Journal of Computational Science 64 (2022) 101833

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Improvement of attack detection performance on the internet of things with PSO-search and random forest

Kurniabudi [a], Deris Stiawan [b,*], Darmawijoyo [c], Mohd Yazid Bin Idris [d], Sarjon Defit [e], Yaya Sudarya Triana [f], Rahmat Budiarto [g]

[a] Faculty of Computer Science, Universitas Dinamika Bangsa, Indonesia
[b] Faculty of Computer Science, Universitas Sriwijaya, Indonesia
[c] Faculty of Mathematics and Natural Sciences, Universitas Sriwijaya, Indonesia
[d] School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, Malaysia
[e] Faculty of Computer Science, Universitas Putra Indonesia YPTK Padang, Indonesia
[f] Department of Information System, Universitas Mercu Buana, Indonesia
[g] College of Computer Science & IT, Albaha University, Saudi Arabia

ARTICLE INFO

Keywords:
IoT network
Complex network
Feature selection
Lightweight IDS
Machine learning

ABSTRACT

The presence of the internet of things allows various smart devices to be connected and interact with each other. Although IoT provides benefits in daily activities, however, with the presence of new technologies, IoT is vulnerable to new types of attacks. The massive IoT traffic results in a large number of traffic features and constructs complex network that makes intrusion detection systems (IDSs) require large resources to identify the type of attacks. On the other hand, most of the intrusion detection techniques are not feasible for IoT networks because they require more computing resources for attack detection, while IoT devices have limited computing resources and storage capacity. Thus, a lightweight IDS that has ability to identify new types of attacks is required. This research proposes a hybrid of Panigrahi and PSO-Search approaches to reduce the complexity of the network by eliminating the number of irrelevant features effectively and efficiently and combine with Random Forest optimization method to improve detection performance. The proposed IDS is validated with training and testing data, using hold-out, Stratified k-fold cross-validation, and percentage split test mode on CICIDS-2017 dataset MachineLearningCSV version. The dataset is chosen, as it represents real IoT network traffic data. Experimental results show that the performance improvement of the proposed hybrid IDS is very encouraging. The accuracy rate reaches 99.9 %, with an average Recall value of 1.000.

1. Introduction

The evolution of various sensors, devices, computing strategies and communication technologies enabling the integration of smart devices with daily activities has driven the emergence of the Internet of things (IoT). IoT enables smart devices to interact and communicate with each other [1]. Guo and Heidemann [2] reveal the presence of vulnerabilities comes from IoT network on the Internet. Ammar et al., [3] state that security is one of the issues that need to be considered in the implementation of IoT.

Various IoT attack detection methods and techniques have been developed to detect various types of attacks on IoT networks, including: Anomaly based IDS, Network based IDS, Host based IDS, and Distributed based IDS [4]. With the presence of security issues in IoT, IDS becomes an important security tool for IoT networks [5]. McDermott et al. [6] propose Deep Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN) to detect BotNet, however, the proposed model requires increasing resources and processing time. Meanwhile, Yang et al. [7] propose Levenberg-Marquardt Back Propagation (BLM-BP) neural network to detect attacks on IoT network. However, this study only detects Denial of Service (DoS), R2L, U2L, and Probing attacks.

Choudhary and Kesswani [5] have mentioned with the presence of the IoT, more and more issues come one after another, including attacks on IoT networks. Although a lot of researches have been carried out to overcome the attacks, however, yet many types of attacks have not been

* Corresponding author.
E-mail address: deris@unsri.ac.id (D. Stiawan).

https://doi.org/10.1016/j.jocs.2022.101833
Received 27 April 2022; Received in revised form 27 June 2022; Accepted 10 August 2022
Available online 28 August 2022

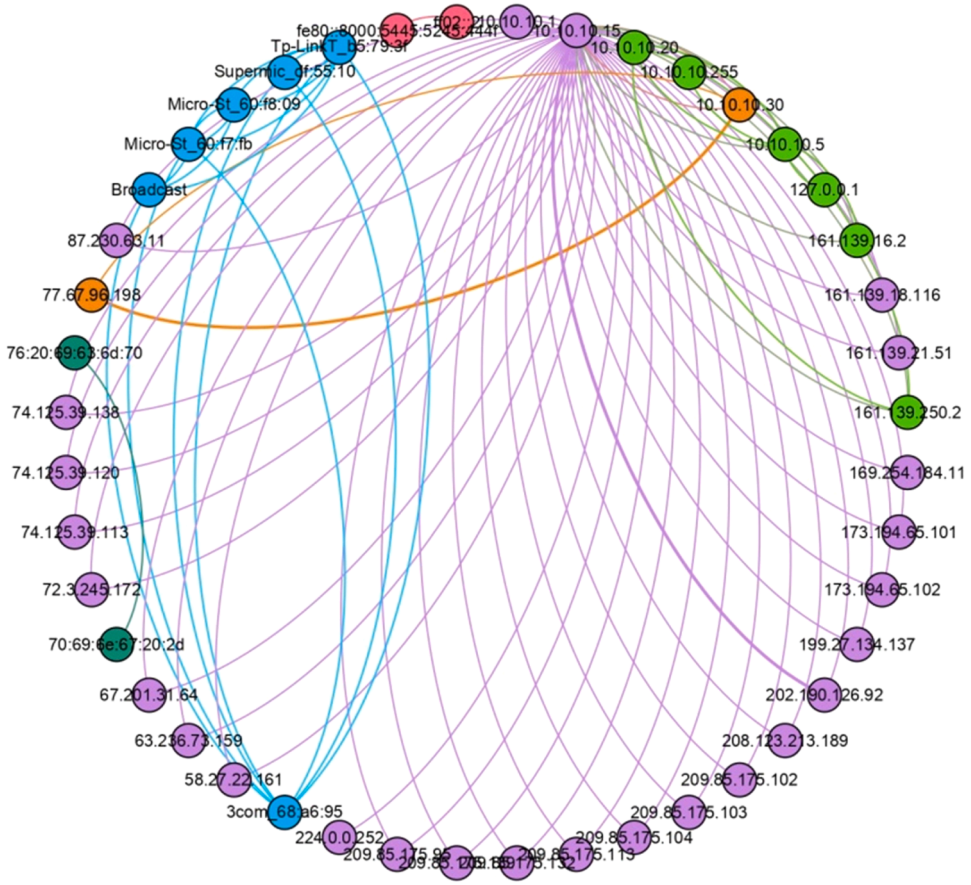

Fig. 1. Illustration of complex network of the node-to-node traffics during attack in an IoT testbed network [11].

investigated. IDSs with machine learning take more time to analyze complex network that contains a lot of noise resulting in a very large volume of IoT traffic. At the same time, IoT devices have limited computing resources. Therefore, a lightweight IDS that consumes less computing resources is demanded [8]. To overcome this issue, Jan et al. [9] proposed lightweight IDS using Support Vector Machine (SVM)-based classifier. The proposed method can perform satisfactorily in detection of attacks. Meanwhile, Fenanir et al. [10] compare several machine learning algorithms to produce lightweight IDS. The experimental results show that the decision tree (DT) algorithm has superior performance compared to others.

The massive IoT traffic results in a large number of traffic features and constructs complex network that makes intrusion detection systems (IDSs) require large resources to identify the type of attacks. Stiawan et al. [11] investigate a small scale IoT network testbed consists of multiple hardware including DHT22 sensor, MQ2 sensor, soil moisture sensor, water level sensor, two Zigbee type sensors and WeMos D1 microcontroller equipped with ESP8266 Wi-Fi module. Two middleware modules using Raspberry Pi microcontroller are used for communicating the Zigbee and Wi-Fi types of equipment. In addition, the testbed utilizes supporting software such as MySQL database, DoS tools Hping3, Apache Web Server and Snort as IDS. Fig. 1 illustrates a complex network of node-to-node traffics during DoS attack in the IoT testbed network.

Approaches have been proposed to solve complex networks problems, including the use of complex networks as artificial intelligence mechanisms [12]; the use of parallel metaheuristic framework based on moth-flame optimization (MFO), clustering and pre-processed datasets [13]; combining particle swarm optimization (PSO) with Monte Carlo simulation (MCS) [14]; and partitioning the network into some communities in such a way that there exist many connections in the communities and few connections between them, because community structure is an important feature in complex networks which has great significant for organization of networks. Mirsaleh et al. [15] propose Michigan memetic algorithm to perform community detection.

