electronics

MDPI

Article

Towards Effective Feature Selection for IoT Botnet Attack Detection Using a Genetic Algorithm

Xiangyu Liu and Yanhui Du *

College of Information and Cyber Security, People's Public Security University of China, Beijing 100038, China
* Correspondence: duyanhui@ppsuc.edu.cn

Abstract: With the large-scale use of the Internet of Things, security issues have become increasingly prominent. The accurate detection of network attacks in the IoT environment with limited resources is a key problem that urgently needs to be solved. The intrusion detection system based on network traffic characteristics is one of the solutions for IoT security. However, the intrusion detection system has the problem of a large number of traffic features, which makes training and detection slow. Aiming at this problem, this work proposes a feature selection method based on a genetic algorithm. The experiments performed on the Bot-IoT botnet detection dataset show that this method successfully selects 6 features from the original 40 features, with a detection accuracy of 99.98% and an F1-score of 99.63%. Compared with other methods and without feature selection, this method has advantages in training time and detection accuracy.

Keywords: feature selection; IoT botnet; intrusion detection; genetic algorithm

1. Introduction

The Internet of Things (IoT) contains tens of billions of embedded computing devices that connect the physical environment and the Internet [1]. With the large-scale use and growth of the Internet of Things devices, the problem of information security is becoming increasingly serious. For the Mirai botnet [2], the DDoS (distributed denial of service) attack it launched caused the service paralysis of many Internet services and even ISPs. The Mirai botnet has shifted from traditional attacks on network nodes to infecting IoT devices to launch large-scale attacks, marking the general trend of using IoT devices to implement network attacks. The way Mirai works is by infecting a large number of IoT devices, such as webcams or routers, to launch a DDoS attack. Because many IoT devices have weak passwords and weak security configurations, viruses such as Mirai can easily break through the vulnerable security lines of IoT devices. This also reflects the importance of intrusion detection systems in IoT infrastructure.

There are several datasets created by researchers for testing intrusion detection methods. However, the number of features in these datasets is usually large. Applying intrusion detection methods directly on the original feature set may have some drawbacks, such as the high model complexity and overfitting. Thus, utilizing feature selection techniques to reduce the number of features helps speed up detection model inference. In this work, we propose a feature selection technique based on genetic algorithm for botnet detection in IoT environments.

Citation: Liu, X.; Du, Y. Towards Effective Feature Selection for IoT Botnet Attack Detection Using a Genetic Algorithm. *Electronics* **2023**, *12*, 1260. https://doi.org/10.3390/electronics12051260

Academic Editor: Elif Bilge Kavun

Received: 7 February 2023
Revised: 1 March 2023
Accepted: 6 March 2023
Published: 6 March 2023

1.1. Intrusion Detection System

Intrusion detection systems are devices and software that defend against intrusions and attacks in network infrastructure. The system's function is not to reduce network attacks but to protect the internal network from attacks. It identifies anomalies by monitoring network traffic or logs. Once an abnormality is detected, it will send alarms to the system administrator. If an attack is detected, the system will react proactively, take measures to stop the attack, and collect intrusion evidence [3].

Intrusion detection systems can be divided into signature-based detection and anomaly-based detection according to different detection methods [4]. Signature-based detection is based on pattern matching. Its main disadvantage is that it cannot detect unknown intrusion behaviors, and the false negative rate is high. Moreover, its implementation depends on the operating system, making it difficult to deploy on different operating systems. Building an anomaly-based intrusion detection system requires a large amount of network traffic data to model normal behaviors to judge abnormal behaviors. Anomaly-based detection algorithms mainly include statistics-based [5], data mining-based [6], and machine learning-based [7] algorithms. The advantage of anomaly-based detection is that it has a stronger ability to discover new types of attacks. However, anomaly detection is immature and has a higher false positive rate.

This paper proposes an IoT botnet detection method based on machine learning. Machine learning studies use data or prior experience to train a model for classification and prediction. It is widely used in many scientific fields and engineering applications, such as natural language processing, computer vision, and sentiment analysis. As the effectiveness of machine learning algorithms has been proven in different application fields, many researchers have applied machine learning algorithms to network intrusion detection and have achieved satisfactory results and performance [8].

1.2. Feature Selection

Machine learning and data processing are inseparable from each other. With the rapid growth of the amount of data in cybersecurity applications, more efficient data management and processing methods are required. In high-dimensional datasets, feature selection compresses the data dimensionality by reducing the number of features in the dataset. One problem with high-dimensional datasets is the curse of dimensionality, and another is overfitting. Too many features could affect the accuracy of the machine learning model. At the same time, a large number of features means more storage and computational resource requirements. Dimension reduction is an effective way to handle high-dimensional data. Dimension reduction can be performed through feature selection [9]. Feature selection refers to selecting the most suitable subset of features from the full feature set to help machine learning models. In practice, the dataset usually has noise and redundant or irrelevant features. Removing redundant or useless features from the data helps improve the accuracy of the classifier while reducing the false positive rate and false negative rate.

