Suboptimal Feature Selection Parameter Optimization Scheme for Efficient Malicious Traffic Detection

So-Eun Jeon[1], Ye-Sol Oh[1], Yeon-Ji Lee[1], Il-Gu Lee[1,2*]

[1] Department of Future Convergence Technology Engineering, Sungshin Women's University
{220237020, 220224010, 220226036, iglee}@sungshin.ac.kr
[2]Department of Convergence Security Engineering, Sungshin Women's University

Abstract

There has been a considerable rise in the use of the Internet of Things (IoT) in industrial as well as domestic applications. The widespread use has raised concerns such as threats against personal data, in addition to economic security threats. Conventionally, static and dynamic analyses were used to detect malicious codes. However, in these methods, real-time detection is difficult. In recent years, machine learning has been used to detect diverse types of malicious traffic efficiently, but it is difficult to process big data in lightweight devices such as IoT-integrated devices, especially in machine-learning environments. Consequently, a feature selection technique was used to reduce the computational costs and enhance the performance of the learning model by removing less relevant or overlapped features during the learning process. Conventional studies, however, primarily focused on enhancing learning performance, without considering how to simultaneously optimize learning performance and computational complexity issues in lightweight device environments. In this study, we propose a suboptimal feature selection model (SFSM), which improves complexity while maintaining malicious traffic performance in IoT environments. The SFSM optimizes the hyperparameters using a grid search technique that selects and explores specific areas within a reduced range among all features. It was observed that the detection accuracy improved by approximately 20% compared to the random model, and the accuracy reduction rate was maintained to ensure security, contrary to the greedy model. Meanwhile, the latency was reduced by approximately 96% and the complexity was improved by 99.78% compared to that of the greedy model.

Keywords: Feature selection, Lightweight device, Machine learning, Malicious traffic

1 Introduction

The Internet of Things (IoT) has become an integral part of individuals' daily lives and is expected to have a significant impact on economic and commercial aspects in the forthcoming years. According to the Internet of Things 2020 report released by Business Insider, the continuous growth of the IoT industry will be the driving force behind changes the environment in all industries. The IoT market is expected to grow by more than $2.4 trillion annually by 2027 [1]. Further, more than 41 billion IoT devices are expected to be connected by 2027.

However, existing lightweight IoT technologies face challenges such as limited resources and security vulnerabilities [2]. As the IoT technology becomes deeply embedded in daily personal life, there are concerns that security threats can cause significant harm to personal information

The 7th International Conference on Mobile Internet Security (MobiSec'23), December 19-21, 2023, Okinawa, Japan
Article No. W-2

*Corresponding author: Department of Future Convergence Technology Engineering, Sungshin Women's University, seoul, (02844), Republic of Korea, Tel. (+82) 02-920-7145 protection and the economy [3]. Security threats that result in malicious traffic or compromise the confidentiality, integrity, and availability of systems through the distribution of malicious codes remain a challenge in various fields, including IoT devices [4]. The typical detection methods for malicious attacks include static and dynamic analyses, signature-based detection, and behavior-based detection. Static analysis involves analyzing and detecting malicious codes without executing them, whereas dynamic analysis entails directly executing malicious codes and detecting abnormal behaviors through monitoring. The main advantage of static analysis is the ability to analyze code without direct execution, which helps observe the structure of the malicious code. However, this requires considerable time and expertise and has limitations in terms of preemptive prevention. Conversely, dynamic analysis allows the observation of actual operational processes and IP flows that cannot be discerned solely from the code. However, it is resource-intensive and has the limitation that malicious codes can infect real computers [5].

The aforementioned conventional countermeasures are post-response methods because they analyze malicious behavior after its occurrence, making it impossible to detect it in advance. Signature-based detection involves creating databases of the behavioral patterns of malicious code and classifying them as malicious if they match. This is the most widely used method because it provides the best detection accuracy for previously identified malicious behavior. However, it does not effectively respond to rapidly changing variants of malicious code or zero-day attacks because it can only address previously occurring patterns of malicious code [6]. The behavior-based detection technique identifies malicious behavior when different behavior patterns occur. It is designed to overcome the limitations of the aforementioned signature-based detection techniques. Unlike signature-based detection methods that rely on databases, the merit of behavior-based detection methods is that they can detect new attacks. However, the category of normal behavior is quite broad, leading to a relatively high false-positive rate compared to signature-based detection techniques. Consequently, conventional signature-based and behavior-based detection techniques are not suitable for use in IoT devices with limited resources because they rely on databases with established detection performance.

