the key working principles of the technologies utilized as well as the algorithms employed. Section 4 describes the experimental method setup and the experimental results and analysis . Finally, Section 5 gives a conclusion and possible future directions.

2 Related work

2.1 Intrusion detection based on machine learning

In recent years, with the advancement of machine learning, an increasing number of researchers have applied these techniques to intrusion detection, gradually becoming mainstream. Debar et al. [27] first introduced neural networks into intrusion detection in 1992, and since then, the application of machine learning and deep learning in intrusion detection has gained popularity.

Wang et al. [28] proposed an effective network intrusion detection method based on Sparse Auto-encoder (SAE) and Random Forest (RF). The ANASYN oversampling technique was employed to address the data imbalance issue. Testing on the NSL-KDD dataset demonstrated that the proposed SAE-RF model outperforms traditional methods in terms of detection performance. However, one main drawback of this technique is the lack of consideration for algorithmic complexity. Both the ANASYN oversampling technique and the nested model of Sparse Autoencoder and Random Forest require significant computation time, resulting in higher detection costs. In another study, Ravi et al. [29] proposed an improved machine learning algorithm called SSML DFNNRRS-K-means(SDRK) for intrusion detection. SDRK combines supervised Deep Neural Networks (DNN) with unsupervised clustering techniques, resulting in improved accuracy. Compared to other models, the SDRK model has shorter testing time and higher precision. This suggests that SDRK is a better model for delay-critical

IoT use cases. However, this method assumes that fog nodes are not under attack, whereas in reality, attackers may infiltrate and control fog nodes, limiting the applicability of the proposed algorithm.Alzaqebah et al. [30] proposed an improved bio-inspired metaheuristic algorithm, which utilizes a paired submodel technique to address multi-class classification problems. Each submodel employs an enhanced Harris Hawk optimization approach, using Extreme Learning Machine (ELM) as the underlying base classifier. This hierarchical structure generates optimal feature subsets for each attack instance and optimized ELM weights, leading to a significant enhancement in detection rates.

Mirsky et al. [31] introduced an online algorithm named KitNET, which can learn to detect attacks on local networks without supervision. It effectively distinguishes between normal and anomalous traffic data in real-time using autoencoders. The efficiency of this algorithm is sufficient to run on a single core of a Raspberry Pi, making it suitable for simple IoT devices. Hazman et al. [32] employed ensemble learning methods and introduced an optimal anomaly detection model based on AdaBoost, along with a Boruta feature selection technique using the Xgboost algorithm. Moreover, the model was evaluated using the NSLKDD and BoT-IoT datasets. This approach achieved commendable performance metrics. Mohy-eddine et al. [33] proposed a network intrusion detection model for IoT environments based on the K-NN classifier and feature selection. Principal Component Analysis (PCA), univariate statistical tests, and Genetic Algorithm (GA) were employed for feature selection. The model was evaluated on the Bot-IoT dataset. The results indicated that, after feature selection, the model achieved improved performance as well as faster detection times. Caville et al. [34] proposed a Graph Neural Network (GNN) approach for intrusion and anomaly detection, utilizing edge features and graph topology in a self-supervised manner. This method allows for the detection of attack patterns by combining edge features and topological patterns without the need for labeled data. However, real traffic data from the internet and testing time, which are crucial in IDS, were not taken into consideration. Yaseen et al. [35] introduced an effective feature envelop selection method. This approach employs the differential evaluation algorithm to select useful features and utilizes the Extreme Learning Machine classifier to evaluate the selected features after feature selection. An evaluation was conducted using the NSL-KDD dataset. The results demonstrated that compared to recent related studies, this method can effectively reduce the training features, thereby enhancing detection accuracy and shortening the processing time of the detection procedure.

Long Short-Term Memory (LSTM) is a recurrent neural network architecture specifically designed to address the challenge of capturing long-term dependencies in sequential data, making it well-suited for intrusion detection applications. Laghrissi et al. [36] proposed an intrusion detection model based on LSTM. In the preprocessing stage, PCA was used for feature extraction. The top three features with the highest explained variance ratio in PCA were selected as inputs. Experimental results demonstrated that this algorithm exhibited good detection performance. However, the dataset used was KDD99, which contains outdated features and attack types, lacking persuasiveness in today's constantly evolving attack methods. Mushtaq et al. [37] introduced a hybrid framework consisting of a deep auto-encoder (AE) with LSTM and Bidirectional Long Short-Term Memory (Bi-LSTM) for IDS. Optimal features were obtained through the AE and LSTM, which were then classified as normal or anomalous samples. The performance of the proposed model was evaluated on the renowned NSL-KDD dataset using error metrics. However, due to its intricate nature, it took longer training time compared to other IDS methods.

The Generative Adversarial Networks(GAN) framework has also been applied to the development of intrusion detection systems in recent years. By generating adversarial attacks to challenge the detector, the performance of the detector can be improved. Rashid et al. [38]

explored the impact of adversarial attacks on deep learning and machine learning models using the latest IoT datasets. They proposed an adversarial retraining method to improve the detection performance of IDS in the face of adversarial attacks. The DS2Os dataset was used, achieving a detection accuracy of 99% for attacks including adversarial attacks. However, the model is sensitive to adversarial attacks and requires the design of features to reduce its sensitivity. Debicha et al. [39] proposed an adversarial detector based on transfer learning. They established multiple adversarial detectors based on transfer learning, with each detector receiving a subset of information passed through an IDS. By combining their individual decisions, it was verified that combining multiple detectors can further improve the detectability of adversarial traffic in the case of parallel IDS design, outperforming a single detector in terms of performance.

In recent years, ToN_IoT datasets have been proposed, which include heterogeneous data sources and can better simulate real industrial IoT scenarios. Sarhan et al. [40] extracted NetFlow features from network traffic in multiple datasets, including UNSW_NB15, BoT_IoT, ToN_IoT. Compared to the complex features used in the original datasets, extracting NetFlow features from network traffic is relatively easy. The generated NetFlow datasets were labeled for binary and multi-class classification and evaluated using an additional tree ensemble classifier consisting of 50 random decision tree estimators on multiple datasets for binary and multi-class classification. Lo et al. [41] introduced an intrusion detection model called E-GraphSAGE. This algorithm captures the edge features of the graph as well as the topological information for IoT network intrusion detection. The model was applied to datasets such as Bot_IoT, NF_BoT-IoT, ToN_IoT, and NF_ToN_IoT, documenting its performance in binary and multi-class classification. One main drawback of this technique is the lack of consideration for algorithmic complexity. As the dataset grows and the number of data entries increases, the resulting graph becomes large. Therefore, only 10% of the data was used for training and evaluation on the ToN_IoT dataset.

2.2 The dilemma of intrusion detection

To address the issue of insufficient utilization of temporal features, Van et al. [42] proposed a method for encoding cyclical features. Many features in datasets are inherently cyclical, such as time: months, days, weekdays, hours, minutes, and seconds, etc. Other examples include seasonal, tidal, and astrological data. The challenge is to let deep learning algorithms know that such features occur cyclically. After encoding, temporal features can be effectively applied in machine learning and deep learning algorithms.

In the field of intrusion detection, class imbalance is a common issue due to the significantly smaller number of attack data compared to normal data. To address the class imbalance problem, Chawla et al. [43] proposed a method called Synthetic Minority Over-sampling Technique(SMOTE). Testing on multiple datasets showed that the SMOTE method can improve the classifier's accuracy in classifying minority classes. Subsequently, researchers made improvements to the SMOTE method and introduced SVM-SMOTE [44] and Borderline-SMOTE [45]. SVM-SMOTE focuses on modifying SVM to effectively handle class imbalance problems, while Borderline-SMOTE performs SMOTE oversampling only on the minority samples near the decision boundary. This solution achieved better classification results compared to SMOTE and other random oversampling methods. However, it may not effectively separate the minority and majority classes when Tomek links exist.

Gupta et al. [46] proposed an intrusion detection system designed using cost-sensitive deep learning and ensemble algorithms. They employed a C-cost-Sensitive deep learning approach

to address the class imbalance problem. The classification performance was evaluated using the NSL-KDD, CIDDS-001, and CICIDS2017 datasets. Experimental results demonstrated that the system achieved high detection rates for both majority and minority attacks in the network and significantly reduced the number of false alarms. Ding et al. [47] introduced a tabular data sampling method to address the imbalanced learning problem. It employs the k-nearest neighbors method for effective undersampling, and the Targeted Adversarial Generative Adversarial Network model (TACGAN) for oversampling of attack samples. The TACGAN model is an extension of the ACGAN model. Two loss functions were added to the generator to measure the information loss between real data and generated data, making the data tend towards balance.