The general process for feature selection can be defined as selecting the best feature subset

$$X = \{x_i | i = 1, \ldots, d; x_i \in Y\} \tag{1}$$

from the original feature set

$$Y = \{y_i | i = 1, \ldots, D\} \tag{2}$$

where D is the number of original features and d is the size of the selected feature subset. A feature selection process should have a criterion function $J(X)$ for evaluating a selected feature subset X. The criterion function can be accuracy, precision, or other evaluation metrics we are interested in. The objective of feature selection is finding the best feature subset $X^{opt} \subseteq Y$ such that J reaches the maximum

$$J(X^{opt}) = \max_{X \subseteq Y} J(X) \tag{3}$$

1.3. Problem Statement

According to the difference in selection strategy, feature selection methods can be divided into three classes: wrapper, filter, and embedded techniques [10]. Wrapper and filter techniques work in separate ways, while embedded techniques combine the two strategies together. The wrapper method searches for a subset of original features and then uses a machine learning algorithm to evaluate the selected subset of features. The process is run iteratively utilizing fulfilling the stop criterion. The disadvantage of the wrapper method is that it is computationally expensive, especially for high-dimensional datasets. To address this, some optimization methods, such as best-first search, hill climbing, branch-and-bound search [11], and genetic algorithms, can be used, which help to find local optimal solutions. Feature selection based on filtering methods is independent of the learning algorithm; however, there is no guidance from the learning algorithm, so the selected features may not be optimal. The filter method has two steps: first, feature rankings are calculated, and second, features are extracted according to the ranking obtained.

This work aims to address the selection of original features, with the goal of obtaining a subset of these features to positively affect the accuracy of IDS. The proposed method is based on the wrapper feature selection method. With the guidance of a specific machine learning algorithm, the genetic algorithm is used to search for the best feature subset. The genetic algorithm converges quickly and provides the best feature subset with a higher detection rate.

1.4. Contributions

This work provides a genetic algorithm-based feature selection solution for an intrusion detection system in an IoT environment that is faced with threats from botnet attacks. By performing a set of experiments to find the optimum feature subset and optimal number of features, the genetic algorithm converges quickly and finds the optimum feature subset that enables better classification results. The contributions of this work are listed as follows:

- Development of genetic algorithm-based feature selection method for intrusion detection systems of IoT botnet attacks;
- The machine learning model is trained using the optimal feature subset selected though the genetic algorithm;
- Performance evaluation on IoT botnet attack detection dataset Bot-IoT with better results compared with state-of-the-art intrusion detection systems.

2. Related Work and Problem Formulation

Today, the amount of traffic data in the IoT environment has increased exponentially. Due to the limited nature of the computational resources of IoT devices, it requires an efficient feature selection method for IDS to detect botnet attacks using minimum computational resources. This section presents the related research on feature selection methods using genetic algorithms for intrusion detection and problem formulation.

2.1. Related Studies

Various methods have been proposed for solving the problem of high-dimensional data in intrusion detection datasets. These methods used different techniques for feature selection to decrease the number of features used for classification, which could either promote the performance of the classifier or shorten the training and detection time of the IDS. Many of these studies were based on genetic algorithms, Table 1 shows a summary of related studies on genetic algorithm-based feature selection in IDS.

Table 1. Related studies on genetic algorithm-based feature selection in IDS.

Year	Reference	Methodology	Dataset	Model	Performance
2005	Stein et al. [12]	GA	KDD Cup 99	Decision Tree	Detection error rate: 0.095 for U2R and 0.199 for R2L
2012	Kannan et al. [13]	GA	KDD Cup 99	Fuzzy SVM	Detection rate of 98.51% and error rate of 3.13%
2017	Raman et al. [14]	Hypergraph-based GA	NSL-KDD	SVM	Detection rate of 97.14% and false alarm rate of 0.83%
2021	Mehanović et al. [15]	GA in MapReduce	NSL-KDD	SVM, ANN, RT, LR, and NB	Accuracy: 90.45%
2021	Halim et al. [16]	GA	CIRA-CIC-DOHBrw-2020, UNSW-NB15, Bot-IoT	SVM, kNN, XGBoost	Accuracy: 99.80%

In 2005, Stein et al. [12] introduced a method using a genetic algorithm to select a subset of original features and then used the selected feature subset to train a decision tree to increase the detection rate and decrease the false alarm rate in network intrusion detection. They used the KDDCUP 99 dataset for training and testing. Their experiments showed that the resulting classifier can have a better performance compared to those with a full feature set. KDD Cup 99 has 41 original features, and there were 32 features in their final optimal feature subset. This work achieved a detection error rate of 0.095 for U2R and 0.199 for R2L.