To address the limitations of conventional technologies, machine learning-based detection technology has been actively studied for use in detecting malicious behaviors [7]. Machine learning can recognize patterns and behaviors through data learning. However, it is not ideal for IoT devices with limited resources. Therefore, recent studies have focused on reducing the complexity of machine learning models or reducing the learning time to enable their use in lightweight devices [8]. However, the use of a high-quality machine learning model requires a vast dataset. In other words, it is difficult to store and learn extensive log datasets effectively to detect malicious behavior in IoT environments. This paper proposes a suboptimal feature selection model (SFSM) to efficiently detect malicious behavior in resource-limited IoT environments, and it addresses the trade-off between detection performance and complexity that conventional feature selection techniques do not consider. The SFSM conducts feature selection based on the importance of features and learning by thoroughly investigating feature sets. During the feature selection process, the number of features subject to exhaustive search and the data sampling ratio parameters were optimized in a suboptimal manner to maintain the detection performance while reducing complexity.

The primary contributions of this paper are as follows:

- We propose a suboptimal method that reduces complexity while maintaining accuracy by optimizing the parameters used in feature selection.
- The performance of the proposed SFSM was evaluated and compared with that of conventional methods using detection accuracy, latency, and complexity evaluation metrics,

which are crucial in the IoT environment.

- By introducing a new lightweight detection model that enhances conventional feature selection techniques, we improved the tradeoff between abnormal behavior detection performance and resource complexity in IoT devices.

The remainder of this paper is organized as follows. Section 2 analyzes previous studies on the detection of abnormal behavior in IoT environments. Section 3 explains the proposed SFSM method. Section 4 details the experimental environment for evaluating the performance of the SFSM and analyzes the experimental results. and Section 5 concludes the paper.

2 Related work

To enhance the security of IoT networks, it is essential to detect and block malicious traffic. In recent years, machine-learning technology has been widely employed to accurately detect and identify malicious attacks in IoT network environments. Machine learning can accurately detect various behavioral patterns. However, a higher complexity is involved with this process, compared to other detection methods [9]. The complexity of machine learning-based malicious behavior detection technology is determined by the learning method and dataset. For instance, in cases where there is significant overlapping data or features that are not related to the learning objective, the computational complexity increases and the detection performance decreases [10]. Therefore, it is important to extract and learn appropriate features to effectively detect malicious traffic in IoT devices [9]. Further, an effective feature selection technology that removes low-relation or overlapping features is important [10].

Table 1 presents the characteristics, contributions, and limitations of previous technologies used to detect malicious behaviors in IoT environments.

As conventional methods were primarily focused on detecting malicious behavior without the use of machine learning, there has been increasing research aimed at detecting malicious behavior, emphasizing network behavior analysis in large networks [11, 12]. A previous study [11] proposed a method for performing OWASP top 10 vulnerability-based static analysis for IoT vulnerability analysis because static analysis is the most effective method for detecting security vulnerabilities in IoT devices. However, this involves inspecting the API of IoT devices after performing a static analysis. However, there are limitations in that expertise and resources are required. Furthermore, it is not clear whether it has improved compared to conventional technology mainly because performance verification and resource considerations have not been made. A previous study [12] used static and dynamic analyses to extract features such as function calls and traffic patterns from various malicious codes, and they compared them to analyze the evolution of malicious codes. This prior study analyzed the spread trend of IoT malicious code through large-scale network traffic analysis. However, it was difficult to detect malicious traffic patterns that were still diverse compared to machine learning techniques. As IoT environments possess limited resources, complex and resource-consuming detection methods are not ideal given that they require lightweight protection mechanisms [16].