Table 1 summarizes the latest research achievements in intrusion detection. It is worth noting that many studies did not address the issue of class imbalance. Existing research has paid limited attention to addressing class imbalance. In the case of class imbalance, even if we predict all instances as normal, we may achieve highly desirable detection performance, but this contradicts the original purpose of intrusion detection. Furthermore, most current studies are based on small, single datasets, which fail to simulate complex real-world scenarios in the context of IoT systems. Moreover, the utilization of temporal features in the datasets is insufficient in current research, leading to difficulties in distinguishing between normal and attack instances in certain scenarios.

Inspired by the work of previous researchers and motivated by the challenges in the current research, we propose a dataset integration approach to simulate complex industrial IoT scenarios realistically. We also address the problems in the dataset by utilizing cyclic encoding of temporal features using sine and cosine components and a combination of SMOTE oversampling technique and an improved redundant Tomek link removal under-sampling technique is utilized to alleviate the issue of class imbalance. Furthermore, several different machine learning and deep learning algorithms are employed for evaluation.

Table 1 Related Classification Research on Intrusion Detection Systems

Author	Model	Dataset	Mode	balance
Wang et al. [28]	SAE,RF	NSL_KDD	M	ANASYN
Ravi et al. [29]	SDRK	NSL_KDD	B	No
Alzaqebah et al. [30]	ELM	UNSW-NB15	B and M	No
Hazman et al. [32]	Xgboost	NSLKDD , BoT-IoT	B and M	No
Mohy-eddine et al. [33]	GA	BoT-IoT	B and Mi	No
Yaseen et al. [35]	ELM	NSL-KDD	B and M	No
Laghrissi et al. [36]	LSTM	KDD99	B and M	No
Mushtaq et al. [37]	AE, Bi-LSTM	NSL-KDD	M	No
Rashid et al. [38]	GAN	DS2Os	B and M	No
Debicha et al. [39]	Transfer learning	NSL-KDD,CIC-IDS2017	Binary	No
Sarhan et al. [40]	Netflow	UNSW-NB15, BoT_IoT,ToN_IoT	B and M	No
Lo et al. [41]	E-GraphSAGE	Bot_IoT,ToN_IoT	B and M	No
Gupta et al. [46]	Ensemble	NSL_KDD,CIDDS_001, CICIDS2017	M	Cost-Sensitive

M means Multi-classification and B means Binary classification

3 Propose approach

3.1 Dataset and merging

The ToN_IoT dataset utilized in this study comprises diverse data obtained from a simulated IoT system, as depicted in Fig. 1. The dataset encompasses telemetry data originating from various IoT devices, including but not limited to "Fridge", "Garage_Door", "GPS_Tracker", "Modbus", "Motion_Light", "Thermostat", and "Weather". Additionally, it incorporates Linux and Windows operating system logs, as well as network traffic data derived from the IoT system. The ToN_IoT dataset exhibits a discernibly elevated degree of heterogeneity, establishing a more proximate correlation with authentic, industry-grade IoT networks in real-world scenarios [48]. To create this dataset, a real and extensive testbed network was established by the Networked Sensing and Control Group at the University of New South Wales in Canberra. This network incorporates a multitude of virtual machines, physical systems, hacking platforms, cloud and fog platforms, and IoT sensors. The purpose of this comprehensive network design is to simulate the complexity and scalability characteristic of IoT and Industry 4.0 networks, thereby providing a realistic environment for data collection and analysis.

In addition, the ToN_IoT dataset is represented in CSV format, with a label column for binary classification, where 0 represents normal behavior and 1 represents attack behavior. It

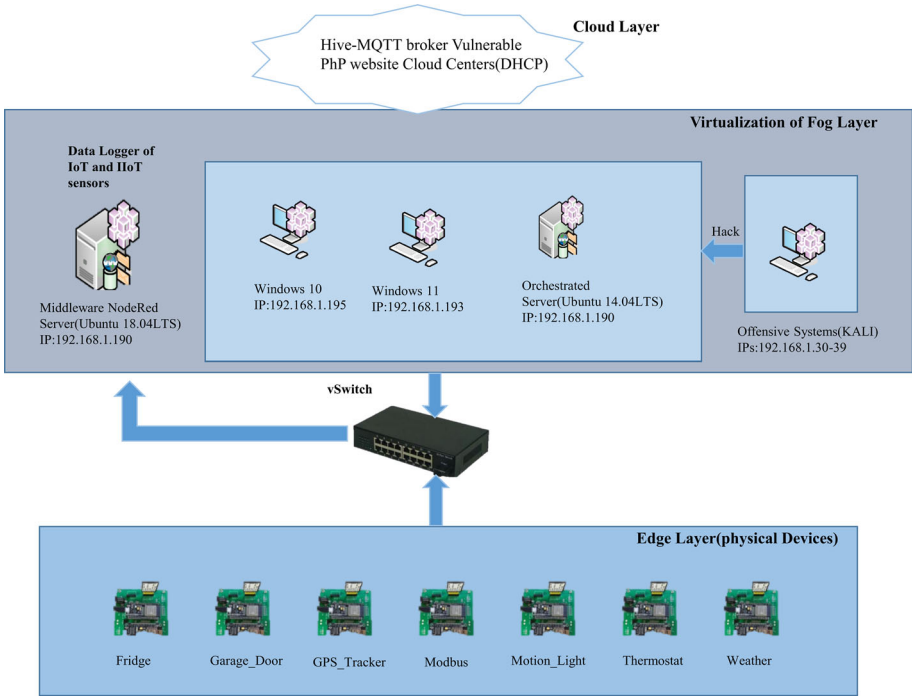

Fig. 1 Testbed environment of ToN_IoT datasets for collecting network traffic

also includes a label column for multi-classification, representing different types of attacks, such as ransomware, password attacks, scanning, denial of service (DoS), distributed denial of service (DDoS), data injection, backdoor, cross-site scripting (XSS), and man-in-the-middle (MITM). These attacks were launched and collected through IoT networks against various IoT sensors.

In the IoT dataset and Network dataset, the "label" is used for binary classification, where "0" represents normal data with a sample size of 245,000 records in IoT dataset and 300000 records in Network dataset, and "1" represents attack data with a sample size of 156,119 records in IoT dataset and 161043 records in Network dataset.

In the multi-classification,the dataset records is shown in Table 2. It is evident that in binary classification, there is a small difference in the quantities between normal data and attack data. However, in multi-class classification, there is a significant disparity between the quantities of minority classes and normal data, which is a common issue of data imbalance in the field of intrusion detection. Figures 2 and 3 utilize Pie charts to compare the quantities of majority and minority classes in multi-class classification.

"Fridge", "Garage_Door", "GPS_Tracker", "Modbus", "Motion_Light", "Thermostat" and "Weather" in the IoT dataset, the seven datasets were combined and fill in missing values using the median. This created a larger IoT dataset, simulating a more complex industrial IoT scenario. It stands out from federated learning due to its lower device requirements and complexity, possessing the characteristics of simplicity and efficiency [49]. The obtained "Network" dataset was concatenated with the previously mentioned IoT dataset, using median to fill in missing values, resulting in a new dataset (we name it IoT_NET dataset) for simulating more complex industrial IoT scenarios under networked conditions. Median was used instead of mean because it is less susceptible to errors caused by outliers [50].

3.2 Cyclic encoding of temporal features

Many features in datasets are inherently cyclical, we use the sine and cosine components to cyclically encode the temporal features. Algorithm 1 presents the specific algorithm for encoding, where X_i represents the temporal features in the data, and i denotes the number of temporal features. The $maxValue$ represents the maximum value for a particular unit scale,

Table 2 Statistics records of the dataset

Type	IoT Dataset	Network Dataset
Normal	245000	300000
Backdoor	35000	20000
Injection	35000	20000
Password	35000	20000
DDos	25000	20000
Ransomware	16030	20000
XSS	6116	20000
Scanning	3973	20000
DOS	0	20000
MITM	0	1043

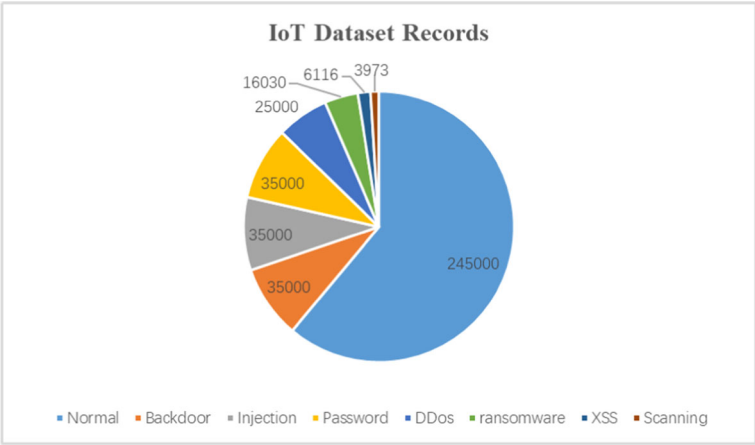

Fig. 2 Statistics records of the IoT dataset of Multiply

such as 23 for hours. In this algorithm, we add the expanded features $Xsin$ and $Xcos$ to the feature set while removing the corresponding original feature X. Instead of using separate numerical features for "hour," "minute," and "second," we utilize sine and cosine values for encoding the temporal information associated with "hour," "minute," and "second." In this way, we have transformed the one-dimensional time feature into two dimensions, effectively converting a straight line into a circle. When the circle completes one full clockwise rotation, it returns to the original point. This enables us to effectively utilize the cyclic nature of periodicity in machine learning and deep learning algorithms.