Although IDS for IoT researches have been carried out and show good performance, there are yet several shortcomings to be considered, i.e.: (1) testing is carried out on limited number of sample data; (2) limited types of traffic attacks, for example, some researchers only identified normal traffic and DDoS attacks, while IoT network is also vulnerable to other types of attacks; and (3) Even the accuracy value shows a high rate, however, there is still room for improvement. Therefore, this study proposes a hybrid method to produce lightweight IDS with high detection rate capability.

In this study, the network complexity is reduced by producing optimal features of traffic data. Thus, during the feature selection phase, the Panigrahi's approach [16] is combined with PSO-Search. The Correlation-based Feature Selection (CFS) was implemented and combined with the PSO-Search method. This performance improvement is expected to produce the most optimal features for detecting attacks on IoT.

For experimental purpose, the CICIDS-2017 dataset is selected. This dataset has been used by researchers because it represents modern network traffic and the internet of things. Random Forest (RF) was chosen as the classification algorithm. The RF algorithm has been applied on IDS research works and is capable of detecting the desired attack, as discussed in [17], [18] and [19]. In the present work, the performance of RF was improved first.

Validation of the proposed method was also carried out. Validation of the method or model is very important to test the reliability of the model. The validation process uses several test modes such as hold-out, Stratified k-fold cross-validation, and percentage split.

The major contributions of this research include:

Table 1
Summary of IDS researches on the IoT networks.

Reference	Dataset	Type of attack	Performance measurement
[7]	KDD Cup 99	DOS, R2L, U2L, and Probing	Detection Rate(DR), False Alarm Rate(FAR)
[8]	IoT Bot and KDD Cup 99	DoS and DDoS	DR, Accuracy(ACC) FAR, CPU Time
[9]	CICIDS2017	DDoS	DR, ACC, True Positive Rate (TPR), False Positive Rate (FPR), False Detection Rate(FDR), CPU Time
[20]	NSL-KDD	DoS, DDoS, Reconnaissance, and Key-logging	TPR, FPR, F-Measure
[21]	Real Time Traffic Captured	DoS Attempt, Overflow Attempt. SSH Brute Force Login, Suspicious DNS query, Cache Poisoning attempt, Malware infection, other attack	F1-Score, Recall, Precision
This Study	CICIDS-2017	DoS/DDoS, PortScan, Bot, Web Attack–Brute Force, Web Attack–XSS, Web Attack–SQL Injection, Infiltration, DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, Heartbleed, FTP- Patator, and SSH-Patator.	ACC, Recall (TPR), Sensitivity, F1 Score (F-Measure), AUC, CPU Time

1) The combination of Panigrahi and PSO-Search approaches was applied to reduce the traffic features;
2) Improvement of PSO-Search performance in selecting important and relevant features;
3) Improved RF performance, in detecting normal and attacks traffics on IoT network;
4) Testing of the proposed IDS is carried out on more complex dataset with a more diverse number of attacks; and
5) The validation of the proposed detection system in the form of insights for researchers, especially in the implementation of machine learning on IDS.

2. Relevant works

The presence of IoT and the security issues that come along with it encourages researchers to develop techniques and methods of detection systems on IoT network. Researchers in [7] propose LM-BP neural network to detect malicious attacks. The experimental results show that the proposed method has a high detection rate and low false alarms in detecting DOS, R2L, U2L, and Probing attacks. Nimbalkar and Kshirsagar [8] use GRip as a classifier. The experimental results show that the proposed method has better performance.

Jan et al., [9] propose a DoS attack detection system on IoT by using SVM. The proposed method is able to increase the accuracy of attack classification and decreases detection time. While Khraisat et al. [20] propose a Hybrid Intrusion Detection System (HIDS) by combining the C5 classification algorithm and One Class SVM. The experimental results show that the proposed method has a high detection rate and low false alarms compared to previous studies. Furthermore, Balakrishnan et al., [21] use a Deep Belief Network (DBN) approach to detect attacks on IoT. Table 1 presents a summary of IDS researches on IoT network.

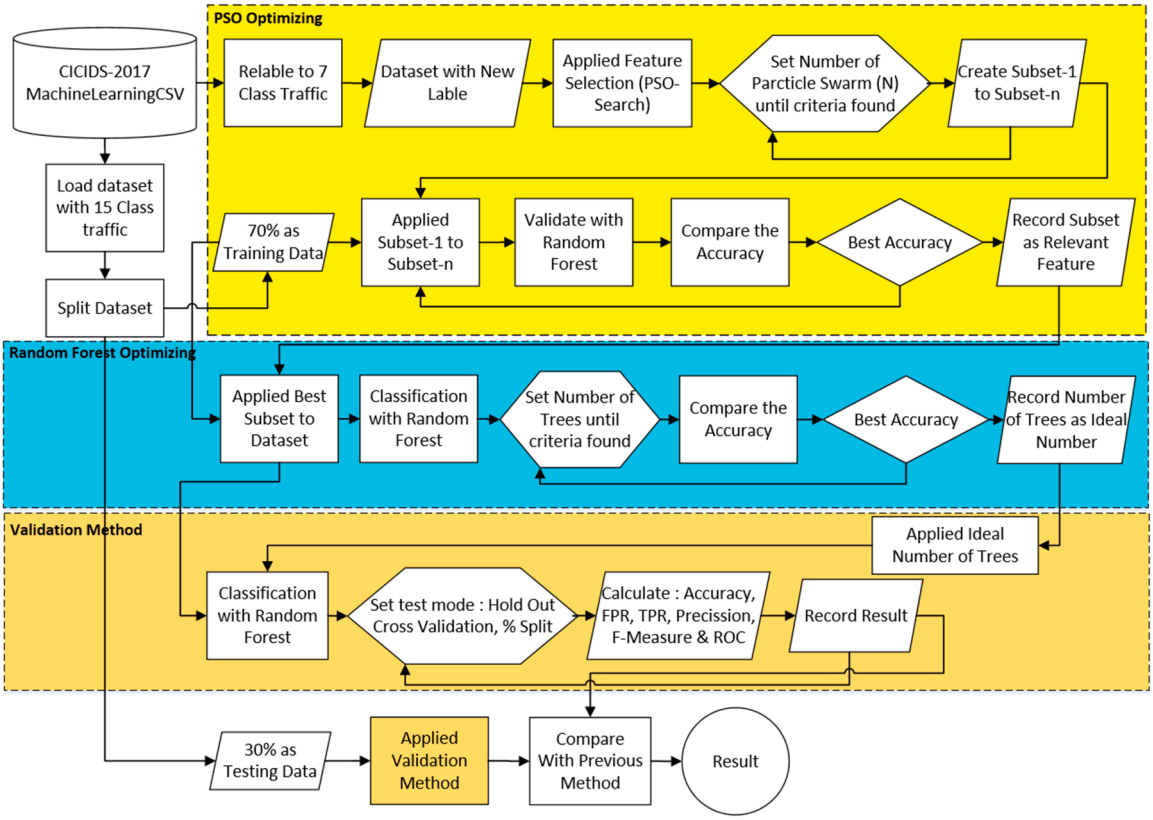

Fig. 2. Research framework.

Table 2
Profile of 30 % CICIDS-2017 dataset with 7 class labels.

Class label	# of record	% of total data
Normal	681.995	80.308
DoS/DDoS	114.241	13.452
PortScan	47.487	5.592
Bot	574	0.068
Web Attack	665	0.078
Infiltration	8	0.001
Brute Force	4.253	0.501
Total	**849.223**	**100**

3. Methodology

3.1. The proposed method

This study proposes an optimal detection system on IoT networks, by combining the CFS-PSO-Search feature selection technique and the RF classification algorithm. Fig. 2 shows the proposed method.

As shown in Fig. 2, there are three main stages of improving the performance of the detection system, namely: 1) Improved CFS-PSO-Search performance on feature selection techniques to produce truly relevant features, 2) Improved RF performance to improve detection performance traffic, and 3) Validation of detection system so that a reliable method is produced. The three main stages can be described in detail as follows.

4. Improved PSO-Search performance

a) PSO-Search performance improvement is achieved by determining the number of particle swarms. The dataset used is the CICIDS-2017 dataset which has been given 7 (seven) new labels.
b) Different number of particle swarms (N) will produce subsets with different feature groups. The values of N used are 2, 5, 7, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 300, 500, and 1000.
c) Furthermore, these subsets are applied to the CICIDS-2017 dataset with old/original labels (15 labels). Then, are validated with RF algorithm. The validation results of each subset are recorded and compared.
d) The subset with the highest accuracy value will be considered as the best subset containing the most relevant set of features, and will be used in the Random Forest performance improvement process.