In 2012, Kannan et al. [13] developed an intrusion detection model combining genetic-based feature selection and a fuzzy support vector machine for effective classification in a cloud computing environment. They found that the detection accuracy improved when the fuzzy SVM was used with the selected feature subset. They tested the method with the KDD Cup 99 dataset and achieved a detection rate of 98.51% and an error rate of 3.13%.

In 2017, Raman et al. [14] proposed an IDS that uses a hypergraph-based genetic algorithm (HG-GA) for parameter tuning and feature selection in SVM. The HG-GA was used to generate the initial population to speed up the convergence of the genetic algorithm and to avoid local minima. They also developed a weighted objective function to find the right balance between the detection rate, false alarm rate, and the optimal number of features. The technique was tested using the NSL-KDD intrusion detection dataset and achieved a detection rate of 97.14% and a false alarm rate of 0.83% with the optimal feature subset.

In 2021, Mehanović et al. [15] developed an intrusion detection technique using a cloud-based parallel genetic algorithm for feature selection. This work aimed to shorten the processing time of feature selection by migrating the algorithm to a MapReduce framework that is suitable for parallel computing. The representative machine learning methods, SVM, ANN, RT, LR, and NB, were embedded into the implementation for feature selection. The methods were applied to the NSL-KDD dataset and achieved 90.45% accuracy with the number of features reduced from 40 to 10.

In 2021, Halim et al. [16] proposed an effective genetic algorithm-based feature selection (GbFS) method for intrusion detection. They designed a novel fitness function for a genetic algorithm. The method was evaluated on three datasets, namely, CIRA-CIC-DOHBrw-2020, UNSW-NB15, and Bot-IoT. By using the SVM, kNN, and XGBoost classifiers, they achieved a maximum accuracy of 99.80%.

In 2022, Mojtahedi et al. [17] developed a NIDS by utilizing a combination of two algorithms, the whale optimization algorithm (WOA) and genetic algorithm, to perform feature selection. The kNN algorithm was then used for classification. The study was carried out on the KDD Cup 99 dataset, and the results indicated a 99.85% accuracy rate.

2.2. Problem Formulation

An intrusion detection system (IDS) plays a significant role in cybersecurity; it protects the network from threats of external attacks. In the development of IDS, machine learning is widely used for its promising performance and results. The development of ML-based IDSs starts with network traffic analysis. Network traffic data are usually of high dimensionality, namely, they have a large number of features. Too many features may have an impact on the performance of machine learning models. Thus, reducing the number of features while preserving the original information as much as possible can improve the accuracy and running time of the model. Reducing the number of features refers to feature selection. There are many existing methods for feature selection, such as forward selection and backward elimination; however, these algorithms are very time-consuming when the number of features and the dataset scale are large. This work aims to explore the use of genetic algorithms to optimize the feature selection process, which uses as few features as possible to achieve the best classification results for the machine learning models.

3. Methodology

This section presents the feature selection process based on the genetic algorithm for the IoT botnet detection system proposed in this paper. The objective is to use as few features as possible while improving the accuracy of the detection system and reducing the false alarm rate. The figure shows the block diagram of the entire system. There are three phases of the system: (1) data preprocessing; (2) feature selection based on a genetic algorithm; and (3) a classification module to detect attacks. The proposed method uses a genetic algorithm for feature selection, which selects the smallest feature set and obtains optimized classification results. Each phase will be described as follows.

3.1. Data Preprocessing

Data preprocessing is extremely important to preprocess data before providing it to a machine learning model. Data preprocessing includes removing duplicates, input and output encoding, normalization, and scaling. In fact, many features in the dataset are not in suitable form for machine learning models. The model can accept input only in numeric form. Therefore, categorical features need to be encoded. Commonly used encoding schemes include one-hot encoding and 1-of-n encoding. In this work, we use 1-of-n encoding for convenience. The values of distinctive features usually have different scales. All numeric features can be scaled to 0–1 by using the standard scaler. Scaling could facilitate the training process of machine learning models. The formula for the standard scaler is

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{5}$$

and standard deviation:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)} \tag{6}$$

3.2. Genetic Algorithm for Feature Selection

Genetic algorithms mimic the Darwinian nature selection process to search for an optimal solution space [18]. First, an initial set of candidate solutions is created, and scores are calculated according to the predefined fitness function. This set of solutions is called a population, and each solution is an individual in the genetic algorithm. Individuals with higher fitness function values are more likely to survive in the evolutionary process, whereas lower fitness function values are filtered out. The solutions are represented by vectors of a certain length compared with the chromosomes containing genes. The chromosomes undergo crossover, which mimics genetic reproduction and is also subject to random mutations. The results of crossover and mutation operations make up the individuals in the next generation. This process is continuously repeated until the process converges to a certain solution or the maximum number if iterations is reached.