Another previous study [13] attempted to improve the complexity and proposed a Tabu Search-Random Forest (TS-RF) feature selection technique aimed at reducing the dimensionality of high-dimensional data. The tabu search is a high-speed search method based on a metaheuristic optimization algorithm. The solution fit of the neighboring features in the tabu list was calculated and moved to the neighboring node with a higher fit. In this previous study [13], 'move' refers to the creation of a neighbor node by randomly adding or deleting features

Table 1: Previous studies of malicious traffic detection method

Category	Main focus	Ref.	Techniques	Contribution	Limitation
Non-machine learning detection model	Static/ Dynamic analysis	[11]	Proposed extension of static analyzer to IoT system	Introduced the expansion direction of the static analyzer through vulnerability analysis	Did not consider resources
		[12]	Used Static/Dynamic analysis to extract features	IoT malicious code spread trends can be analyzed	- Poor performance and less information available than machine learning - Did not consider resources
Machine learning detection model	Reducing complexity	[13]	Proposed Tabu Search-Random Forest FS	- Reduced computational complexity by reducing feature space and vector - Improved detection accuracy even for attacks with a small number of data samples	Because the tabu list must be referenced every time, a reference delay occurs as the number of features increases
		[14]	Estimated the importance of features and select the top k features	Improved learning performance and shortened learning time	- Insufficient basis for selecting the importance threshold - Did not consider resource
		[15]	- Feature selection based on correlations - Verified learning performance with DNN model in multiple scenarios	Improved learning time and calculated amount through feature selection	Some datasets result in poor performance
	Improving detection performance	[16]	Lightweight signature generation through multi-level clustering	- Lightweight detection method suitable for IoT environment - Improved malware detection rate through cluster merging	- Unable to detect new malware - Cluster merging process takes a long time
		[17]	- Improved attack detection rate over standard WOA - Reduced search space by adding an intersection operator	- Detected network attacks with high accuracy - Improvement of WOA's local optimization problem	High computational complexity and time delay due to GA operators
		[9]	- Effective feature set filtering with Bijective soft set - Improved accuracy using CAE and ACC metrics together	- The first study using the Corr ACC metric for botnet identification attacks - Performance measurement for various machine learning classifiers	Computational complexity is not considered when reducing the number of features
		[18]	Built a malicious Android application package dataset for accurate feature selection	- Shorter detection time compared to commercial anti-virus scanners - High detection rate compared to previous literature	It did not consider limited resources

from the feature vector. However, when a predefined number of iterations is reached, improvements cannot be attained after a certain number of iterations, or the objective function reaches the required threshold. Subsequently, the search is stopped. In this study, to prevent local optimization problems, recently explored solutions were stored in a tabu list, and each time a solution was moved, the tabu list was used to prevent the next move. This study [13] reduced the computational complexity by reducing the feature space and vector by more than 60% and improved the detection accuracy, even in attacks with small data samples. However, because the tabu list is referenced at every movement to prevent local optimization, the reference delay may increase with the number of features. A recent study [14] focused on smartphones using the Android operating system among IoT devices and proposed a general automated classification framework for Android malware detection. The framework involves three main steps—preprocessing the dataset, including feature selection, applying feature subset selection methods, and proceeding with classification. At this time, feature subset selection methods strengthen the training of the learning algorithm and improve the classification performance using features that highly correlate with the purpose of classification. Accordingly, the importance of the extent to which the original dataset affected the classification was estimated, and the top-k features were selected. The proposed method improves learning performance and reduces the learning time. However, it has not been sufficiently considered and analyzed in terms of resource usage, which is a critical evaluation metric in the IoT environment. A recent study [15] addressed the problem of handling large and high-dimensional data in an IoT environment with limited resources. This study proposes a high-performance malware detection system using deep learning and feature selection. The method proposed in this previous study preprocesses the dataset and performs feature selection based on correlations. Subsequently, the LSTM-based DNN model was learned, and its performance was evaluated in various scenarios by combining the presence or absence of the FS model, number of features, and various machine learning models. A previous study [15] implemented a lightweight model for IoT devices by minimizing the number of features and reducing the learning time and computation by utilizing feature selection. However, as the number of features is reduced by more than half on a Unix/Linux-based platform, the accuracy of the performance decreases, and the latency is similar to that of a model that does not perform feature selection.