1) Random Forest performance improvement

a) Random Forest performance improvement is executed by determining the most ideal number of Tree parameters. This process is repeated with a different number of Tree parameters. The numbers of trees used are: 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, and 150.
b) This process is applied to the CICIDS-2017 dataset with 15 labels.
c) The results of the RF classification with different Tree values are then recorded and compared.
d) The number of trees that produce the highest classification accuracy will be considered as the ideal value and will be used in the validation process.

2) Method/Model Validation

The purpose of method/model validation is to test its reliability. After obtaining the ideal number of trees for the RF learning process, several classification tests were carried out by applying hold-out, Stratified 5-fold cross-validation, Stratified 10-fold cross-validation, and percentage split test modes. This process is carried out on training data as well as on testing data. Furthermore, in the experiment, comparisons were also made with state-of-the-art methods/models.

Table 3
Profile of 70 % of the CICIDS-2017 dataset.

No.	Class label	# of record	% of total data
1	Benign	1,591,102	80.29704
2	DDoS	89,600	4.52178
3	PortScan	111,443	5.62412
4	Bot	1392	0.07025
5	Web Attack–Brute Force	1052	0.05309
6	Web Attack–XSS	450	0.02271
7	Web Attack–Sql Injection	13	0.00066
8	Infiltration	28	0.00141
9	DoS slowloris	4057	0.20474
10	DoS Slowhttptest	3894	0.19652
11	DoS Hulk	161,814	8.16616
12	DoS GoldenEye	7087	0.35765
13	Heartbleed	6	0.00030
14	FTP- Patator	5516	0.27837
15	SSH-Patator	4066	0.20520
	Total	**1,981,520**	**100**

Table 4
Profile of 30 % of the CICIDS-2017 dataset.

No.	Class label	# of record	% of total data
1	Benign	681,995	80.30812
2	DdoS	38,427	1.35749
3	PortScan	47,487	1.67755
4	Bot	574	0.02028
5	Web Attack–Brute Force	455	0.01607
6	Web Attack–XSS	202	0.00714
7	Web Attack–Sql Injection	8	0.00028
8	Infiltration	8	0.00028
9	DoS slowloris	1739	0.06143
10	DoS Slowhttptest	1605	0.05670
11	DoS Hulk	69,259	2.44667
12	DoS GoldenEye	3206	0.11326
13	Heartbleed	5	0.00018
14	FTP- Patator	2422	0.08556
15	SSH-Patator	1831	0.06468
	Total	**849,223**	**100**

4.1. Dataset for the experiments

To meet the real network traffic data in this study, the CICIDS-2017 dataset was considered. This dataset was chosen because of the large number of traffic features and the number of different attack classes [22]. In addition, this dataset is the most up-to-date data, and contains common attack types, and can be used to evaluate IDS at a large scale [23]. According to Sharafaldin et al. [24], 11 datasets available since 1998 are expired now and are not reliable for use.

Details of the 30 % of the CICIDS-2017 dataset with 7 labels consisting of 1 normal class and 6 attack classes are presented in Table 2. Meanwhile, for subsets of testing, Random Forest performance improvement, and validation, the dataset used is the Machine-LearningCSV version of the CICIDS-2017 dataset. This dataset has 15 traffic class labels consisting of 1 normal traffic class and 14 attack traffic classes. This dataset is separated into training data and testing

data with a proportion of 70 % for training data and 30 % for testing data. Distribution details for training and testing data are presented in Tables 3 and 4.

Each portion of the training and testing data is divided by stratified sampling, with the aim that each portion has data containing the same traffic class. Referring to Tables 3 and 4, it can be seen that the dataset has an imbalance distribution in each class and there is a majority class, namely Benign traffic and a minority class, i.e.: HeartBleed attack traffic class. The data in the table shows that in real network traffic the amount of attack traffic is very small when compared to normal traffic. The ideal IDS must be able to detect the attack traffic even in a very small portion of data in real network traffic.

4.2. Feature selection with CFS and PSO-search

CFS is an evaluator to evaluate attributes/features. The CFS evaluates the relationship between features and related classes, and between features and features in subsets known as Merit [25]. The merit is calculated using Eq. 1.

$$Ms = R_{FC} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \quad (1)$$

Where,

Ms: Merit
k: Number of features
$\overline{r_{cf}}$: The average value between features and related classes
$\overline{r_{ff}}$= The average value between one feature and another

PSO-Search uses the PSO Algorithm (shown in Algorithm-1) to explore attribute-space or feature-space [26]. PSO is a powerful algorithm, easy to implement, and computationally efficient. PSO is an evolutionary algorithm that searches using a population (swarm) of individuals (particles) that is updated from iteration to iteration [27]. This study uses *WEKA* PSO-Search module developed by Valero [25].

.

1: **for** each particle i **do**
2: initialize position x_i at random in the search space
3: **end for**
4: **while** stop criteria not met do
5: for each particle i do
6: set personal best $\hat{x}_i$ as best position found so far by the particle
7: set global best $\hat{g}$ as best position found so far by the whole swarm
8: **end** for
9: **for** each particle i **do**
10: update position using a randomized convex combination
11: mutate xi
12: **end for**
13**: end while**

4.3. Random forest classification algorithm

RF is one of the classification algorithms included in the decision tree. RF is commonly used to classify large data. A random forest itself forms a decision tree. The more trees used will greatly affect the classification. RF is called the ensemble method, where this method creates forest consisting of a collection of trees [24]. Each tree casts a vote which indicates the tree's decision about the object class. Forest chooses the class with the most votes (major votes) for the object [28].

The selection of the RF algorithm in this study is based on its ability to process large data and has a low misclassification rate compared to other classification algorithms [17]. The selection of this algorithm is also based on the results of the initial experiment which shows the ability of RF classification is better than other classification algorithms.

4.4. Experiment configuration

For the purposes of feature selection and learning experiments, a notebook with an Intel Core i7 processor, 2.70 GHz and 8 GB RAM with Windows 10 Operating System was used. *WEKA* 3.8.5 software with a heap size configuration of 3072 MB was used as an analysis tool.

To maintain the reliability of the testing, in the experiment several test modes were applied, i.e.:

a) Training set/Holdout: Testing classification performance using all input data
b) Stratified k-fold Cross-Validation: 10-fold and 5-fold cross validations.
c) Percentage Split: The experiment uses split 10 to split 60.

5. Result and analysis

This section describes the performance improvements and the overall experimental results. The presentation includes, improved feature selection performance, improved Random Forest classification performance, detection system performance validation, and detection system performance comparison.

5.1. Optimizing PSO-search performance

PSO-Search is a search method on PSO-based feature selection techniques. The purpose of improving the performance of PSO-Search is that this method is able to produce features that are truly relevant and important to identify IoT traffic. In this study, the increase in PSO-Search performance was achieved by determining the number of particle swarms. This method is performed following the research work carried out by Tama and Rhee [29]. However, in this present study, we use a larger number of particle swarm options and use Random Forest as a classification algorithm to validate the selected features. In addition, the PSO-Search performance improvement was applied to CICIDS-2017 as IoT dataset. The results of the performance improvement will get the most ideal number of particle swarms which have an impact on selecting the most relevant features so as to improve the performance of the detection system.