The genetic algorithm is a kind of optimization algorithm used for searching an optimal solution in the solution space. For feature selection, the optimal solution is the best feature subset with the best classification performance. In feature selection, each feature is represented as a binary variable, where 1 indicates this feature is selected, and 0 means this feature is ignored. Therefore, each solution can be seen as a binary vector, where each position represents one feature. In the genetic algorithm, the solution is represented by a chromosome. The crossover operation defines combining two parent chromosomes to produce offspring for the next generation. When selecting parents, a random crossover point is chosen. The first portion of the first child is taken from parent 1, and the second portion is taken from parent 2. The remaining portions of the parents are used to create the second child. This process is repeated until a new population is generated. The mutation operation is defined as swapping the bit value with a mutation probability. The fitness function is a measure of model performance, such as classification accuracy or F1-score.

The genetic algorithm for feature selection includes the following steps:

1. Population initialization: Randomly generate a set of binary vectors as the initial population;
2. Fitness Function: Define a fitness function to evaluate the quality of each solution. In feature selection problems, the fitness function can be classification accuracy or other metrics that measure model performance;
3. Selection Operation: Use selection operators to select a group of parents from the population, where individuals with higher fitness have a greater chance of being selected;
4. Crossover Operation: Use crossover operators to exchange genetic segments between parent individuals, producing new offspring;
5. Mutation Operation: Use mutation operators to mutate certain genes of offspring individuals to increase the diversity of the population;
6. Repeat Steps 3 to 5 until the termination condition is met (e.g., reaching the maximum number of iterations or reaching an appropriate fitness threshold);
7. Return the Best Solution: Return the individual with the highest fitness in the population as the final feature selection result.

4. Experimental Results and Analysis

In this section, we describe the dataset and experimental results. The proposed method is compared with four related solutions using the same dataset. The experimental results of the proposed method are reported using the original feature set and the selected feature subset. The whole solution is coded in Python 3.8 utilizing scikit-learn [19]. The machine used for experiments had an Intel i9-12900H CPU, a Windows 10 OS, and RAM of 32.0 GB.

4.1. Dataset Description

Experiments were conducted on the benchmark dataset Bot-IoT [20,21] using the proposed feature selection method to select the appropriate feature set for classification. The Bot-IoT dataset was created by the Cyber Range Lab of UNSW Canberra in a realistic network environment. This network environment contains a mixture of benign and botnet traffic, and the original PCAP files contain over 72 million records. The traffic flow extracted from the PCAP files includes 10 types of attacks. The dataset creator also provides a smaller dataset that contains 5% of the original network traffic (over 3 million records) with additional generated features. Table 2 lists the number of samples for each attack and benign traffic in the 5% dataset.

Table 2. Class distribution of normal and attack traffic in the Bot-IoT dataset.

	Class	**# Samples**
1	Service Scanning	73,168
2	OS Fingerprinting	17,914
3	DDoS TCP	977,380
4	DDoS UDP	948,255
5	DDoS HTTP	989
6	DoS TCP	615,800
7	DoS UDP	1,032,975
8	DoS HTTP	1485
9	Normal	477
10	Keylogging	73
11	Data Exfiltration	6

4.2. Evaluation Metrics

To evaluate the performance of the model, this paper uses the accuracy, precision, recall, F1-score, and ROC curve for evaluation. These evaluation metrics are based on a confusion matrix. A confusion matrix is a quantitative summary of the predicted values for a classification problem. From a confusion matrix, the following four values can be derived: true positive (TP), false positive (FP), false negative (FN), and true negative (TN).

According to the above four values, the precision, precision, recall, and F1 score can be calculated. The precision, precision, recall, and F1 score are defined as formulas.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

$$precision = \frac{TP}{TP + FP} \tag{8}$$

$$recall = \frac{TP}{TP + FN} \tag{9}$$

$$F1 - score = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{10}$$

4.3. Experimental Results with Full Features

The first experiment is performed on the benchmark dataset Bot-IoT using the complete feature set. According to the three steps described above, this experiment skips the second step and directly uses the complete feature set and classification model. All classification tasks use a 70:30 train test split.

The Bot-IoT dataset has 29 original features plus 14 computed features, for a total of 43 features, where pkSeqID, seq is the sequence number of the row and flow, saddr, sport, daddr, and dport are the identifiers of the flow. These should not be put into the model as features because the classes of the dataset are arranged in sequential order, and the sequence number and identifier will leak the class label information, which leads to better classification results, so these six features are removed. From Table 2, it can be noticed that there are two classes of attacks, namely, keylogging and data exfiltration, whose numbers of data samples are too small. It could have a negative influence on the classification results, and the traditional machine learning system cannot obtain enough information from few data samples. Thus, in the following experiments, these two classes are ignored.