Previous research [16] focused on improving the detection performance and creating a lightweight signature to overcome the resource constraints of IoT devices. It was assumed that IoT malware is often simple, without obfuscation or evasion techniques, and it was argued that the detection rate of IoT malware can be increased using a high-level code structure and the string function capture method. The proposed mechanism is a multistage clustering technique that clusters IoT malware samples into several families using n-gram string functions. They underwent coarse-grained, fine-grained, and cluster-merging processes. A combination of the coarse- and fine-grained clustering can reduce the clustering calculation cost compared to using only fine-grained clustering, and the malware detection rate can be improved through cluster merging. This paper proposes a model that is suitable for IoT devices and improves the malware detection rate by generating lightweight signatures. However, because it is difficult to detect malware that has low similarity to conventional malware when detected based on the similarity score, continuous IoT sample updates are required. Further, a major limitation is that the cluster merging process requires significant time. A previous study [17] proposed the Whale Optimization Algorithm (WOA), which improved the attack detection rate compared to the standard WOA. Adding intersection and mutation operators to the standard WOA helps improve the search space and prevents local optimization problems. The improved WOA designates a randomly selected feature from the initial dataset as the initial location and evaluates the fitness of each feature subset using an SVM. The position of the whale is updated based on the best feature subset, and the diversity of the solutions is increased through intersection and mutation operators. This process was repeated several times to determine the most efficient feature set, and the selected features were used as inputs for the attack-detection classifier. The IDS wherein the WOA proposed in this previous study was integrated can detect network attacks with high accuracy by selecting relevant features and overcoming the local optimization problem of the WOA. However, the use of genetic algorithm (GA) operators increases computational complexity and time delay, and the detection rate for classes with a small number of datasets is low. Previous research [9] filtered features using a bijective softset to select an effective feature set for botnet IoT attacks in IoT networks and improved the accuracy using correlation attribute evaluation (CAE) and accuracy metrics. This study also constructed a correlation table between features, performed union and intersection operations between columns and rows, selected meaningful features, and assigned weights by ranking features using the Pearson correlation coefficient. The method proposed in this study selected seven out of 39 features in the dataset and accurately identified traffic when using a decision tree and random forest. This study is the first to use the Corr and ACC metrics to identify IoT botnet attacks and measure the performance of various machine learning classifiers. However, their use in IoT devices is limited because computational complexity is not considered in reducing the number of features through feature selection. A previous research [18] attempted to build an effective Android malware-detection model, arguing that correct feature selection affects malware-detection performance. For this purpose, they collected Android application package files and determined whether they were malicious based on the results of an antivirus scanner. In addition, they constructed a dataset that extracts permissions and API calls. Subsequently, the final dataset was created through feature selection and used as the input values to implement the least-squares support vector machine (LS-SVM) model. The model proposed in this previous study was compared with various antiviral scanners used in the market, and it was observed that it reduced the time required for detection compared to conventional systems. The model showed a 3% higher detection rate compared to models proposed in the literature. However, this study did not analyze improvements related to limited resources or essential considerations for IoT devices.

Thus, the static and dynamic analysis research that has not applied machine learning in previous studies shows that it is inappropriate for application in an IoT environment with limited resources because it requires specialized personnel and does not consider limitations in support. However, existing studies that integrate machine learning aims to improve detection performance or computational complexity and considers the resource-constrained environment of IoT rather than non-machine learning detection models. Overall, the evaluation metrics for limited resources have not been sufficiently considered, and the complexity and latency increase with the introduction of machine learning models. In addition, it does not solve the trade-off problem of lower performance when minimizing resource use. Therefore, research on the tradeoff between detection performance and complexity is required to efficiently detect malicious network traffic in resource-constrained IoT environments.

3 Proposed scheme

This section describes the feature search mechanism and the overall operation of the proposed SFSM.

3.1 Search algorithm for feature selection

Feature selection is aimed at selecting a minimal representative feature subset from the entire feature set. This approach removes redundant and irrelevant features from the dataset. Determining the optimal subset of all features is challenging and is considered a combinatorial NP-hard problem [19]. This section describes the search algorithms used in the feature selection process.