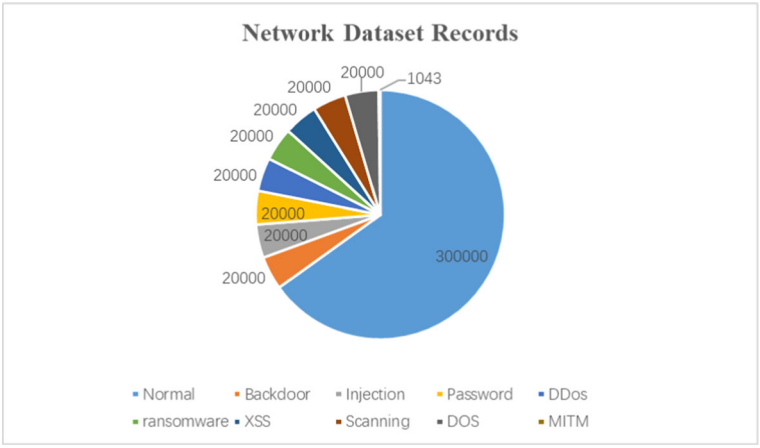

Fig. 3 Statistics records of the Network dataset of Multiply

Algorithm 1 Cyclic encoding of temporal features

Input: Temporal features X_i for $i = 1, 2, \ldots$
Output: X_i sine, X_i cosine
for X_i *in temporal features* **do**
  Calculate the maxValue of X_i;
  Calculate the sine component: $X_i sine = \sin(2\pi x/\text{maxValue})$;
  Calculate the cosine component: $X_i cosine = \cos(2\pi x/\text{maxValue})$;
  Delete X_i;
end
return X_i *sine,* X_i *cosine*;

3.3 Class imbalance processing

In the field of intrusion detection, class imbalance is a prevalent issue due to the significant disparity in the amount of attack data compared to normal data. To address this problem, the SMOTE is employed. Unlike basic random oversampling that duplicates existing samples, SMOTE generates synthetic samples for the minority class, thereby alleviating the risk of overfitting. SMOTE enhances random oversampling by creating new data points through a linear combination of comparable minority samples. Rather than directly replicating instances, SMOTE interpolates feature values between a minority sample and its nearest neighbors, generating new data points along the line connecting them. This process continues until the data becomes balanced, reducing the class imbalance. The specific algorithm for SMOTE oversampling is outlined in Algorithm 2, providing a step-by-step procedure for generating synthetic minority class samples. This technique effectively augments the minority class and contributes to mitigating the class imbalance issue in intrusion detection tasks.

In addition, we employed an improved redundant-based Tomek link removal under-sampling technique combined with SMOTE oversampling technique. Assuming sample points X_i and X_j belong to different classes, and $d(X_i, X_j)$ represents the distance between the two sample points. (X_i, X_j) is referred to as a Tomek link pair if there does not exist a third sample point such that $d(X_i, X_l) < d(X_i, X_j)$ or $d(X_j, X_l) < d(X_i, X_j)$ holds. In other words, X_i and X_j are the closest pair of points but belong to different classes (as shown in Fig. 4). It is easy to observe that if two sample points form a Tomek link, one of them is considered noise (deviating significantly from the normal distribution or both samples lying on the boundary of the two classes). When there are a considerable number of Tomek links in the data, it becomes challenging for classification algorithms to distinguish them, thus necessitating a reduction in the number of Tomek links.

Traditional methods remove the majority class sample if it forms a Tomek link with a minority class sample, thereby reducing the difference in quantity between the majority and minority classes and achieving the goal of Tomek link removal. Building upon the traditional Tomek link Removal under-sampling technique, we further integrated redundancy and noise factors by utilizing Euclidean distance as a similarity measure to identify majority class instances located near the decision boundary and highly similar to other majority instances. The detailed algorithm is presented in Algorithm 3. By combining the SMOTE oversampling technique with the improved redundant-based Tomek link Removal under-sampling technique, we effectively alleviate the class imbalance issue in the dataset. The framework diagram is illustrated in Fig. 5.

Algorithm 2 SMOTE algorithm

Input: T; N; k;
// T: Number of minority class examples, N: Amount of oversampling, K: Number of nearest neighbors
Output: $(N/100) \times T$; // Synthetic minority class samples
Variables: $Sample[\,][\,]$; // Array for original minority class samples
$newindex$; // Keeps a count of number of synthetic samples generated
$Synthetic[\,][\,]$; // Array for synthetic samples
if $N < 100$ **then**
  Randomize the T minority class samples;
  $T = (N/100) \times T$;
  $N = 100$;
end
$N = \text{int}(N/100)$;
for $i = 1$ **to** T **do**
  Calculate k nearest neighbors for i, and save the indices in the nn-array;
  Calculate $(N, i,$ nn-array$)$; // N: Instances to create, i: Original sample index, nn-array: Array of nearest neighbors
end
while $N \neq 0$ **do**
  $nn = \text{random}(1, k)$;
  for $attr = 1$ **to** *numattrs* **do**
    ; // numattrs: number of attributes
    Calculate: $dif = Sample[\text{nn-array}[nn]][attr] - Sample[i][attr]$;
    Calculate: $gap = \text{random}(0, 1)$;
    $Synthetic[newindex][attr] = Sample[i][attr] + gap \times dif$;
  end
  $newindex = newindex + 1$;
  $N = N - 1$;
end
return *Synthetic*

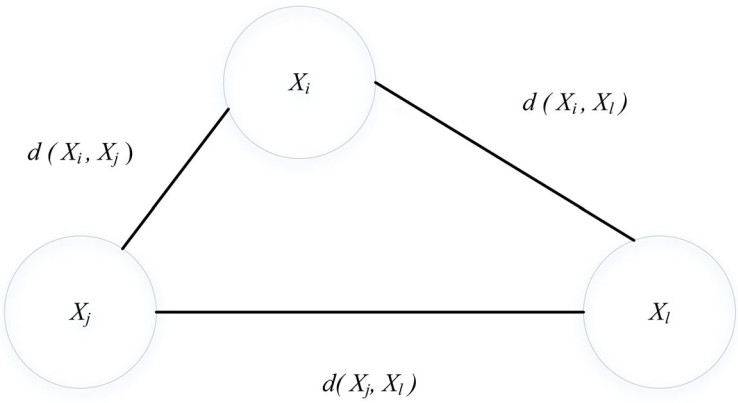

Fig. 4 Tomek link

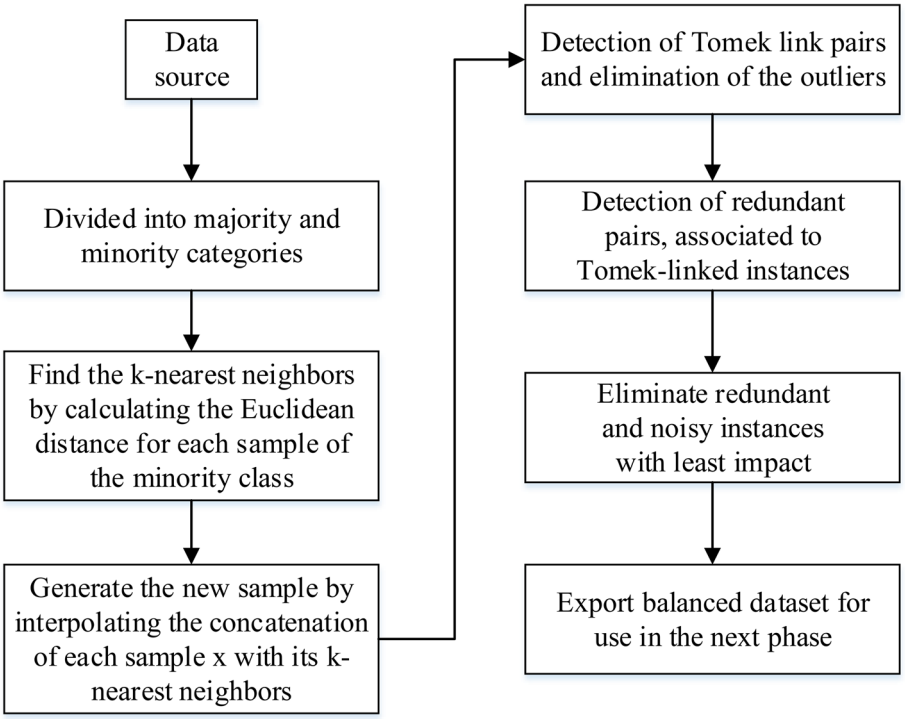

Fig. 5 Architecture of proposed class imbalance algorithm

3.4 Data preprocessing

Cleaning and preparation are core necessary stages to obtain good accuracy and accelerate the learning process before feeding data into Machine Learning methods to achieve high performance. This is usually achieved by removing unnecessary features that may degrade performance, transforming non-numerical features and fill of missing values.