Three machine learning algorithms, namely, k-nearest neighbors (kNN), random forest (RF), and decision tree (DT), are tested on the original full features in the Bot-IoT dataset. Table 3 shows the hyperparameter settings of each algorithm.

Table 3. Hyperparameter settings of each machine learning algorithm.

Algorithm	Hyperparameter Setting
k-Nearest Neighbors	Euclidean distance, n_neighbors = 3
Random Forest	number_of_trees = 100, max_depth = 3
Decision Tree	criterion = gini, min samples split = 2

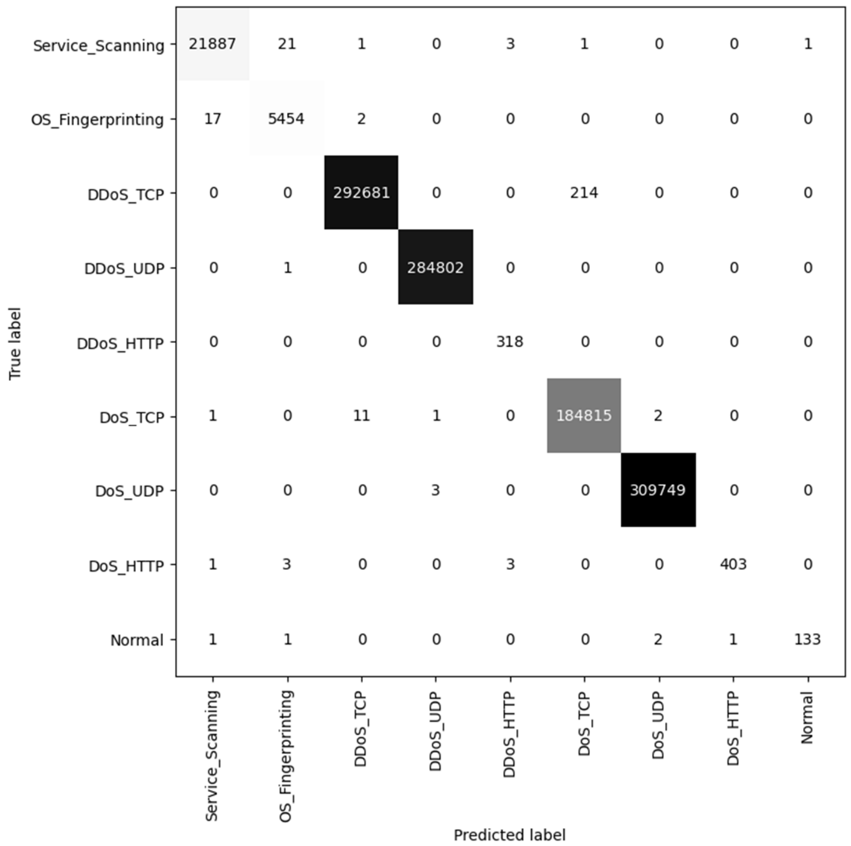
The classification results of the three classifiers are listed in Table 4. For k-nearest neighbors, the training process lasted more than 600 s and was terminated manually. The performance of kNN is poor, so it is excluded from the following discussion. Compared with random forest, decision tree presents better results and less running time. The classification results for each class using the decision tree are listed in Table 5. In addition, the confusion matrix for the decision tree is shown in Figure 1. It can be concluded that for classes whose number of samples is large, such as DDoS/DoS TCP and DDoS/DoS UDP, the classification results are nearly perfect. For minority classes, such as OS fingerprinting and normal, their classification results are relatively poor. This phenomenon is due to the data imbalance problem, but the solution of this problem is beyond the scope of this paper. For imbalanced datasets, the macro average metrics are more important than micro or weighted metrics [22], because macro metrics will not be influenced by the number of samples in each class. Thus, we will focus on macro metrics in the following experiments and discussion.

Table 4. Classification results for three classifiers using weighted metrics.

Classifier	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Running Time (s)
kNN	-	-	-	-	> 600
RF	94.13	94.43	93.83	94.43	164.4
DT	99.97	99.97	99.97	99.97	27.7

Table 5. Classification results of each class using a decision tree.