In the early stages of optimization, feature selection is carried out by implementing PSO-Search on the CICIDS-2017 dataset which has been re-labeled with 7 new class labels. Pseudo code for feature selection using PSO is presented as follows:

```
Input :
    N= particle swarm (1....1000)
Output :
    best_params=number of selected features
Pseudecode
def PSO(obj, p_num, N, scale, eps=1e-5, early_stopping=100,
max_iter=1000)do
    np.random.seed(random_state)
    params_num = 2 * N + 1
    p = np.random.normal(scale=scale, size=(p_num,
    params_num))
    v = np.random.normal(scale=scale, size=(p_num,
    params_num))
    t = np.linspace(0, 4 * np.pi, 101)
    error = obj(t, p)
    pbest = p.copy()
    gbest = p[error.argmin()]
    best_params = [gbest]
    e = [error.min()]
    while error.min() > eps and iter_num < max_iter and
    count < early_stopping do
        r1 = np.random.uniform(size=(params_num))
        r2 = np.random.uniform(size=(params_num))
        r1 = np.tile(r1, p_num).reshape(p_num, -1)
        r2 = np.tile(r2, p_num).reshape(p_num, -1)
        v = w * v + c1 * r1 * (pbest - p) + c2 * r2 * (gbest - p)
        w = w * r
        p = p + v
        error = obj(t, p)
        errorbest = obj(t, pbest)
        gbest = p[error.argmin()]
        min_idx = np.array([error, errorbest]).argmin(axis=0)
        for i, idx in enumerate(min_idx)do
            if idx == 0: pbest[i,:] = p[i,:].copy() do
            iter_num += 1
            if error.min() >= e[-1]:
            count += 1
            end if
            else:
            count = 0
            e.append(error.min())
            end if
            best_params.append(gbest)
            if verbose do
            if len(gbest) <= 7 do
            params ={gbest}\n\t\terror = {e[-1]:.4f}')
            end if
        end for
        if count == early_stopping do
        iter_num = iter_num – count

        best_params = best_params[:-early_stopping]
        e = e[:-early_stopping]
        end if
  return iter_num, np.array(best_params), np.array(e)
end while
end def
```

The number of data used is 849,223 data. This initial stage performs feature selection by changing the number of particle swarm parameters. The experiment was carried out 20 times, resulting in 20 subsets of each different particle swarm parameter. This experiment is stopped when the particle swarm values produce the same number and type of features. The experimental results in Table 5 show that the number of particle swarms affects the number of features and the type of features selected. It can be seen that with particle swarms of 1000, 1500, and 2000 the PSO selects 7 (seven) same features, i.e.: *Bwd Packet Length Std, Destination Port, Bwd Packet Length Min, Subflow Bwd Bytes, Init_Win_bytes_backward, min_seg_size_forward,* and *Active Min*.

To determine the most ideal number of particle swarms and which subset has the most relevant features, each subset is validated using the Random Forest algorithm. To validate this subset, 20 experiments were conducted, using 30% of the CICIDS-2017 dataset with 15 labels. The number of data used is 849,223 records. The experimental results show that the selected features in each subset greatly affect the accuracy rate. The highest accuracy value was achieved with particle swarm N = 7 and N = 10. Where particle swarm N = 7 produces 28 features with an accuracy rate of 99.988 %. Meanwhile, with particle swarm N = 10, 31 features were produced with an accuracy rate of 99.988%. So, particle swarm N = 7 and N = 10 produce the same accuracy with a different number of features. The rationale is there exists 17 features that are shared by subset 3 and subset 4, i.e.: *Destination Port, Fwd Packet Length Max, Fwd Packet Length Min, Bwd Packet Length Min, Bwd Packet Length Mean, Flow IAT Min, Bwd IAT Min, Fwd PSH Flags, FIN Flag Count, SYN Flag Count, ECE Flag Count, Down/Up Ratio, Fwd Header Length, Init_Win_bytes_forward,Init_Win_bytes_backward, min _seg_size_forward,* and *Active Max*.

Since subset 3 and subset 4 have the same accuracy value, with different number of features, in this case, of course, the subset with the least number of features is subset 3. Thus, because subset 3 is generated with particle swarm parameters N = 7, then the number of particle swarm is considered ideal.

The experimental results also show that the number of particle swarm (N) > 100 produces a feature subset with less number of features which also provides the lower accuracy rates. As seen in Fig. 3, accuracy rates for particle swarm N = 200 to N = 2000 produce lesser and lesser number of features and have lower accuracy rate. This low rate may result features that are not sufficient enough to identify all traffic classes exist in the testing dataset.

Furthermore, to see the impact of the number of features on the accuracy rate, the number of features is sorted from the least to the most. Fig. 4 presents the accuracy based on the number of features. It can be seen that the higher the number of features, the higher the accuracy value. This phenomenon is due to the more complete the relevant features produced so as to increase the accuracy value.

5.2. Optimizing PSO-search performance

In this study, RF was adopted as the proposed classification method to be used to identify traffic in the testing dataset. In the RF algorithm, there are two parameters that determine the classification performance, namely: the number of variables and the number of trees. In this study, RF performance improvement is achieved by determining the ideal number of trees, because the analyzed variables or features are the features that are considered the most relevant because they have gone through the feature selection process. For the purpose of improving performance, 11 experiments were carried out that relates to the setting of the number of trees used, i.e.: 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, and 150. Experiments on the detection system with Random Forest were carried out with *WEKA* tool. Meanwhile, the Random Forest pseudocode is presented as follows:

```
Input
    Dt = dataset labeled
    X = training data
    Y = testing data
Output
    Hrf = result of classification RF
Pseudocode
import cross_val_score
import RepeatedStratifiedKFold
import RandomForestClassifier
def get_dataset()do
    x, y = read_dataset(Dt)
    return X, y
end
def get_models()do
    models = dict()
    for i in range(len(Dt)do
    models[str(i)] = RandomForestClassifier(max_features=i)
    end for
    return models
end
def evaluate_model(model, X, y)do
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
    random_state=1)
    scores = cross_val_score(model, X, y, scoring='accuracy',
    cv=cv, n_jobs=-1)
    return scores
end
X, y = get_dataset()
models = get_models()
Hrf, names = list(), list()
for name, model in models.items()do
    scores = evaluate_model(model, X, y)
    Hrf.append(scores)
    names.append(name)
end for
```

The experimental results are presented in Table 6. The results show that the RF performance reaches a stable performance with the number of trees > 100 with an accuracy rate of 99.98%.

The experimental results show that the number of trees affects the value of the classification accuracy of RF, as shown in Fig. 5. In addition, the results also show that the number of trees also affects the testing time and processing time. Testing time is the time required to test the model, while processing time is the total time required to build the model until the model testing process ends. It is observed in Fig. 6, the larger the number of trees, the greater the testing time.

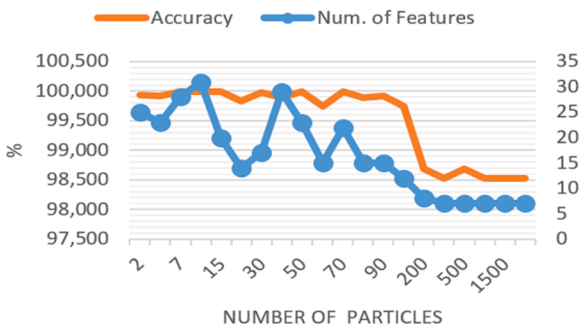

Fig. 3. Number of particle swarms on accuracy.

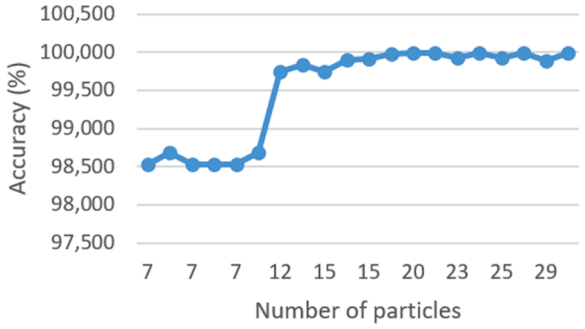

Fig. 4. Number of features on accuracy.

5.3. Validation of the proposed method

To maintain the reliability of the proposed method, 9 experiments were carried out for each training data and testing data with different validation modes, i.e.: Hold-out, 5-Fold, 10-Fold, Split-10, Split-20, Split-30, Split-40, Split-50, and Split-60. The results of accuracy on training data are presented in Table 7 while accuracy on testing data is presented in Table 8.

Table 5
PSO-Search feature selection results.