3.4.1 Feature selection

For intrusion detection, various features must be examined, some of which are useful while others are not. Eliminating unnecessary features improves accuracy, reduces computation time, and reduces overfitting, thus improving performance. In the ToN_IoT dataset, there are many features in the Network dataset, and we use methods such as correlation to select features. The correlation is calculated using the Pearson correlation coefficient(PCCs), as shown in (1).

$$Pearson = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1)$$

Algorithm 3 Redundancy based Tomek link removal under-sampling

Input: In the dataset, S has n instances and the feature space has l attributes. Each instance is characterized as $(\vec{x}_a, \vec{y}_b)$ where $\vec{x}_a$ is the input vector for the a-th instance, denoted as $(x_{a1}, x_{a2}, \ldots, x_{al})$ with a total of l features, and $\vec{y}_a = (y_{a1}, y_{a2}, \ldots, y_c)$ is the designated class labels of the instances with a total of c classes; k is the number of nearest neighbors for detecting Tomek-link pair, Initialize to 1

Output: Revised dataset

The dataset is categorized into two subsets, S_{min} and S_{maj}

Tomek link pair and outlier detection:

for *all* $x_i \in S_{min}$ **do**

  Create a subset T_l with its k-NN from S_{maj}, as $T_l = \{(x_i, x_j) \mid x_i \in S_{\text{min}}, x_j \in S_{\text{maj}}\}$

  Calculate the index x_j as the number of x_i to which x_j is linked, for all $x_j \in T_l$

  Retain x_j in T_l, if index $x_j < t$ where t is the threshold, Otherwise, create a subset $OUT = \{x_j \mid x_j : \forall x_j, \text{index } x_j > t\}$, and is selected for direct elimination.

  Update T_l: $T_l = T_l - \{x_j\}$

  Update S_{maj}: $S_{\text{maj}} = S_{\text{maj}} - OUT$

end

Redundant pair detection:

for *all* $x_m \in S_{maj} (1 < m < j)$ **do**

  Create a set $Resub$ using Euclidean distance as the similarity measure

  Create a revised subset as $resub = \bigcap x_m \in S_{\text{maj}} \mid x_m \in T_l, x_m \in Resub$

end

if *either* x_u *or* x_v *satisfies the objective of elimination, delete* x_u *or* x_v, $(x_u, x_v \in resub)$ **then**

  Update S_{maj}: $S_{\text{maj}} = S_{\text{maj}} - resub$

end

Combine the updated S_{maj} and S_{min} to generate a balanced dataset

return the Revised dataset

3.4.2 Filling of missing values

ToN_IoT datasets often contain missing values. In order to construct useful analyses, these values must be treated correctly. We use the median of the whole column of the feature column to fill in the missing values because it is less susceptible to errors caused by outliers [50].

3.4.3 Converting non-numerical featuresFeature selection

We use a combination of label encoding and one-hot encoding to transform non-numeric features into numeric features that are more suitable for machine learning algorithms. For example, in the IoT device dataset, features with categorical values “high” and “low” are mapped to “0” and “1”, respectively. In the Network dataset, we use one-hot encoding to transform non-numeric features into numeric features. One-hot encoding is a method of representing parameters using 0 and 1 and using N-bit state registers to encode N states. The specific method is to use N-bit state registers to encode N states, with each state having its independent register bit, and at any time, only one bit is valid. Only one bit is 1, and the rest are 0.

3.4.4 Data normalization

Some features have larger values than others, and this can lead to inaccurate results since a model might be biased to the large feature values. Therefore, data standardization plays a crucial role in avoiding high-value features surpassing low-value features by scaling the

feature vector. Standard deviation normalization (Standard Scaler) is applied to numerical features in order to transform the processed data into a standard normal distribution, with a mean of 0 and a standard deviation of 1. The transformation function is (2).

$$X = \frac{x - \mu}{\sigma} \tag{2}$$

where μ is the mean of all sample data, and σ is the standard deviation of all sample data.

3.5 Models

The following lists the algorithms used in our experiments; We employed grid search to determine the optimal parameters, Table 3 shows the list of the used parameters in the experiments.

3.5.1 K-Nearest neighbor (KNN)

The KNN algorithm operates under the assumption of a provided training dataset wherein the class labels of instances have been determined. When classifying new instances, predictions are made by considering the classes of the k closest training instances, determined through majority voting or similar techniques. The Euclidean distance metric is employed to measure the proximity between instances, and it is computed according to the formula defined as (3).

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2} \tag{3}$$

3.5.2 Classification and regression trees (CART)

CART is a decision tree-based algorithm that constructs a binary tree structure from a training set. Each root node, also known as a non-leaf node, represents a single input variable (x) and a split point on that variable, while each leaf node (one with no children node) corresponds to an output (y) of the variable for prediction. The Gini coefficient itself reflects the uncertainty of the sample. When the Gini coefficient is smaller, the difference between samples is smaller and the uncertainty is lower. The classification process itself is a process of reducing uncertainty, i.e., increasing purity. Therefore, when constructing a classification tree, the CART algorithm chooses the attribute with the smallest Gini coefficient as the partition attribute. The formula for calculating the Gini coefficient is (4)

$$GINI(t) = 1 - \sum_{k}[p(C_k|t)]^2 \tag{4}$$

Table 3 List of the used parameters in the experiments

Model	Layers	Neurons/layer	Batch Size	Epochs	Activation Function	Optimizer
CNN	3	16/16/64	512	80	Relu, Softmax	Adam
LSTM	3	128/100/64	512	80	Tanh, Softmax	Adam
DNN	6	1024/768/512/256/128/10	512	80	Relu, Softmax	Adam

3.5.3 Random forest (RF)

Random Forest is an ensemble learning method that integrates multiple decision trees, each trained on randomly selected subsets of data points. RF is commonly employed for classification tasks, where it leverages the collective predictions of its constituent decision trees to classify observations. The final classification outcome is typically determined through majority voting, where the class with the most votes across the decision trees is assigned as the final prediction. Alternatively, weighted voting can be applied, where the individual decision trees' predictions are weighted, leading to a more nuanced classification result. The integration of multiple decision trees in RF enhances the model's robustness, improves accuracy, and helps mitigate the risk of overfitting.

3.5.4 Convolutional neural network (CNN)

CNN, short for Convolutional Neural Network, is a type of feed-forward neural network architecture that encompasses one or more convolutional layers, pooling layers, and fully connected layers. The Relu activation function is typically utilized between the hidden layers, as indicated by (5) , to introduce non-linearity and enhance the model's ability to learn complex patterns. In binary classification tasks, the output layer employs the sigmoid activation function, represented by (6) , to generate a probability value ranging between 0 and 1. For multi-class classification, the Softmax activation function, outlined in (7) , is utilized to compute class probabilities. The choice of loss function depends on the classification task at hand. In binary classification, the Binary Crossentropy loss function, depicted by (8) , quantifies the dissimilarity between the predicted probabilities and the true binary labels. On the other hand, in multi-class classification scenarios, the Sparse Categorical Crossentropy loss function, expressed in (9) , measures the discrepancy between the predicted class probabilities and the actual class labels. These loss functions play a critical role in optimizing the neural network parameters during the training process.

$$Relu(x) = \max(x, 0) \tag{5}$$

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{7}$$

$$\text{Binary\_crossentropy} = -\frac{1}{\text{outputsize}} \sum_{i=1}^{\text{outputsize}} y_i \log(\overline{Y}) + (1 - y_i) \log(1 - \overline{Y}) \tag{8}$$

$$\text{Spares\_categorical\_crossentropy} = -\sum_{i=1}^{\text{outputsize}} y_i \log(\overline{Y}_i) \tag{9}$$

3.5.5 Long short-term memory (LSTM)

LSTM, which stands for Long Short-Term Memory, is a specific type of recurrent neural network (RNN) architecture extensively utilized in the domain of deep learning. Its primary objective is to effectively model time series data and capture long-term dependencies by

incorporating a collection of memory units within the recurrent hidden layers. LSTM architecture typically consists of memory units and gates. The inclusion of distinct gate units in LSTM assists in mitigating the issues of gradient vanishing or explosion that can arise due to the loss of long sequence memory during RNN training. In the context of supervised classification, the LSTM model can be viewed as a problem of mapping input observations $x = (x_1, x_2, \ldots, x_n)$ to output labels y within the range of [0,1]. This mapping function is computed through the activation function of the network units, specifically designed for intrusion detection. Notably, the tanh activation function is employed between the hidden layers of LSTM, as illustrated in (10) .