	Precision (%)	Recall (%)	F1-Score (%)	Support
Service Scanning	99.91	99.88	99.89	21914
OS Fingerprinting	99.53	99.65	99.59	5473
DDoS TCP	100.00	99.93	99.96	292895
DDoS UDP	100.00	100.00	100.00	284803
DDoS HTTP	98.15	100.00	99.07	318
DoS TCP	99.88	99.99	99.94	184830
DoS UDP	100.00	100.00	100.00	309752
DoS HTTP	99.75	98.29	99.02	410
Normal	99.25	96.38	97.79	138
Accuracy			99.97	1100533
Macro avg	99.61	99.35	99.47	1100533
Weighted avg	99.97	99.97	99.97	1100533



	Service_Scanning	OS_Fingerprinting	DDoS_TCP	DDoS_UDP	DDoS_HTTP	DoS_TCP	DoS_UDP	DoS_HTTP	Normal
Service_Scanning	21887	21	1	0	3	1	0	0	1
OS_Fingerprinting	17	5454	2	0	0	0	0	0	0
DDoS_TCP	0	0	292681	0	0	214	0	0	0
DDoS_UDP	0	1	0	284802	0	0	0	0	0
DDoS_HTTP	0	0	0	0	318	0	0	0	0
DoS_TCP	1	0	11	1	0	184815	2	0	0
DoS_UDP	0	0	0	3	0	0	309749	0	0
DoS_HTTP	1	3	0	0	3	0	0	403	0
Normal	1	1	0	0	0	0	2	1	133

Figure 1. Confusion matrix of classification results for the decision tree.

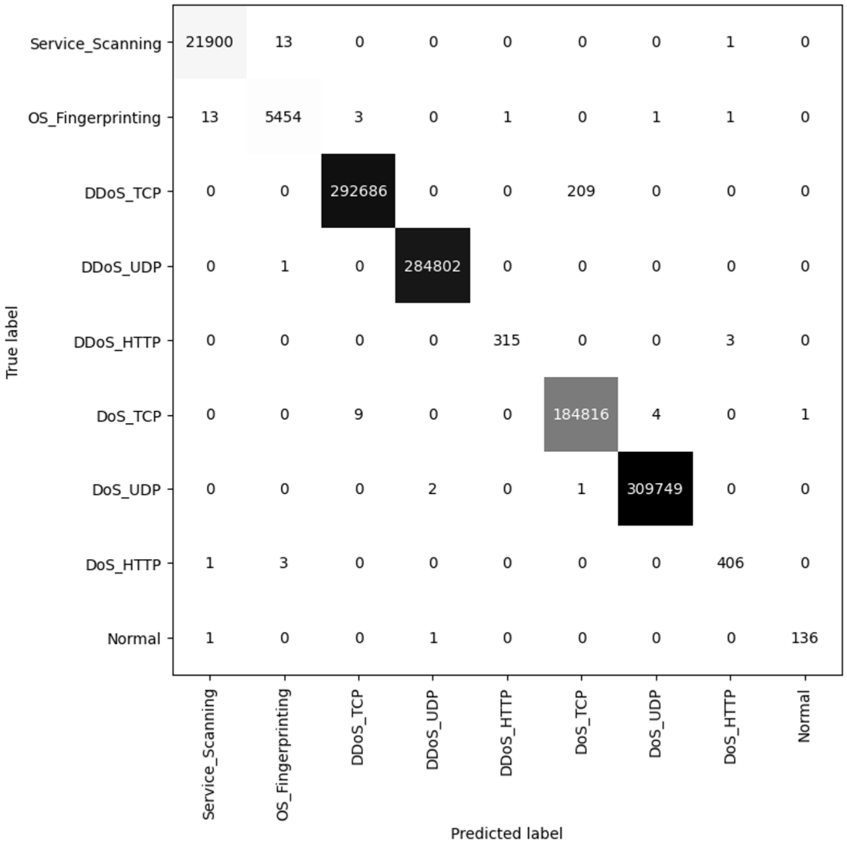
4.4. Experimental Results with Feature Selection

Unlike the previous experiments, this set of experiments included genetic algorithm-based feature selection, and the experiments were performed on the Bot-IoT dataset. By limiting the maximum number of features based on the feature selection process of the genetic algorithm and using the decision tree as the target classifier, the optimal feature set is selected through stratified cross-validation. Three experiments were performed with 4, 6, 8, and 10 as the maximum number of features in the feature set. For the genetic algorithm, the initial population size is set to 10, and the generations are set to 20. The figure shows the process of convergence of the feature selection method.

From Table 6, it can be obtained that the classification results with six selected features are better than those of nine features and three features. Although the accuracy of these results is higher than 99%, the macro metrics for three features are relatively unsatisfactory. Compared with the results of nine features, the results of the six-feature configuration present a slight improvement. However, using six features requires less computational resources than using nine features. Therefore, it is concluded from the above experiments that six features make the best trade-off between classification performance and efficiency. The confusion matrix for classification using six selected features is shown in Figure 2.

Table 6. Selected features and the corresponding classification results when the set max number of features equals 3, 6, and 9.