# of Particle (N)	Selected features	Accuracy (%)
2	5,8,9,10,11,14,16,18,19,20,28,36,38,39,41,44,45,48,51,55,68,69,73,76,77	99,928
5	1,4,8,13,17,18,19,23,27,28,31,36,38,40,42,44,45,51,55,65,69,73,76	99,926
7	1,4,8,14,15,18,19,20,23,25,31,32,37,38,41,43,44,46,50,51,55,62,66,67,69,72,76,77	99,988
10	2,3,5,8,9,13,14,15,19,20,23,31,32,36,39,40,42,43,44,50,51,55,63,65,66,67,68,69,70,72,73	99,988
15	1,5,7,8,9,13,18,19,39,43,50,54,62,65,66,67,69,70,73,76	99,986
20	8,18,19,27,31,42,51,63,65,67,69,70,72,76	99,835
30	1,2,8,13,18,19,20,23,27,42,43,55,66,67,69,73,77	99,983
40	1,7,8,13,14,15,16,18,19,20,36,40,41,44,48,50,51,52,55,65,66,67,68,69,70,72,73,76,77	99,890
50	1,8,9,13,15,18,19,27,31,38,41,43,48,51,54,63,66,67,69,70,72,73,77	99,986
60	1,8,13,19,20,41,43,52,65,66,67,69,70,73,76	99,749
70	1,7,8,13,18,19,20,23,25,39,42,43,47,51,52,65,66,67,69,70,73,77	99,986
80	1,8,13,18,19,27,40,41,48,66,67,69,72,73,76	99,894
90	1,8,13,19,20,37,41,51,64,65,66,67,69,72,73	99,913
100	8,19,41,43,50,52,65,66,67,69,70,73	99,742
200	1,8,19,41,65,67,69,73	98.683
300	1,8,19,65,67,69,73	98,530
500	8,13,19,41,67,69,73	98,683
1000	1,8,19,65,67,69,73	98,530
1500	1,8,19,65,67,69,73	98.530
2000	1,8,19,65,67,69,73	98.530

Table 6
Random Forest performance improvement results.

Number of trees	Accuracy	Time to test (Sec.)	Time process (Sec.)
50	99.98	12.09	329
60	99.98	15.37	392
70	99.98	18.70	462
80	99.98	20.61	523
90	99.98	22.30	598
100	99.98	28.39	691
110	99.98	29.16	734
120	99.98	29.77	771
130	99.98	31.03	839
140	99.98	35.25	931
150	99.98	38.66	1118

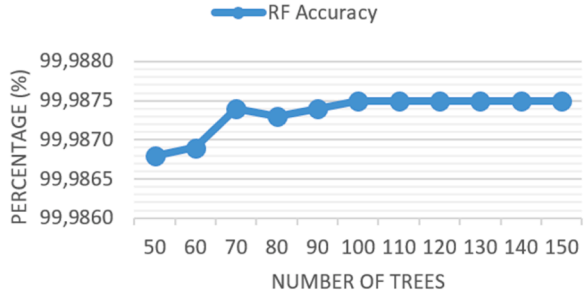

Fig. 5. Effect of the number of trees on accuracy rate.

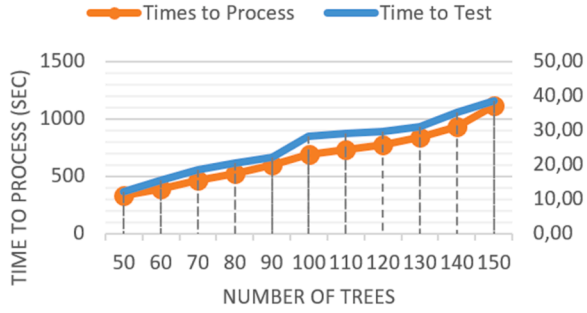

Fig. 6. The effect of the number of trees on the testing time and processing time.

The testing results show that the hold-out mode produces the highest accuracy with an accuracy rate of 99.98 % on training data and 99.99 % on testing data. These results happen because in the Hold-out mode all data is used in the learning process. Overall, experiments using the training data and the testing data show results that are not too different, where the average value of accuracy rate on the training data and on the testing data reaches 99.86% and 99.85%, respectively.

As shown in Tables 3 and 4, the datasets are very unbalanced. Therefore, to test the reliability of the proposed method in classifying the data, Stratified k-fold validation was used. Table 9 shows the validation results with Stratified k-fold. The test results show very good performance, and can also be seen from the calculations of the RMSE values of the proposed method. The accuracy of the classification method can also be seen from the small RMSE values. The testing results show that the proposed method has a small RMSE value.

5.4. Performance observation

Table 10 presents the performance of the detection system for selected features using the PSO-Search method and validated by the RF algorithm before improving performance. Based on the Recall and sensitivity value, it can be observed that with the features produced by PSO-Search without the performance improvement, it is able to properly detect every normal traffic and attacks, however not optimal. Several types of attacks that have not been detected optimally are the type of Web Attack XSS attack with a Recall value of 0.515 and Web Attack SQL with a Recall value of 0.625. Nevertheless, taking into account the values of Recall, Sensitivity, F1 Score, and AUC, the proposed method is able to detect Benign, DDoS, PostScan, Bot, Web Attack Brute Force, Infiltration, DoS Slowris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, HeartBleed, FTP-Patator, and SSH-Patator.

The results of experiment on the performance improvement of attack detection are presented in Table 11. Based on the Recall and Sensitivity values, it can be seen that the detection system of the proposed method shows very good performance. In addition, based on the F1 Score, and AUC values, the prediction results of the proposed method can be relied upon. Furthermore, this study also considers the confusion matrix of the proposed method's classification performance as shown in Tables 12 and 13. The aim is to showing the correct classification among all classes.

Table 7
Performance of the proposed method on training data.

Test mode	Total instance	Correctly classified	Incorrectly classified	Accuracy
Hold-Out	1,981,520	1,981,087	443	99.98
5-Fold	1,981,520	1,978,755	2,765	99.86
10-Fold	1,981,520	1,978,740	2,780	99.86
Split-10	1,783,368	1,780,392	2,976	99.83
Split-20	1,585,216	1,582,661	2,555	99.84
Split-30	1,387,064	1,384,926	2,138	99.85
Split-40	1,188,912	1,187,129	1,783	99.85
Split-50	990,760	989,292	1,468	99.85
Split-60	792,608	791,458	1,150	99.85
Average				**99.86**

Table 8
Performance of the proposed method on testing data.

Test mode	Total instance	Correctly classified	Incorrectly classified	Accuracy
Hold-Out	849,223	849,117	106	99.99
5-Fold	849,223	847,926	1,297	99.85
10-Fold	849,223	847,952	1,271	99.85
Split-10	764,301	762,788	1,513	99.80
Split-20	679,378	679,255	1,123	99.83
Split-30	594,456	593,480	976	99.84
Split-40	509,534	508,664	870	99.83
Split-50	424,611	423,924	687	99.84
Split-60	339,689	339,155	534	99.84
Average				**99.85**

Table 9
Performance of the proposed method with Stratified k-fold.

Fold-n	Total instances	Correctly classified	Incorrectly classified	% Correctly classified	RMSE
1	283,075	283,056	19	99.993	0.0054
2	283,075	283,064	11	99.996	0.0051
3	283,075	283,057	18	99.993	0.0052
4	283,074	283,056	18	99.993	0.0052
5	283,074	283,055	19	99.993	0.0051
6	283,074	283,059	15	99.994	0.0052
7	283,074	283,054	20	99.992	0.0054
8	283,074	283,054	20	99.992	0.0054
9	283,074	283,059	15	99.994	0.0051
10	283,074	283,058	16	99.994	0.0052

Table 10
Detection system performance before performance improvement.

Class	Recall	Sen-sitivity	F1 Score	AUC
Benign	0.999	0.999	0.999	1.000
DDoS	1.000	1.000	1.000	1.000
PortScan	0.999	0.999	0.996	1.000
Bot	0.939	0.939	0.958	1.000
Web Attack Brute Force	0.897	0.897	0.844	1.000
Web Attack XSS	0.515	0.515	0.608	0.999
Web Attack Sql	0.625	0.625	0.769	1.000
Infiltration	1.000	1.000	1.000	1.000
DoS slowloris	0.999	0.999	0.999	1.000
DoS Slowhttptest	0.999	0.999	1.000	1.000
DoS Hulk	0.997	0.997	0.994	1.000
DoS GoldenEye	0.999	0.999	1.000	1.000
Heartbleed	1.000	1.000	1.000	1.000
FTP-Patator	1.000	1.000	1.000	1.000
SSH-Patator	1.000	1.000	1.000	1.000

Table 11
Detection performance of the proposed method.