The conventional exhaustive search method was a widely adopted conventional method for finding an optimal feature set [13]. An exhaustive search examines all possible feature subsets to determine the optimal features exactly; however, the search complexity is very high at 2^n [13, 20], where n represents the number of features. The greater the feature dimensions, the more exponential the increase in the search complexity. This further leads to NP-hard problems [13, 20]. In other words, the exhaustive search method is not suitable for the IoT environment and has high computational complexity because it requires significant computational time and memory to obtain an optimal solution [20]. To address this issue, search algorithms such as random search and metaheuristics have emerged [13].

A random search can continuously generate subsets and thus improve the quality of the selected features through repeated selection [19]. It also prevents local optimization problems by introducing randomness into the search procedure [13]. In each step, the next subset was randomly generated using the information collected in the previous step [19]. However, in the worst-case scenario, we may need to explore all possible solutions, leading to problems that are similar to those encountered in an exhaustive search [19].

The metaheuristic method is a general-purpose optimization technology that can determine the optimal solution in a reasonable time. Metaheuristic methods are divided into single-solution-based (S-metaheuristics) and population-based (P-metaheuristics) methods [19]. The S-metaheuristics construct and iterate a single solution for improvement, whereas P-metaheuristics generate multiple solutions in each iteration and improve them by selecting the best available solution [13]. Metaheuristic techniques are characterized by their ability to determine a reasonable solution without exploring the entire search space [20]. These techniques probabilistically solve optimization problems with the randomness of random search, and they are optimization processes that randomly explore and utilize search spaces at specific probabilities, starting with a random solution [21]. Metaheuristic techniques have been widely utilized owing to excellent performance capability; however, increasing the dimensions of the dataset can affect their performance [21]. Additionally, a reasonable solution does not necessarily guarantee an optimal solution [20].

Conventional search methods are not suitable for lightweight IoT environments due to high computational complexities. Furthermore, a consistently good performance is not always achieved because the derived feature set is not necessarily optimal. Therefore, this study suggests the use of a grid search method, which is a lightweight search approach, for feature selection while obtaining an optimal solution. A grid search involves selecting a combination of feature sets that demonstrate optimal performance by exploring candidate values from the entire set. This method addresses the complexity problem associated with efficient but highly complex exhaustive search algorithms, in addition to the performance problem of random search, which is efficient but lacks performance optimization.

3.2 Sub-optimal feature selection model

In this section, we describe the proposed SFSM. Fig.1 illustrates the overall operational structure of the SFSM.

The SFSM operates in four phases—data preprocessing, input parameter optimization, feature selection, and attack classification. First, data preprocessing was conducted on the dataset to derive the final features for selection. The input parameter optimization was then performed on the preprocessed dataset. In the initial step of feature selection, the top k features are selected from the entire feature set using permutation importance [22]. The permutation importance ranks features based on their impact on the classification performance of the dataset. Additionally, the SFSM employs data sampling to reduce computational complexity while ensuring that the performance is not compromised. During this process, parameter optimization

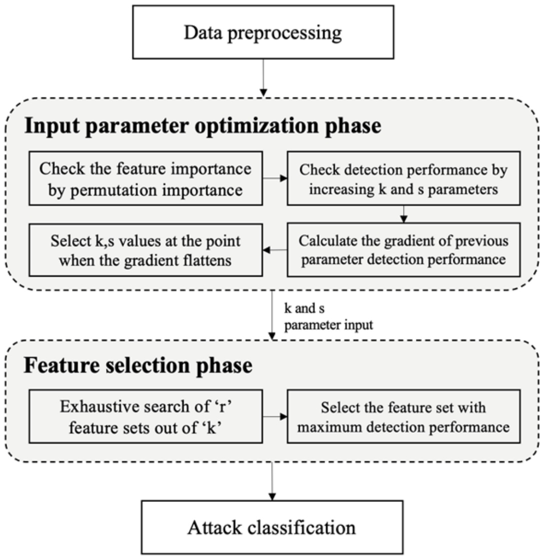
Data preprocessing
Input parameter optimization phase
Check the feature importance by permutation importance
Check detection performance by increasing k and s parameters
Calculate the gradient of previous parameter detection performance
Select k,s values at the point when the gradient flattens
k and s parameter input
Feature selection phase
Exhaustive search of 'r' feature sets out of 'k'
Select the feature set with maximum detection performance
Attack classification