Max # Features	**3**	**6**	**9**
Selected Features	rate, TnBPDstIP, TnP_PerProto	bytes, rate, state, TnBPDstIP, TnP_PerProto, AR_P_Proto_P_SrcIP	pkts, bytes, rate, proto, state, TnBPDstIP, TnP_PerProto, AR_P_Proto_P_SrcIP, AR_P_Proto_P_Sport
macro precision (%)	95.73	99.69	99.57
macro recall (%)	93.32	99.57	99.36
macro f1-score (%)	94.22	99.63	99.46
accuracy (%)	99.59	99.98	99.97



True label \ Predicted label	Service_Scanning	OS_Fingerprinting	DDoS_TCP	DDoS_UDP	DDoS_HTTP	DoS_TCP	DoS_UDP	DoS_HTTP	Normal
Service_Scanning	21900	13	0	0	0	0	0	1	0
OS_Fingerprinting	13	5454	3	0	1	0	1	1	0
DDoS_TCP	0	0	292686	0	0	209	0	0	0
DDoS_UDP	0	1	0	284802	0	0	0	0	0
DDoS_HTTP	0	0	0	0	315	0	0	3	0
DoS_TCP	0	0	9	0	0	184816	4	0	1
DoS_UDP	0	0	0	2	0	1	309749	0	0
DoS_HTTP	1	3	0	0	0	0	0	406	0
Normal	1	0	0	1	0	0	0	0	136

Figure 2. Confusion matrix with six selected features using a decision tree.

4.5. Comparison with Previous Studies

Four state-of-the-art methods are used to perform experiments on the Bot-IoT dataset. These methods include those of Tama et al. [23], Zhao et al. [24], Kannari [25], and Hailm et al. [16]. Table 7 shows the comparison with other existing work using the Bot-IoT dataset. Compared with their results, the methods in this work perform best in terms of accuracy score. The method used by Halim et al. [16] is also based on genetic algorithms for feature selection; however, their definition of chromosome and fitness function are different from ours, and they did not provide a list of selected features.

Table 7. Comparison with previous studies on the Bot-IoT dataset.

Method	Year	Method	Accuracy (%)
Tama et al. [23]	2019	Ensemble learning	92.66
Zhao et al. [24]	2021	Representativeness-based instance selection	94.25
Kannari et al. [25]	2021	Sparse autoencoder	98.10
Halim et al. [16]	2021	Genetic algorithm and XGBoost, SVM	98.90
This work	2022	Genetic algorithm and decision tree	99.98

5. Conclusion and Future Scope

In the Internet of Things network environment, IDS is a powerful tool for defending against botnet attacks. The IDS monitors and captures the network traffic passively and then uses a machine learning model for benign and malicious traffic classification. However, due to the limited nature of computational resources in IoT devices, there is a demand for developing a fast and lightweight IDS with a high detection rate. Feature selection is a method that decreases the number of features used in classifiers while maintaining satisfactory performance compared with using full features. This work proposed a feature selection method based on a genetic algorithm. Guided by an objective classifier, the genetic algorithm searches for the optimal feature subset in a metaheuristic way. The experiments were performed on the botnet attack detection dataset Bot-IoT, and the proposed method selects 3–9 features from the data. From the comparison of performance for different numbers of features, the results for six features achieve a trade-off between training time and detection rate. The accuracy for the optimal feature subset is 99.87%; compared with the previous method on the Bot-IoT dataset, the feature subset is smaller, and the performance is better.

There are also some limitations of this work. For example, the variation in accuracy may be due to various other factors, the difference in the feature selection process and classification model used would influence the final classification performance. However, these details are very hard to analyze since too many unknown values and factors could influence the results in each method. This kind of analysis may be beyond the scope of this work.

There are many potential future research directions for this work. The genetic algorithm is the fundamental technique in this work. In the future, more metaheuristic methods, such as swarm intelligence, could be applied for feature selection. On the other hand, class imbalance problems exist in many intrusion detection datasets, and handling data imbalance is a challenging task in practical applications. Integrating feature selection techniques with imbalanced learning is a future research direction.

Author Contributions: Conceptualization, X.L., Y.D.; methodology, X.L.; software, X.L.; writing, X.L., Y.D.; supervision, Y.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Research Funds of People's Public Security University of China under Grant No. 2021JKF105, and the Open Research Fund of the Public Security Behavioral Science Laboratory of People's Public Security University of China under Grant No. 2020SYS20.

Data Availability Statement: The dataset used is located at https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 5 October 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Internet of Things (IoT) Connected Devices Installed Base Worldwide from 2015 to 2025. 2016. Available online: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ (accessed on 10 October 2022).
2. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84.