Class	Recall	Sen-sitivity	F1-Score	AUC
Benign	1.000	1.000	1.000	1.000
DDoS	1.000	1.000	1.000	1.000
PortScan	1.000	1.000	0.999	1.000
Bot	1.000	1.000	1.000	1.000
Web Attack Brute Force	1.000	1.000	1.000	1.000
Web Attack XSS	1.000	1.000	1.000	1.000
Web Attack Sql	1.000	1.000	1.000	1.000
Infiltration	1.000	1.000	1.000	1.000
DoS slowloris	1.000	1.000	1.000	1.000
DoS Slowhttptest	1.000	1.000	1.000	1.000
DoS Hulk	1.000	1.000	1.000	1.000
DoS GoldenEye	1.000	1.000	1.000	1.000
Heartbleed	1.000	1.000	1.000	1.000
FTP-Patator	1.000	1.000	1.000	1.000
SSH-Patator	1.000	1.000	1.000	1.000

5.5. Visualization of detection result

The visualization shows the actual traffic that is classified correctly and misclassified by the detection system. A proper classified traffic is visualized with a cross symbol (x). Meanwhile, traffic that is classified incorrectly is visualized with a box symbol (□). The visualization of the detection results also aims to show a comparison of the performance of the detection system in identifying normal traffic and attacks. The performance of PSO-Search and RF before the performance improvement is presented in Fig. 7 and visualization of the results with performance improvement is presented in Fig. 8. Fig. 8 shows that the proposed method is able to identify normal traffic and attack traffic, and is able to identify the type of traffic according to its actual class. Visually this can be seen from misclassified traffic with fewer square symbols (□).

5.6. Comparison with state of the arts classification methods

Experiments on comparisons with other state of the arts classification methods including J48, REPTree, Bayes Network, Naïve Bayes, OneR, and Adaboost are also conducted. The experiments are carried out using training data and testing data with the same number and type of features. The results of comparison on training data and on testing data are presented in Tables 14 and 15, respectively.

Observing Tables 14 and 15, it appears that the proposed method has a better accuracy rate than other classification methods with an accuracy value of 99.978 % on training data and 99.987 % on testing data. Thus, it can be concluded that the proposed method is able to properly analyze the features generated from the performance-enhanced PSO-Search, so as to improve the performance of the proposed method.

5.7. Comparison of previous researches

The main objective of this research is to produce a feature selection technique that is effectively able to produce the most optimal and relevant features so that it reduces the network complexity of the IoT traffic model. Then, the optimal selected features are used to detect IoT

Table 12
Confusion matrix of detection system before performance improvement.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a	681031	8	273	12	9	0	0	0	0	0	662	0	0	0	0
b	5	38422	0	0	0	0	0	0	0	0	0	0	0	0	0
c	44	0	47426	0	0	0	0	0	0	0	17	0	0	0	0
d	35	0	0	539	0	0	0	0	0	0	0	0	0	0	0
e	11	0	0	0	408	36	0	0	0	0	0	0	0	0	0
f	3	0	0	0	94	104	0	0	0	0	1	0	0	0	0
g	2	0	0	0	1	0	5	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0
i	2	0	0	0	0	0	0	0	1737	0	0	0	0	0	0
j	1	0	0	0	0	0	0	0	0	1604	0	0	0	0	0
k	183	1	0	0	0	0	0	0	0	0	69075	0	0	0	0
l	1	0	0	0	0	0	0	0	0	0	1	3204	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
n	0	0	0	0	0	0	0	0	0	0	0	0	0	2422	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1831

*a=BENIGN, b=DDoS, c=PortScan, d=Bot, e=Web Attack BruteForce, f=Web Attack XSS, g=Web Attack Sql, h=Infiltration, i=DoS slowloris, j=DoS Slowhttpte, k=DoS Hulk, l=DoS GoldenEye, m=Heartbleed, n=FTP-Patator, dan o=SSH-Patator

Table 13
Confusion matrix of the proposed method.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
a	681900	0	67	0	0	0	0	0	0	0	28	0	0	0	0
b	0	38427	0	0	0	0	0	0	0	0	0	0	0	0	0
c	4	0	47483	0	0	0	0	0	0	0	0	0	0	0	0
d	0	0	0	574	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	0	455	0	0	0	0	0	0	0	0	0	0
f	0	0	0	0	0	202	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0
i	0	0	0	0	0	0	0	0	1739	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	0	1605	0	0	0	0	0
k	7	0	0	0	0	0	0	0	0	0	69252	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	3206	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0
n	0	0	0	0	0	0	0	0	0	0	0	0	0	2422	0
o	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1831

*a=BENIGN, b=DDoS, c=PortScan, d=Bot, e=Web Attack BruteForce, f=Web Attack XSS, g=Web Attack Sql, h=Infiltration, i=DoS slowloris, j=DoS Slowhttpte, k=DoS Hulk, l=DoS GoldenEye, m=Heartbleed, n=FTP-Patator, dan o=SSH-Patator

normal and attacks traffic. In Table 13, the experimental results and achievements of previous studies are presented. To maintain fairness, the proposed method is compared with the IDS studies that conduct experiments using the CICIDS-2017 dataset. Table 16 shows that the proposed method outperforms previous studies in terms of accuracy rate, where the proposed method has accuracy rate of 99.9%. Although the research in [9] has a lowest testing time, however, the experiment was carried out on a limited sample data, and considered only one type of attack (DDoS attack). In addition, it only focuses on three features to be analyzed. In fact, to detect more than one type of attack, multiple features are required. Table 16 concludes that the proposed lightweight IDS, while maintaining execution time, still being able to detect various possible attacks on IoT networks.

Experiments on testing data and on training data using hold-out method provide accuracy of 99.99 % and 99.98 %, respectively; while experiments on validation using Stratified k-fold cross validation produce average accuracy of 99.99 %. On the other hand model validation results using several classifiers provide average accuracy of 99.85% on

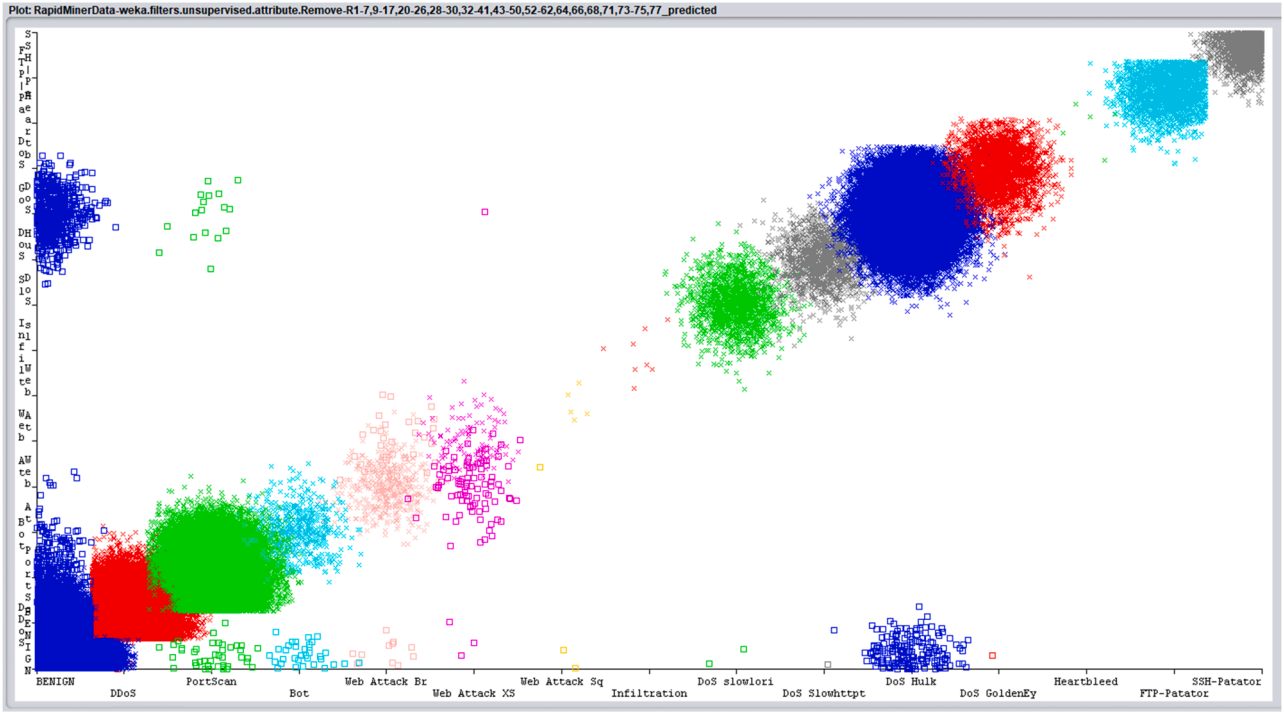

Fig. 7. Visualization of traffic detection before performance improvement.