Figure 1: Flowchart of SFSM

was performed to determine the optimal values of k and s. Here, k represents the number of features considered in the exhaustive search and s represents the data sampling ratio. Initially, k and s values were inputted, and the detection performance was evaluated by incrementally adjusting each parameter. The detection performance results for each iteration were used to calculate the gradient of the rate of increase in performance compared to the previous parameter setting. When the gradient flattened, further parameter adjustments did not have a significant impact on the performance. Therefore, the k and s parameters are selected at this point, and feature selection proceeds based on these chosen values. Data were sampled according to the s value, which is the data sampling ratio, and an exhaustive search was performed to find the optimal r feature set from the k selected features. In essence, this process evaluates the detection accuracy performance for all possible feature sets kCr, and the feature set with the best accuracy is chosen for subsequent learning. Fig. 2 illustrates the system configuration of the SFSM.

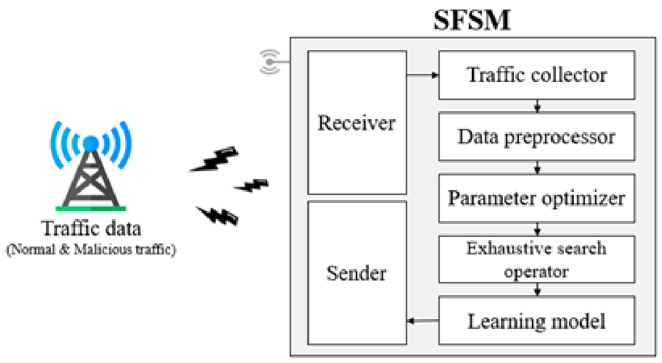

Figure 2: System architecture of SFSM

As depicted in Fig. 2, the SFSM receives mixed traffic that comprises both normal and malicious traffic from the receiver and aggregates it using a traffic collector. Subsequently, the collected traffic data were preprocessed using the data preprocessor, and the optimized parameters were selected using the parameter optimizer. The k and s parameters determined using the parameter optimizer were applied to the dataset for an exhaustive search. The goal was to identify the optimal feature set that exhibited the highest detection performance. This selected feature set served as an input for the learning model, which was responsible for attack classification.

4 Performance Evaluation

4.1 Experimental Environment

In this section, we describe the experimental environment used to demonstrate the performance of the proposed SFSM. This experiment was modeled using Python3 in an Intel(R) Core(TM) i9-10850K 3.60 GHz CPU in a 32.0 GB RAM environment. In addition, to implement an environment wherein malicious traffic attacks occur, the UNSW-NB15 dataset [23] was used and learned using a decision tree, which is a machine learning model. In this experiment, we aimed to multi-classify one normal label and two attack labels, Benign, Exploits, and Generic, which possess sufficient data. The Benign label refers to normal traffic, exploits refer to attacks using vulnerabilities, and a generic label refers to attacks that infiltrate by bypassing the block cipher. Table 2 lists the number of data points for each label and the number of data points after preprocessing.

Table 2: Number of data points by label

Label	Number of data points	
	Before preprocessing	After preprocessing
Benign	37,000	10,000
Exploits	11,132	10,000
Generic	18,871	10,000

The number of data points was balanced at 10,000 for each label. In addition, to preprocess the dataset, ID and time information that were not related to the classification among the features were deleted. In addition, the NaN and string data were preprocessed with zero padding and one-hot encoding, respectively.