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{10}$$

3.5.6 Deep neural networks (DNN)

DNN represents a neural network architecture comprising multiple hidden layers. These internal layers can be categorized into three distinct types: the input layer, hidden layers, and output layer. The Relu activation function is commonly employed between hidden layers to introduce non-linearity into the network. In binary classification tasks, the output layer employs the sigmoid activation function to produce a probability value ranging from 0 to 1. Conversely, for multi-class classification, the Softmax activation function is utilized to compute class probabilities. When dealing with binary classification, the loss function employed is Binary Crossentropy, which measures the dissimilarity between the predicted probabilities and the true binary labels. On the other hand, in multi-class classification scenarios, the loss function utilized is Sparse Categorical Crossentropy, which assesses the disparity between the predicted class probabilities and the actual class labels. These loss functions play a crucial role in optimizing the neural network parameters during the training process.

3.6 System framework

Our intrusion detection systems framework and workflow is shown in Fig. 6, we concatenate and merge seven IoT device datasets from the original IoT dataset. Then, we generate the IoT_Time dataset as well as the NET_Time dataset by cyclic encoding of temporal features on the IoT dataset and the network dataset. Subsequently, we merge the IoT dataset as well as the NET dataset using the same method as above to generate the IoT_NET dataset. Finally, we apply cyclic encoding of temporal features to the IoT_NET dataset, creating the IoT_NET_Time dataset. These six datasets (IoT dataset, IoT_Time dataset, Network dataset, NET_Time dataset, IoT_NET dataset, and IoT_NET_Time dataset) are used for subsequent analysis. This sequence of steps enhances the temporal perspective, underlining the dynamic aspects of the data and enriching the depth of analysis.We perform data preprocessing on the six datasets, including feature selection, converting non-numerical features, class imbalance processing, and data normalization. Then, we employ three commonly used machine learning algorithms in intrusion detection (KNN, RF, CART) and three deep learning algorithms (CNN, LSTM, DNN) to construct binary classifiers for categorizing data to differentiate between attack data and normal data. Following that, we engage in multiclass classification on the attack data, aiming to discern distinct attack types, thereby augmenting the attack database. This enhancement serves to provide attack data for the future development of new

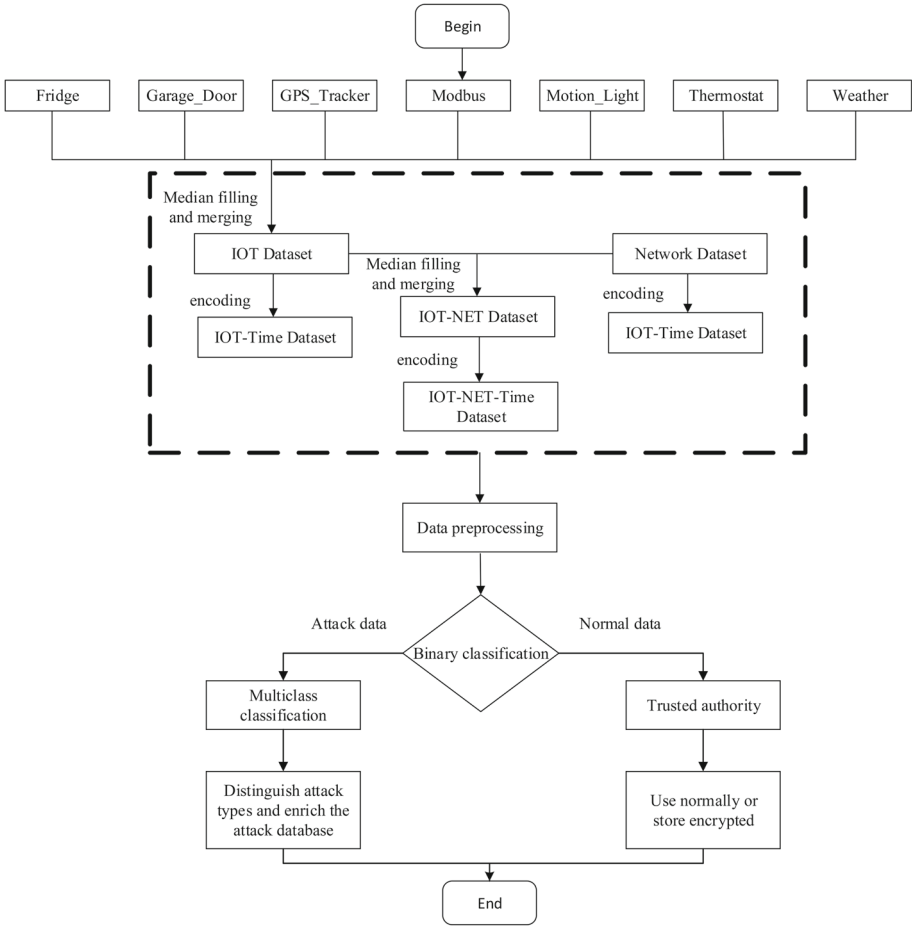

Fig. 6 System flow chart of the intrusion detection systems

intrusion detection systems. As for the normal data, it can be utilized as is or stored in an encrypted form for subsequent use.

4 Experimental results and analysis

As described in this section, we will describe the experimental setup, the evaluation metrics and discuss the experimental results and analysis. We will evaluate binary classification and multi-class classification for the synthetic datasets IoT dataset, IoT_Time dataset with temporal features, Network dataset, NET_Time dataset with temporal features, IoT_NET dataset, and IoT_NET_Time dataset with temporal features. As discussed in the previous sections, we evaluated the performance of six algorithms: KNN, RF, CART, CNN, LSTM,

and DNN. The parameters used were described in the previous sections, and all unspecified parameters were set to their default values. We used 4-fold cross-validation and calculated the average value of all evaluation metrics, which were displayed as the final results. Additionally, we provided the testing time.

4.1 Experiment environment

The experiments were conducted in Python version 3.8.15. Data processing and model evaluation were implemented with extension packages such as Numpy, Scipy, Pandas, and Scikit-learn v1.1.3. For model implementation, TensorFlow v2.11.0 was used together with Keras v2.11.0. All experiments were executed under Windows 11 OS with R7 6800HS 3.20 GHz CPU and 16GHZ RAM.

4.2 Classifier performance evaluation

The evaluation metrics adopted in this study include accuracy, precision, recall, and F1-score. Accuracy represents the percentage of correctly classified instances as normal or attack in overall efficiency. Precision represents the percentage of correctly identified attacks among all detected attacks. Recall represents the percentage of correctly detected attacks in the test dataset out of the total attacks. The F1-score is the weighted average of precision and recall. The formulas for calculating these metrics are (11) - (14). However, the data used in our work is imbalanced. Therefore, during testing, we use the weighted-F1 score instead of the F1 score as the evaluation metric. The weighted-F1 score is calculated by multiplying the F1 score for each class by the weight of the class, which is the ratio of the number of instances in that class to the total number of instances, and then summing up the results.

Here, True Positive(TP) represents instances that are actually positive and are correctly predicted as positive. True Negative(TN) represents instances that are actually negative and are correctly predicted as negative. False Positive(FP) represents instances that are actually negative but are falsely predicted as positive. False Negative(FN) represents instances that are actually positive but are falsely predicted as negative.

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

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

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

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

4.3 Binary classification results

Table 4 presents the binary classification results on the IoT dataset. The DNN algorithm achieved the highest accuracy at 84.88%, followed by the CART algorithm with an accuracy of 76.25%, and the RF algorithm had the lowest accuracy at 67.24%. CART performed well in the crucial metric of test time in intrusion detection systems. Table 5 provides the binary

Table 4 Evaluation of binary classification models on IoT dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	75.81	74.79	74.07	74.28	13.739
RF	67.24	69.16	63.94	63.61	0.474
CART	76.25	77.45	72.36	73.06	0.046
CNN	74.55	83.12	74.55	72.79	1.0
LSTM	67.78	82.28	67.78	66.74	0.504
DNN	84.88	89.08	84.88	85.04	0.941

Table 5 Evaluation of binary classification models on IoT_Time dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	44.19	54.33	43.21	37.28	15.690
RF	1.0	1.0	1.0	1.0	0.353
CART	1.0	1.0	1.0	1.0	0.046
CNN	54.39	78.72	54.39	49.28	0.451
LSTM	57.19	76.87	57.19	47.61	1.0
DNN	78.60	86.16	78.60	78.64	0.9

Table 6 Evaluation of binary classification models on Network dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	49.26	55.37	50.60	46.96	32.738
RF	75.82	80.20	70.68	68.70	0.754
CART	53.34	57.18	52.55	49.29	0.083
CNN	78.64	79.90	78.64	78.98	1.0
LSTM	84.21	87.85	84.21	83.83	1.0
DNN	89.88	92.13	89.88	90.09	1.0

Table 7 Evaluation of binary classification models on Network_Time dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	80.52	88.48	72.12	74.16	294.920
RF	90.45	93.60	86.33	88.66	0.986
CART	97.95	98.46	97.07	97.71	0.112
CNN	69.40	68.18	69.40	68.35	1.0
LSTM	84.53	88.10	84.53	84.16	1.0
DNN	90.33	92.41	90.33	90.53	1.0

classification results after incorporating temporal features. Both CART and RF algorithms significantly improved detection rates after adding encoded temporal features, achieving 100% accuracy in accurately distinguishing between attack and normal data. Additionally, compared to not using temporal features, there was no significant increase in test time, indicating an overall improvement in detection performance.