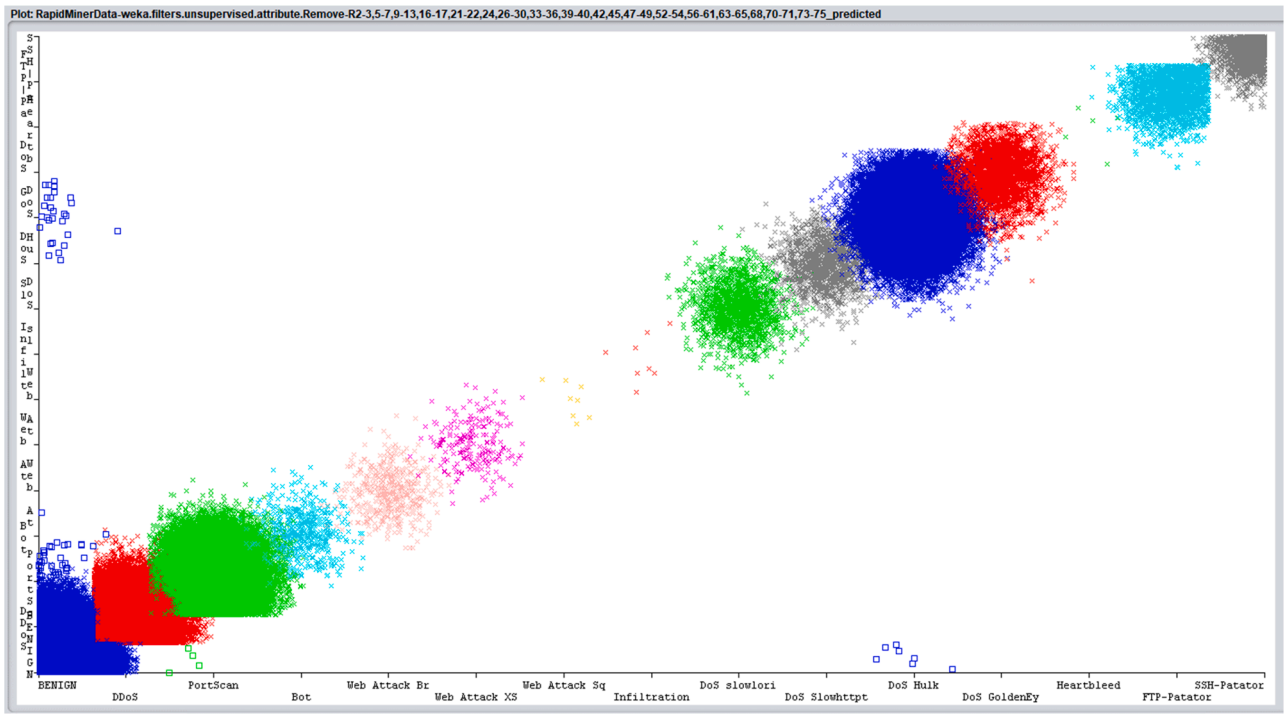

Fig. 8. Visualization of the detection results of the proposed method.

Table 14
Comparison of classification performance on training data.

Method	Total instance	Correctly classified	Incorrectly classified	Accuracy
J48	1,981,520	1,979,323	2197	99.8891
REPTree	1,981,520	1,979,306	2214	99.8883
Bayes Network	1,981,520	1,952,260	29,260	98.5234
Naïve Bayes	1,981,530	1,351,674	629,856	68.214
OneR	1,981,520	1,853,082	128,438	93.5182
Adaboost	1,981,520	1,682,702	298.818	84.9198
Proposed	1,981,520	1,981,087	433	99.9781

Table 15
Comparison of classification performance on testing data.

Method	Total instance	Correctly classified	Incorrectly classified	Accuracy
J48	849,223	848,233	990	99.8834
REPTree	849,223	848,189	1034	99.8782
Bayes Network	849,223	833,412	15,811	98.1382
Naïve Bayes	849,223	583,913	265,310	68.7585
OneR	849,223	792,902	56,321	93.3679
Adaboost	849,223	721,257	127,966	84.9314
Proposed	849,223	849,117	106	99.9875

testing data, and 99.99 % on training data (refer to Table 7 and Table 8). All the results show consistency on accuracy that means the proposed PSO-RF classifier method improves the detection accuracy. In other word, the proposed PSO-RF is able to generalize to new dataset, since the PSO contributes to the best feature selection that in turn, reduces the dataset complexity, eliminates overfitting problem and increase the accuracy.

6. Conclusion

In this study, intrusion detection system for IoT networks is proposed, which is a combination of PSO-Search feature selection technique and Random Forest classifier algorithm. Improving the performance of PSO-Search is achieved by tuning the number of particle swarm parameters so that the ideal number of particle swarms is obtained. The experimental results show that the number of particle swarms affects the

Table 16
Comparison of performance with state of the arts.

Studies	Method	Number of records	Execute time (Sec.)	Performance
[7]	LM-BP	1000 (Normal, and 4 types of attacks)	N/A	Detecting Rate: 93.31 %
[9]	SVM Based-Classifier	90,298 (Benign and DDoS)	2.281	Accuracy: 98 %, F1-Score and Precision note mentioned
[10]	Comparing logistic regression (LR), naive Bayes (NB), decision tree (DT), random forest (RF), k-nearest neighbor (KNN), support vector machine (SVM) and multilayer perceptron (MLP).	311,029 (includes benign and 9 types of attacks)	N/A	Accuracy: 98 %, F1-Score: 0.900, Precision: 0.910
[15]	C5 classifier	18,494 (Normal, and 4 types of attacks)	N/A	Accuracy: 9.97 %, F-Measure: 0.957
The Proposed (*PSO-RF)*	Feature selection using Correlation-Based with Optimized PSO Search method and Random Forest Classifier	849,223 (includes benign and 14 types of attacks)	28.39	Accuracy 99.9 %, Avg, F1 Score(F Measure) 0.999, Avg. Sensitivity: 1.000, Recall: 1.000

number and the types of features produced. With the ideal number of particle swarms N = 7, 28 of the most relevant features are generated. RF performance improvement is achieved by tuning the parameter number of the Trees. The experimental results show that the Tree with 100 parameters produces a very good random tree performance in detecting normal traffic and 14 types of attack traffics. Overall, the proposed method has a better ability than other methods with an accuracy rate of 99.9781 % on training data and 99.9875 % on testing data. By paying attention to the values of Recall, Sensitivity, F1 Score and AUC, it can be concluded that the proposed method is able to work well to detect attacks on IoT networks. The optimal number of traffic features reduces the complexity of the network model of the problem, which is in turn speed up the detection time.

Our future research is to consider an adaptive value of particle swarm, so as to produce a more intelligent feature selection technique. Future detection systems must also be able to adapt to traffic developments as well as the increasing sophisticated forms of attacks on internet of things networks.

CRediT authorship contribution statement

We Declare The article has not been previously published in whole or in part, and that it is not being considered for publication elsewhere. All authors have read the final manuscript, have approved the submission to the journal, and have accepted full responsibilities pertaining to the manuscript's delivery and contents. We have sent the manuscript to our English Language Center, Universitas Sriwijaya for proof reading. We declare that there is no conflict of interests on the submitted manuscript. We are also aware of the APC of the journal upon the acceptance; therefore, we are willing to pay it.

Declaration of Competing Interest

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

References

[1] B.B. Zarpelão, R.S. Miani, C.T. Kawakani, S.C. de Alvarenga, A survey of intrusion detection in Internet of Things, J. Netw. Comput. Appl. 84 (2017) 25–37, https://doi.org/10.1016/j.jnca.2017.02.009.

[2] H. Guo and J. Heidemann, IP-based IoT device detection. In: Proceedings of the 2018 Workshop on IoT Security and Privacy - IoT S&P '18, 2018, 36–42. doi:10.1145/3229565.3229572.

[3] M. Ammar, G. Russello, B. Crispo, Internet of things: a survey on the security of IoT frameworks, J. Inf. Secur. Appl. 38 (2018) 8–27, https://doi.org/10.1016/j.jisa.2017.11.002.

[4] J.C.S. Sicato, S.K. Singh, S. Rathore, J.H. Park, A comprehensive analyses of intrusion detection system for IoT environment, J. Inf. Process. Syst. 16 (4) (2020) 975–990, https://doi.org/10.3745/JIPS.03.0144.