3. Kolandaisamy, R.; Noor, R.M.; Kolandaisamy, I.; Ahmedy, I.; Kiah, M.L.M.; Tamil, M.E.M.; Nandy, T. A stream position performance analysis model based on DDoS attack detection for cluster-based routing in VANET. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 6599–6612.
4. Otoum, Y.; Nayak, A. As-ids: Anomaly and signature based ids for the internet of things. *J. Netw. Syst. Manag.* **2021**, *29*, 1–26.
5. Boero, L.; Cello, M.; Marchese, M.; Mariconti, E.; Naqash, T.; Zappatore, S. Statistical fingerprint-based intrusion detection system (SF-IDS). *Int. J. Commun. Syst.* **2017**, *30*, e3225.
6. Sun, J.; Sun, K.; Shenefiel, C. Automated IoT Device Fingerprinting Through Encrypted Stream Classification; In *Security and Privacy in Communication Networks*; Chen, S., Choo, KK., Fu, X., Lou, W., Mohaisen, A.., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 147–167.
7. Nassif, A.B.; Talib, M.A.; Nasir, Q.; Dakalbab, F.M. Machine learning for anomaly detection: A systematic review. *Ieee Access* **2021**, *9*, 78658–78700.
8. Zhang, C.; Jia, D.; Wang, L.; Wang, W.; Liu, F.; Yang, A. Comparative research on network intrusion detection methods based on machine learning. *Comput. Secur.* **2022**, *121*, 102861.
9. Xue, B.; Zhang, M.; Browne, W.N.; Yao, X. A survey on evolutionary computation approaches to feature selection. *IEEE Trans. Evol. Comput.* **2015**, *20*, 606–626.
10. Jović, A.; Brkić, K.; Bogunović, N. A review of feature selection methods with applications. In Proceedings of the 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) 2015, Ieee, Opatija, Croatia, 25–29 May 2015.
11. Huang, S.H. Supervised feature selection: A tutorial. *Artif. Intell. Res*. **2015**, *4*, 22–37.
12. Stein, G.; Chen, B.; Wu, A.S.; Hua, K.A. Decision tree classifier for network intrusion detection with GA-based feature selection. In Proceedings of the 43rd Annual Southeast Regional Conference-Volume 2, Kennesaw, GA, USA, 18–20 March 2005.
13. Kannan, A.; Maguire, G.Q., Jr.; Sharma, A.; Schoo, P. Genetic algorithm based feature selection algorithm for effective intrusion detection in cloud networks. In Proceedings of the 2012 IEEE 12th International Conference on Data Mining Workshops, 2012, IEEE, Brussels, Belgium, 10 December 2012.
14. Raman, M.G.; Somu, N.; Kirthivasan, K.; Liscano, R.; Sriram, V.S. An efficient intrusion detection system based on hypergraph-Genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowl.-Based Syst.* **2017**, *134*, 1–12.
15. Mehanović, D.; Kečo, D.; Kevrić, J.; Jukić, S.; Miljković, A.; Mašetić, Z. Feature selection using cloud-based parallel genetic algorithm for intrusion detection data classification. *Neural Comput. Appl.* **2021**, *33*, 11861–11873.
16. Halim, Z.; Yousaf, M.N.; Waqas, M.; Sulaiman, M.; Abbas, G.; Hussain, M.; Ahmad, I.; Hanif, M. An effective genetic algorithm-based feature selection method for intrusion detection systems. *Comput. Secur.* **2021**, *110*, 102448.
17. Mojtahedi, A.; Sorouri, F.; Souha, A.N.; Molazadeh, A.; Mehr, S.S. Feature Selection-based Intrusion Detection System Using Genetic Whale Optimization Algorithm and Sample-based Classification. *arXiv preprint* **2022**, arXiv:2201.00584.
18. Mitchell, M. *An Introduction to Genetic Algorithms*; MIT press: Cambridge, MA, USA, 1998.
19. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
20. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796.
21. Koroniotis, N.; Moustafa, N.; Sitnikova, E. A new network forensic framework based on deep learning for Internet of Things networks: A particle deep framework. *Future Gener. Comput. Syst.* **2020**, *110*, 91–106.
22. Bagui, S.; Li, K. Resampling imbalanced data for network intrusion detection datasets. *J. Big Data* **2021**, *8*, 6.
23. Tama, B.A.; Comuzzi, M.; Rhee, K.-H. TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access* **2019**, *7*, 94497–94507.
24. Zhao, F.; Xin, Y.; Zhang, K.; Niu, X. Representativeness-based instance selection for intrusion detection. *Secur. Commun. Netw.* **2021**, *2021*, 6638134.
25. Kannari, P.R.; Shariff, N.C.; Biradar, R.L. Network intrusion detection using sparse autoencoder with swish-PReLU activation model. *J. Ambient. Intell. Humaniz. Comput.* **2021**. https://doi.org/10.1007/s12652-021-03077-0

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.