Table 6 presents the binary classification results on the Network dataset. The DNN algorithm still achieved the highest accuracy at 89.88%, followed by the LSTM algorithm with an accuracy of 84.21%, and the KNN algorithm had the lowest accuracy at 49.26%. Clearly, the accuracy of the three machine learning algorithms is significantly lower than that of the three deep learning algorithms. This is attributed to the higher dimensionality of the Network dataset, which is more complex than the lower-dimensional IoT dataset. Deep learning algorithms with higher depth demonstrate stronger nonlinear modeling capabilities and perform better in complex real-world tasks. In contrast, machine learning algorithms typically use shallow models, which may be effective for linearly separable or simple nonlinear tasks but may face difficulties in handling complex tasks.

After incorporating encoded temporal features, as shown in Table 7, the classification performance of DNN and LSTM algorithms improved, but the improvement was more significant for the three machine learning algorithms. CART achieved the highest accuracy of 97.95%, performing optimally in the critical test time metric for intrusion detection systems, with a test time of only 0.112 seconds. This indicates that fully utilizing temporal features can significantly enhance the algorithm's detection performance.

Table 8 displays the binary classification results for the IoT_NET dataset. The DNN algorithm still achieved the highest accuracy at 92.98%, followed by the LSTM algorithm with an accuracy of 90.06%. After merging the datasets, the accuracy showed a significant improvement compared to using either the IoT dataset (84.88%) or the Network dataset (89.88%) alone. This indicates that our model can adapt well to more complex IoT scenarios. By combining the IoT dataset with the Network dataset, we can leverage the relatively simple features of the IoT dataset to aid in classifying the Network dataset. This approach allows differentiation of certain data that was previously indistinguishable in the Network dataset, thereby improving overall performance and reducing the risk of attacks.

After introducing temporal features, as shown in Table 9, the accuracy of both CART and RF algorithms significantly improved, reaching 100% accuracy in accurately distinguishing between attack and normal data. Clearly, effectively utilizing temporal features in binary classification can distinguish previously indistinguishable attack data, suggesting that attackers may preferentially target IoT devices during specific time periods.

Overall, RF and CART have achieved the best performance, followed by DNN and LSTM, while CNN and KNN exhibited relatively poorer performance. This is attributed to the fact that in binary classification, RF and CART algorithms utilize shallow models, which may

Table 8 Evaluation of binary classification models on IoT_NET dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	64.19	81.91	56.69	51.33	1220.856
RF	70.61	76.13	63.74	62.35	1.748
CART	62.16	63.86	56.15	53.56	0.160
CNN	58.63	77.65	58.63	56.52	2.0
LSTM	90.06	91.69	90.06	89.96	2.0
DNN	92.98	94.08	92.98	93.08	2.0

Table 9 Evaluation of binary classification models on IoT_NET_Time dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	53.74	69.91	56.15	45.71	155.261
RF	1.0	1.0	1.0	1.0	2.782
CART	1.0	1.0	1.0	1.0	2.266
CNN	50.72	57.77	50.72	50.91	3.0
LSTM	89.23	91.13	89.23	89.10	2.0
DNN	92.19	93.55	92.19	92.30	2.0

be more effective for linearly separable or simple non-linear tasks. On the contrary, overly complex networks tend to amplify noise and introduce redundancy, resulting in suboptimal performance.

4.4 Multi-class classification results

The Tables 10 - 15 show various algorithms and their results using SMOTE oversampling and the combination of SMOTE oversampling with the improved redundant-based Tomek link removal under-sampling technique, where Model+S is mean the Model after using the SMOTE technique, Model+S+T is mean the Model after using the SMOTE with the improved redundant-based Tomek link removal technique.

Table 10 Evaluation of multi-class classification models on IoT dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	67.18	40.60	38.84	38.81	15.138
KNN+S	41.32	42.09	33.55	28.89	71.260
KNN+S+T	41.32	42.09	33.54	28.89	519.086
RF	44.98	45.70	37.69	32.17	0.591
RF+S	44.67	47.98	38.51	32.74	0.603
RF+S+T	44.34	46.48	38.32	32.46	0.954
CART	68.12	57.22	50.96	46.29	0.036
CART+S	64.25	47.23	52.54	45.98	0.035
CART+S+T	64.26	47.31	52.59	46.91	0.073
CNN	81.38	92.49	81.38	84.24	0.950
CNN+S	83.14	92.70	83.14	85.62	1.0
CNN+S+T	84.66	93.27	84.66	86.66	0.450
LSTM	84.26	93.00	84.26	86.52	1.0
LSTM+S	90.47	94.53	90.47	91.50	2.0
LSTM+S+T	90.96	95.06	90.96	92.27	1.0
DNN	89.45	94.24	89.45	90.67	0.9356
DNN+S	92.52	95.28	92.52	93.21	1.0
DNN+S+T	92.71	95.83	92.71	93.84	0.804

Table 10 displays the multi-class classification results of the IoT dataset, where the DNN algorithm achieves the highest accuracy of 89.45%, followed by the LSTM algorithm with an accuracy of 84.26%. Moreover, after applying SMOTE oversampling, the accuracy of DNN increases to 92.52%, and LSTM achieves 90.47%, showing a significant improvement compared to not using SMOTE oversampling. This is because in multi-class classification, there may be a situation where the number of samples in minority classes is much smaller than that in majority classes, causing the classifier to focus mainly on classifying the majority classes and neglecting the minority classes, resulting in poor classification performance. As mentioned in Section 3, the SMOTE oversampling technique effectively addresses this issue. When combining SMOTE oversampling with the improved redundant-based Tomek link removal under-sampling technique, the accuracy is significantly enhanced. For example, in the case of DNN, the accuracy increases by 3.26% to reach 92.71% compared to the original DNN. This improvement is slightly higher than using SMOTE oversampling alone.

After incorporating the encoded temporal features, as shown in Table 11,similar to the binary classification results, the CART algorithm achieves an accuracy of 98.65% after handling the class imbalance. Furthermore, CART exhibits the best performance in testing time, completing the classification in only 0.036 seconds. Compared to the best result without using encoded temporal features (92.71%), the performance is significantly improved (98.65%) by utilizing the temporal features. For most algorithms, the combination of SMOTE oversampling with the improved redundant-based Tomek link removal under-sampling technique results in a certain degree of improvement in accuracy compared to using SMOTE oversampling alone.

Table 11 Evaluation of multi-class classification models on IoT_Time dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	35.70	42.76	31.30	25.28	16.863
KNN+S	62.66	64.44	49.64	44.54	67.385
KNN+S+T	62.66	64.44	49.64	44.54	552.227
RF	98.65	96.73	98.07	96.07	0.499
RF+S	98.48	96.23	96.74	94.99	0.484
RF+S+T	98.65	97.10	98.07	96.28	0.838
CART	98.64	96.99	98.05	96.22	0.037
CART+S	98.65	97.10	98.07	96.28	0.036
CART+S+T	98.65	97.10	98.07	96.28	0.091
CNN	38.11	88.79	38.11	43.37	0.471
CNN+S	54.19	89.70	54.19	60.94	0.673
CNN+S+T	54.70	89.77	54.70	60.63	0.408
LSTM	76.91	91.71	76.91	80.69	1.0
LSTM+S	81.52	92.44	81.52	84.35	2.0
LSTM+S+T	81.93	93.16	81.93	85.09	1.0
DNN	87.49	93.72	87.49	89.09	0.892
DNN+S	92.88	95.42	92.88	93.51	0.860
DNN+S+T	93.72	96.53	93.72	94.70	0.794

Table 12 Evaluation of multi-class classification models on Network dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	58.02	79.69	78.23	70.56	32.592
KNN+S	90.07	87.44	95.34	89.15	234.236
KNN+S+T	90.07	87.44	95.34	89.15	814.835
RF	91.75	90.47	96.68	91.69	0.986
RF+S	91.57	91.65	95.86	92.19	1.030
RF+S+T	92.01	92.58	97.08	93.47	3.909
CART	60.83	75.75	75.10	67.14	0.062
CART+S	52.76	69.31	69.25	59.77	0.065
CART+S+T	52.76	70.40	69.25	60.85	2.056
CNN	90.79	91.76	90.79	91.20	1.0
CNN+S	98.22	98.41	98.22	98.27	1.0
CNN+S+T	98.45	98.92	98.45	**98.57**	2.0
LSTM	93.83	96.16	93.83	94.44	1.0
LSTM+S	95.46	96.81	95.46	95.81	3.0
LSTM+S+T	95.54	96.88	95.54	95.88	2.0
DNN	95.69	96.97	95.69	96.02	0.977
DNN+S	95.74	96.98	95.74	96.06	1.0
DNN+S+T	95.84	96.96	95.84	96.13	1.0

Table 12 presents the multi-class classification results of the Network dataset, where CNN achieves the highest accuracy. By applying SMOTE oversampling technique, CNN achieves an accuracy of 98.22%. Moreover, when combining SMOTE oversampling with Tomek-link under-sampling technique, the accuracy further improves to 98.45%, significantly outperforming other algorithms. This can be attributed to the relatively high dimensionality and complexity of the Network dataset in multi-class classification. Particularly, with the application of SMOTE oversampling, as the data volume increases, CNN's convolutional approach can more effectively extract features. Additionally, the dataset contains a considerable number of Tomek links, which are difficult to distinguish and can impact classification performance. Removing Tomek links results in improved classification performance. Even after incorporating the encoded temporal features, as shown in Table 13,CNN remains the algorithm with the highest accuracy, achieving 98.87%, slightly higher than the 98.45% without using temporal features. Furthermore, there are some improvements in recall, precision, and F1 scores. It can be observed that making full use of temporal features significantly enhances the detection performance of the algorithm.