[5] S. Choudhary, N. Kesswani, A survey: Intrusion detection techniques for Internet of Things, Int. J. Inf. Secur. Priv. 13 (1) (2019) 86–105, https://doi.org/10.4018/IJISP.2019010107.

[6] C.D. McDermott, F. Majdani, and A.V. Petrovski, Botnet detection in the Internet of Things using deep learning approaches. In: Proceedings of the International Joint Conference on Neural Networks, vol. 2018-July, 1–8. doi:10.1109/IJCNN.2018.8489489.

[7] A. Yang, Y. Zhuansun, C. Liu, J. Li, C. Zhang, Design of intrusion detection system for Internet of Things based on improved BP neural network, IEEE Access 7 (2019) 106043–106052, https://doi.org/10.1109/ACCESS.2019.2929919.

[8] P. Nimbalkar, D. Kshirsagar, Feature selection for intrusion detection system in Internet-of-Things (IoT), ICT Express 7 (2) (2021) 177–181, https://doi.org/10.1016/j.icte.2021.04.012.

[9] S.U. Jan, S. Ahmed, V. Shakhov, I. Koo, Toward a lightweight intrusion detection system for the Internet of Things, IEEE Access 7 (2019) 42450–42471, https://doi.org/10.1109/ACCESS.2019.2907965.

[10] S. Fenanir, F. Semchedine, A. Baadache, A machine learning-based lightweight intrusion detection system for the internet of things, Rev. d'Intelligence Artif. 33 (3) (2019) 203–211, https://doi.org/10.18280/ria.330306.

[11] D. Stiawan, M.Y. Idris, R.F. Malik, S. Nurmaini, N. Alsharif, R. Budiarto, Investigating brute force attack patterns in IoT networks, Article ID 4568368, J. Electr. Comput. Eng. (2019), https://doi.org/10.1155/2019/4568368.

[12] H.F. de Arruda, C.H. Comin, F. da F. Costa, Problem-solving using complex networks, Eur. Phys. J. B 92 (2019), 132, https://doi.org/10.1140/epjb/e2019-100100-8.

[13] R. Barham, A. Sharieh, A. Sleit, Multi-moth flame optimization for solving the link prediction problem in complex networks, Evol. Intel. 12 (2019) 563–591, https://doi.org/10.1007/s12065-019-00257-y.

[14] W. Yeh, Y. Lin, Y.Y. Chung, M. Chih, A particle swarm optimization approach based on Monte Carlo simulation for solving the complex network reliability problem, IEEE Trans. Reliab. 59 (1) (2010) 212–221, https://doi.org/10.1109/TR.2009.2035796.

[15] M.R. Mirsaleh, M.R. Meybodi, A Michigan memetic algorithm for solving the community detection problem in complex network, Neurocomputing 214 (2016) 535–545, https://doi.org/10.1016/j.neucom.2016.06.030.

[16] R. Panigrahi, S. Borah, A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems, Int. J. Eng. Technol. 7 (24) (2018) 479–482.

[17] N. Farnaaz, M.A. Jabbar, Random forest modeling for network intrusion detection system, Procedia Comput. Sci. 89 (2016) 213–217, https://doi.org/10.1016/j.procs.2016.06.047.

[18] J. Lee, D. Park, and C. Lee, Feature Selection Algorithm for Intrusions Detection System using Sequential Forward Search and Random Forest Classifier, 11(10) (2017), 5132–5148.

[19] A. Abd, A. Hadi, Performance analysis of big data intrusion detection system over random forest algorithm, Int. J. Appl. Eng. Res. 13 (2) (2018) 1520–1527.

[20] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, A. Alazab, A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks, Electronics 8 (11) (. 2019), https://doi.org/10.3390/electronics8111210.

[21] N. Balakrishnan, A. Rajendran, D. Pelusi, V. Ponnusamy, Deep belief network enhanced intrusion detection system to prevent security breach in the Internet of Things, Internet Things 14 (2021), https://doi.org/10.1016/j.iot.2019.100112.

[22] Z. Groff and S. Schwartz, Data preprocessing and feature selection for an intrusion detection system dataset. In: Proceedings of the 34th Annual Conference of The Pennsylvania Association of Computer and Information Science Educators, (2019) 103–110. [Online]. Available: ⟨http://granite.sru.edu/~pacise/proceedings/pacise-proceedings-2019.pdf⟩.

[23] S. Huang, K. Lei, IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks, Ad Hoc Netw. 105 (2020), https://doi.org/10.1016/j.adhoc.2020.102177.

[24] I. Sharafaldin, A.H. Lashkari, and A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy, vol. 2018-Janua, no. Cic, pp. 108–116, 2018, doi:10.5220/0006639801080116.

[25] I. Syarif, R.F. Afandi, and F. Astika Saputra, Feature selection algorithm for intrusion detection using cuckoo search algorithm. In: IES 2020 - International Electronics Symposium: The Role of Autonomous and Intelligent Systems for Human Life and Comfort, 2020, 430–435. doi:10.1109/IES50839.2020.9231840.
[26] A.I. Madbouly, T.M. Barakat, Enhanced relevant feature selection model for intrusion detection systems, Int'l J. Intell. Eng. Inform. 4 (1) (2016) 21, https://doi.org/10.1504/ijiei.2016.074499.
[27] I. Syarif, Feature selection of network intrusion data using genetic algorithm and particle swarm optimization, EMITTER Int. J. Eng. Technol. 4 (2) (2016) 277–290, https://doi.org/10.24003/emitter.v4i2.149.
[28] K. Singh, B. Nagpal, Random forest algorithm in intrusion detection system: a survey, Int'l J. Sci. Res. Comp. Sci., Eng. IT 3 (5) (2018) 673–676.
[29] B.A. Tama and K.H. Rhee, An integration of PSO-based feature selection and random forest for anomaly detection in IoT network. In: Proceedings of the MATEC Web of Conferences, vol. 159, 2018, doi:10.1051/matecconf/201815901053.

Kurniabudi received a Doctor of Engineering degree from Universitas Sriwijaya. He is currently a Senior Lecturer at the Faculty of Computer Science, Universitas Dinamika Bangsa, Indonesia. His research interests include technology adoption, information technology, information security, and network security

Deris Stiawan received a Ph.D. degree in Computer Engineering from Universiti Teknologi Malaysia, Malaysia. He is currently an Associate Professor at the Department of Computer Engineering, Faculty of Computer Science, Universitas Sriwijaya. His research interests include computer networks, intrusion detection/prevention systems, and heterogeneous networks.

Darmawijoyo received a Doctor of Mathematics degree from the Delft University of Technology, Netherlands. He is currently an Associate Professor at the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Sriwijaya. His research interests include problem-solving, applied mathematics, modeling, and mathematical thinking.

Mohd Yazid Bin Idris. Received an M.Sc. degree in Software Engineering in 1998 and a Ph.D. degree in Information Technology (IT) Security in 2008. He is currently an Associate Professor at the Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia. In Software Engineering, he focuses on the research of designing and developing mobile and telecommunication software. His main research interest in IT security includes intrusion prevention and detection (IPD).

Sarjon Defit received a Ph.D. degree in Computer Engineering from Universiti Teknologi Malaysia, Malaysia. He is currently a Professor at the Department of Information System, Faculty of Computer Science, Universitas Putra Indonesia YPTK Padang, Indonesia. His research interests include artificial intelligence applications, expert systems and big data. analysis.

Yaya Sudarya Triana received B.Sc. degree in Statistics from University of Padjadjaran, Indonesia in 1988, M.Kom. in Master of Information Technology from University of Indonesia in 2002 and Ph.D. in Statistics from Universiti Malaysia Trengganu, Malaysia in 2013, Software Engineer 1991–2003 at an IT Company at Tokyo, Bandung & Jakarta, respectively. Currently, he is an Assistant Professor at Dept. of Information System, Universitas Mercu Buana Indonesia. His research interests include Fuzzy Systems, Data Science, Artificial Intelligent, Business Intelligent, Data Analytics, Statistics, Information System, and Machine Learning.

Rahmat Budiarto received a B.Sc. degree from Bandung Institute of Technology in 1986 and an M.Eng. and Dr.Eng. degrees in Computer Science from Nagoya Institute of Technology in 1995 and 1998, respectively. He is currently a Full-Time Professor at the College of Computer Science and IT, Albaha University, Saudi Arabia. His research interests include intelligent systems, brain modeling, IPv6, network security, wireless sensor networks, and MANETs.