The multi-class classification results of the IoT_NET dataset are presented in Table 14, with the DNN algorithm achieving the highest accuracy of 98.24% and ranking first. The LSTM algorithm follows with an accuracy of 97.60%, slightly lower than DNN. When using SMOTE oversampling and combining it with Tomek-link under-sampling technique, both accuracies are improved. Even in more complex IoT scenarios simulated under interconnected conditions, we still achieve excellent detection performance. After incorporating the encoded temporal features, the DNN algorithm ranks first with an accuracy of 98.71%, and the LSTM algorithm ranks second with an accuracy of 97.27%. This indicates that the proposed method effectively addresses the issue of data imbalance. As shown in Table 15,Compared to the best result of 98.24% without using encoded temporal features, there is an improvement in accuracy, which is rare considering the already high accuracy. This is because the dataset

Table 13 Evaluation of multi-class classification models on Network_Time dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	58.02	79.69	78.23	70.56	37.416
KNN+S	90.65	87.27	95.50	88.20	267.597
KNN+S+T	90.65	87.27	95.50	88.20	1895.053
RF	94.37	93.55	96.92	93.90	1.083
RF+S	94.30	95.56	94.96	93.64	1.048
RF+S+T	93.36	94.32	95.32	93.01	3.162
CART	67.09	88.05	86.41	79.10	0.062
CART+S	73.47	84.24	87.35	80.24	0.067
CART+S+T	71.63	83.43	85.89	78.78	1.153
CNN	89.09	91.02	89.86	89.86	1.0
CNN+S	98.30	98.54	98.36	98.36	1.0
CNN+S+T	98.87	99.36	99.18	99.18	2.0
LSTM	93.56	96.05	94.22	94.22	1.0
LSTM+S	96.46	97.35	96.46	96.69	3.0
LSTM+S+T	96.70	97.48	96.70	96.90	3.0
DNN	93.72	96.12	93.72	94.35	1.0
DNN+S	98.44	98.64	98.49	98.49	1.0
DNN+S+T	98.44	98.64	98.44	98.49	1.0

Table 14 Evaluation of multi-class classification models on IoT_NET dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	53.19	45.07	54.42	44.43	139.849
KNN+S	50.33	59.68	64.16	54.65	869.192
KNN+S+T	50.33	59.68	64.16	54.65	2012.963
RF	56.54	67.91	67.15	60.69	2.720
RF+S	55.79	68.39	66.39	60.81	2.710
RF+S+T	57.54	68.91	68.75	60.99	4.010
CART	45.14	50.56	54.39	44.35	0.128
CART+S	41.04	41.96	48.68	38.93	0.163
CART+S+T	46.04	51.96	54.68	44.93	1.013
CNN	68.12	89.09	68.12	74.47	2.0
CNN+S	86.65	86.65	94.03	88.76	2.0
CNN+S+T	74.29	92.08	74.28	79.35	8.0
LSTM	94.45	96.42	94.45	94.96	2.0
LSTM+S	97.60	98.04	97.60	97.71	5.00
LSTM+S+T	96.70	97.48	96.70	96.90	8.0
DNN	96.26	97.27	96.26	96.51	2.0
DNN+S	97.61	98.0 5	97.61	97.72	2.0
DNN+S+T	98.24	98.45	98.24	98.30	2.0

Table 15 Evaluation of multi-class classification models on IoT_NET_Time dataset

Model	ACC(%)	Pr(%)	Re(%)	F1(%)	Test time(s)
KNN	35.17	54.61	57.37	45.00	153.946
KNN+S	69.90	65.35	68.50	60.49	1009.159
KNN+S+T	69.90	65.35	68.50	60.49	2568.233
RF	93.73	90.48	92.06	87.54	2.087
RF+S	92.43	93.56	93.09	89.78	2.272
RF+S+T	93.76	90.49	92.63	89.98	4.102
CART	80.66	86.45	86.38	81.99	0.121
CART+S	82.17	84.31	85.57	79.14	0.213
CART+S+T	82.66	86.37	86.85	82.04	1.011
CNN	70.12	90.29	70.1	76.06	3.0
CNN+S	79.36	92.46	79.36	83.20	3.0
CNN+S+T	79.66	92.61	79.66	84.31	4.0
LSTM	89.72	94.91	89.72	91.15	3.0
LSTM+S	97.09	97.70	97.09	97.24	8.0
LSTM+S+T	97.27	97.95	97.27	97.65	3.0
DNN	96.09	97.18	96.09	96.36	2.0
DNN+S	98.26	98.50	98.26	98.32	2.0
DNN+S+T	98.71	98.84	98.71	98.74	2.0

inevitably contains some noise or indistinguishable data, making it challenging to achieve 100% accuracy in multi-class classification.

Overall, DNN have achieved the best performance, followed by CNN, LSTM and CART. This is because when the dataset has higher dimensions, deep learning algorithms with greater depth exhibit stronger nonlinear modeling capabilities, performing better in complex real-world tasks. It is worth noting that the KNN algorithm performs poorly in all the mentioned classifications and significantly lags behind other algorithms in the critical testing time for intrusion detection. This is because KNN is a lazy classifier that stores data during training and uses it for predictions during testing. With large data volumes and high feature dimensions, a significant amount of Euclidean distance calculations is required, resulting in slower testing phases. This explains why KNN is not an ideal choice for intrusion detection. When constructing an intrusion detection system, considering the characteristics of different algorithms yields better classification results. Combining SMOTE oversampling with improved redundant Tomek-link removal under-sampling technique leads to improved accuracy, demonstrating the effectiveness of these two methods in addressing data imbalance issues.

5 Conclusion

This paper presents a novel intrusion detection system for the internet of things, utilizing the ToN_IoT dataset. ToN_IoT offers a more comprehensive range of attacks compared to pre-

vious datasets like KDD-CUP99, NSL-KDD, and UNSW-NB15. Nevertheless, the ToN_IoT dataset encounters challenges such as class imbalance and missing values. Furthermore, existing research predominantly concentrates on single datasets, which fail to adequately simulate complex industrial IoT scenarios and underutilize temporal features. Therefore, we propose a methodology that combines datasets to simulate intricate industrial IoT scenarios in real-world contexts. To address the issue of data imbalance, we employ cyclic encoding to incorporate temporal features and propose a combination of the SMOTE oversampling technique and the improved redundant-based Tomek link Removal under-sampling technique. Subsequently, we preprocess the data, conduct feature selection, and handle missing values. Finally, we apply various machine learning and deep learning algorithms construct binary classifiers for categorizing data to differentiate between attack data and normal data. Following that, we engage in multiclass classification on the attack data, aiming to discern distinct attack types, thereby augmenting the attack database. This enhancement serves to provide attack data for the future development of new intrusion detection systems. As for the normal data, it can be utilized as is or stored in an encrypted form for subsequent use. The intrusion detection system serves as a valuable reference for the construction of intrusion detection systems in other datasets or scenarios.

Upon evaluating diverse machine learning and deep learning algorithms, we deduce that the CART algorithm performs well in relatively simpler scenarios, exhibiting the shortest testing time among other algorithms while achieving excellent detection performance. However, in complex scenarios, the DNN and CNN algorithms outperform others, displaying higher accuracy. Compared to existing research conducted on the same dataset, our investigation demonstrates enhancements in accuracy, precision, recall, and F1 score. This demonstrates the effectiveness of our work in simulating more complex IoT scenarios and addressing class imbalance issues. In future endeavors, we intend to explore unsupervised approaches to intrusion detection algorithms on the ToN_IoT dataset, utilizing auto-encoder-related methods.

Funding This work was supported in part by National Key R&D Program of China under Grant 2021YFB2012300, the National Natural Science Foundation of China under Grant 62173101, Basic and Applied Basic Research Funding of Guangdong Province under Grant 2022A1515011558 and Grant 2022A1515010865, the Guangzhou Science and Technology Funding under Grant 202201020217, the Key Laboratory of Guangdong Higher Education Institutes under Grant 2023KSYS002.

Data Availability The source code and data that support the findings of this study are available on https://github.com/czant1977/Intrusion-Detection

Declarations

Conflicts of interests/competing interests 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.

References

1. Saha HN, Mandal A, Sinha A (2017) Recent trends in the internet of things. In: 2017 IEEE 7th Annual computing and communication workshop and conference (CCWC), pp 1–4. IEEE
2. Sharma A, Zheng Z, Kim J, Bhaskar A, Haque MM (2021) Assessing traffic disturbance, efficiency, and safety of the mixed traffic flow of connected vehicles and traditional vehicles by considering human factors. Transp Res C: Emerg Technol 124:102934
3. Asha P, Natrayan L, Geetha B, Beulah JR, Sumathy R, Varalakshmi G, Neelakandan S (2022) Iot enabled environmental toxicology for air pollution monitoring using ai techniques. Environ Res 205:112574

4. Stojkoska BLR, Trivodaliev KV (2017) A review of internet of things for smart home: Challenges and solutions. J Cleaner Prod 140:1454–1464
5. Alshamrani M (2022) Iot and artificial intelligence implementations for remote healthcare monitoring systems: A survey. J King Saud Univ-Comput Inf Sci 34(8):4687–4701
6. Balaji M, Roy SK (2017) Value co-creation with internet of things technology in the retail industry. J Market Manage 33(1–2):7–31
7. Abiodun OI, Abiodun EO, Alawida M, Alkhawaldeh RS, Arshad H (2021) A review on the security of the internet of things: Challenges and solutions. Wirel Personal Commun 119:2603–2637
8. Zipperle M, Gottwalt F, Chang E, Dillon T (2022) Provenance-based intrusion detection systems: A survey. ACM Comput Surv 55(7):1–36
9. Pradeepthi C, Maheswari BU (2023) Network intrusion detection and prevention strategy with data encryption using hybrid detection classifier. Multimed Tools App 1–32
10. Yang Z, Liu X, Li T, Wu D, Wang J, Zhao Y, Han H (2022) A systematic literature review of methods and datasets for anomaly-based network intrusion detection. Comput Secur 116:102675
11. Kilincer IF, Ertam F, Sengur A (2022) A comprehensive intrusion detection framework using boosting algorithms. Comput Electr Eng 100:107869
12. Maldonado J, Riff MC, Neveu B (2022) A review of recent approaches on wrapper feature selection for intrusion detection. Expert Syst Appl 198:116822
13. Falco G, Caldera C, Shrobe H (2018) Iiot cybersecurity risk modeling for scada systems. IEEE Int Things J 5(6):4486–4495
14. Antonakakis M, April T, Bailey M, Bernhard M, Bursztein E, Cochran J, Durumeric Z, Halderman JA, Invernizzi L, Kallitsis M et al (2017) Understanding the mirai botnet. In: 26th USENIX security symposium (USENIX Security 17), pp 1093–1110
15. Chanal PM, Kakkasageri MS (2020) Security and privacy in iot: a survey. Wirel Personal Commun 115(2):1667–1693
16. Khraisat A, Alazab A (2021) A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecurity 4:1–27
17. Alazab M, Khurma RA, Awajan A, Camacho D (2022) A new intrusion detection system based on moth-flame optimizer algorithm. Expert Syst Appl 210:118439
18. Hsu C-Y, Wang S, Qiao Y (2021) Intrusion detection by machine learning for multimedia platform. Multimed Tools App 80(19):29643–29656
19. Lee S-W, Mohammadi M, Rashidi S, Rahmani AM, Masdari M, Hosseinzadeh M et al (2021) Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review. J Netw Comput App 187:103111
20. Kumar R, Kumar P, Tripathi R, Gupta GP, Garg S, Hassan MM (2022) A distributed intrusion detection system to detect ddos attacks in blockchain-enabled iot network. J Parallel Distrib Comput 164:55–68
21. Thakkar A, Lohiya R (2023) Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system. Inf Fus 90:353–363
22. Chaabouni N, Mosbah M, Zemmari A, Sauvignac C, Faruki P (2019) Network intrusion detection for iot security based on learning techniques. IEEE Commun Surv Tutor 21(3):2671–2701
23. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp 1–6. Ieee
24. Moustafa N, Slay J (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military communications and information systems conference (MilCIS), pp 1–6. IEEE
25. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 1:108–116
26. Moustafa N (2021) A new distributed architecture for evaluating ai-based security systems at the edge: Network ton_iot datasets. Sustain Cities Soc 72:102994
27. Debar H, Becker M, Siboni D (1992) A neural network component for an intrusion detection system. IEEE Symp Secur Priv 727:240–250
28. Wang Z, Jiang D, Huo L, Yang W (2021) An efficient network intrusion detection approach based on deep learning. Wirel Netw 1–14
29. Ravi N, Shalinie SM (2020) Semisupervised-learning-based security to detect and mitigate intrusions in iot network. IEEE Int Things J 7(11):11041–11052
30. Alzaqebah A, Aljarah I, Al-Kadi O (2023) A hierarchical intrusion detection system based on extreme learning machine and nature-inspired optimization. Comput Secur 124:102957

31. Mirsky Y, Doitshman T, Elovici Y, Shabtai A (2018) Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv preprint arXiv:1802.09089
32. Hazman C, Guezzaz A, Benkirane S, Azrour M (2023) Toward an intrusion detection model for iot-based smart environments. Multimed Tools App 1–22
33. Mohy-eddine M, Guezzaz A, Benkirane S, Azrour M (2023) An efficient network intrusion detection model for iot security using k-nn classifier and feature selection. Multimed Tools App 1–19
34. Caville E, Lo WW, Layeghy S, Portmann M (2022) Anomal-e: A self-supervised network intrusion detection system based on graph neural networks. Knowl-Based Syst 258:110030
35. Al-Yaseen WL, Idrees AK, Almasoudy FH (2022) Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system. Pattern Recognit 132:108912
36. Laghrissi F, Douzi S, Douzi K, Hssina B (2021) Intrusion detection systems using long short-term memory (lstm). J Big Data 8(1):65
37. Mushtaq E, Zameer A, Umer M, Abbasi AA (2022) A two-stage intrusion detection system with auto-encoder and lstms. Appl Soft Comput 121:108768
38. Rashid MM, Kamruzzaman J, Hassan MM, Imam T, Wibowo S, Gordon S, Fortino G (2022) Adversarial training for deep learning-based cyberattack detection in iot-based smart city applications. Comput Secur 120:102783
39. Debicha I, Bauwens R, Debatty T, Dricot J-M, Kenaza T, Mees W (2023) Tad: Transfer learning-based multi-adversarial detection of evasion attacks against network intrusion detection systems. Futur Gener Comput Syst 138:185–197
40. Sarhan M, Layeghy S, Moustafa N, Portmann M (2021) Netflow datasets for machine learning-based network intrusion detection systems. In: Big data technologies and applications: 10th EAI international conference, BDTA 2020, and 13th EAI International conference on wireless internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings 10, pp 117–135. Springer
41. Lo WW, Layeghy S, Sarhan M, Gallagher M, Portmann M (2022) E-graphsage: A graph neural network based intrusion detection system for iot. In: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, pp 1–9. IEEE
42. Wyk A (2018) Encoding cyclical features for deep learning. Rep, EPI-USE Lab, Pretoria, South Africa, Tech
43. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
44. Tang Y, Zhang Y-Q, Chawla NV, Krasser S (2008) Svms modeling for highly imbalanced classification. IEEE Trans Syst Man Cybernet Part B (Cybernetics) 39(1):281–288
45. Han H, Wang W-Y, Mao B-H (2005) Borderline-smote: a new over-sampling method in imbalanced data sets learning. In: International conference on intelligent computing, pp 878–887. Springer
46. Gupta N, Jindal V, Bedi P (2022) Cse-ids: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems. Comput Secur 112:102499
47. Ding H, Chen L, Dong L, Fu Z, Cui X (2022) Imbalanced data classification: A knn and generative adversarial networks-based hybrid approach for intrusion detection. Futur Gener Comput Syst 131:240–254
48. Booij TM, Chiscop I, Meeuwissen E, Moustafa N, Den Hartog FT (2021) Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. IEEE Int Things J 9(1):485–496
49. Friha O, Ferrag MA, Shu L, Maglaras L, Choo K-KR, Nafaa M (2022) Felids: Federated learning-based intrusion detection system for agricultural internet of things. J Parallel Distrib Comput 165:17–31
50. Douglass MJ (2020) Book Review: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow, by Aurélien Géron: O'Reilly Media, 2019, 600 pp ISBN: 978-1-492-03264-9. Springer