In this study, greedy and random models were implemented and evaluated as comparative models of the SFSM. The greedy model [13] performs an exhaustive search that targets all the features. The greedy model exhibits the most efficient detection performance because it tests the performance of all possible feature combinations and selects the set with the best performance. However, this model possesses an extremely large search complexity, 2^n. A random model [13], which performs a random search that targets all features, was implemented randomly selecting r features from the entire feature set. The detection accuracy, latency, and complexity were used as evaluation metrics to verify the SFSM performance. Detection accuracy is an evaluation metric that demonstrates whether malicious traffic has been accurately classified, and it was calculated using Eq. 1.

$$Detection\,accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Latency measures the total time from the dataset preprocessing to the learning process. The complexity was derived by defining an equation that calculates the amount of computation based on the data size. Complexity was determined considering the complexity consumed in the feature importance calculation and the complexity consumed in the exhaustive search process, as shown in Eq. 2 [24]:

$$Complexity = k\mathrm{C}_{\mathrm{r}} \times \mathrm{l} + \mathrm{n} \times \mathrm{l} \times \mathrm{s} \tag{2}$$

where k, r, l, n, and s are the number of exhaustive search target features, number of features used when learning is performed, total number of data points, total number of features, and data sampling ratio, respectively.

In this experiment, the performances of the three evaluation metrics were evaluated while increasing r, and the gradient for selecting the k and s parameters of the SFSM was set to 0.005.

4.2 Evaluation results and analysis

In this section, to demonstrate the feasibility of the proposed SFSM, its performance is compared with that of the comparative, greedy, and random models in terms of detection accuracy, latency, and complexity. In addition, to prove that the k and s parameters can be optimized, the detection performance was confirmed by increasing the number of features k and the data sampling ratios. The detection performance as the number of features was increased for data sampling ratios of 0.2, 0.5, and 0.8 was demonstrated. When the data sampling ratio was increased, the detection performance was confirmed when the number of features k was 3, 5, or 7. Fig. 3 shows the detection accuracy performance evaluation based on the number of features and data sampling ratio.

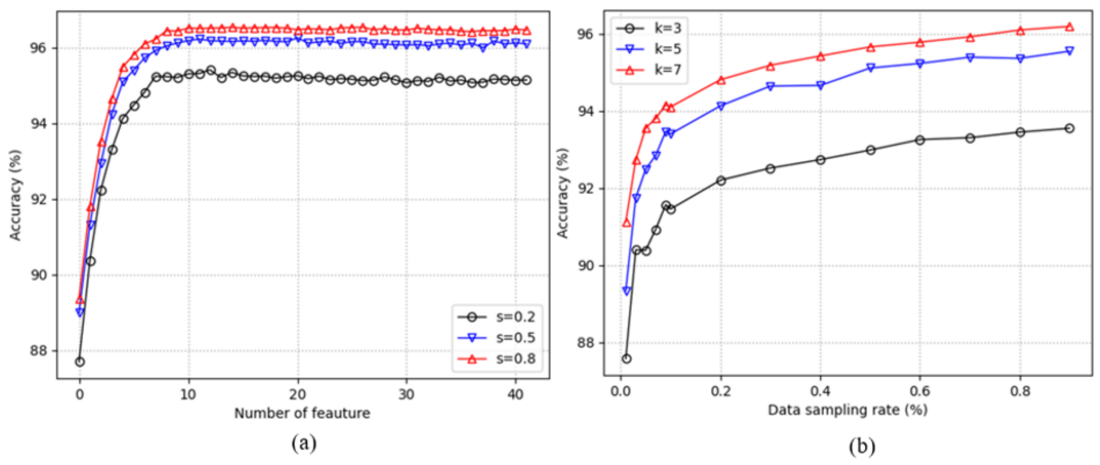

Figure 3: Detection accuracy performance by parameters: (a) Number of feature; (b) Data sampling ratio.

As shown in Fig. 3, when the number of features and the data sampling ratio were increased, the rate of increase in the detection performance plateaued at a specific point. In addition, when the number of features was increased, the rate of increase in detection performance plateaued at a point where all three seconds values were similar. When the data sampling ratio increased, the growth rate of the detection performance slope plateaued at the point where all three k values were identical. This study determined the number of features and the data sampling ratio based on a slope value of less than 0.005.

Accordingly, the proposed SFSM optimizes the selection of k and s values when the slope of the detection performance growth rate plateaus as the k and s values increase. Fig. 4 shows a comparison of the performances of the SFSM and conventional models.

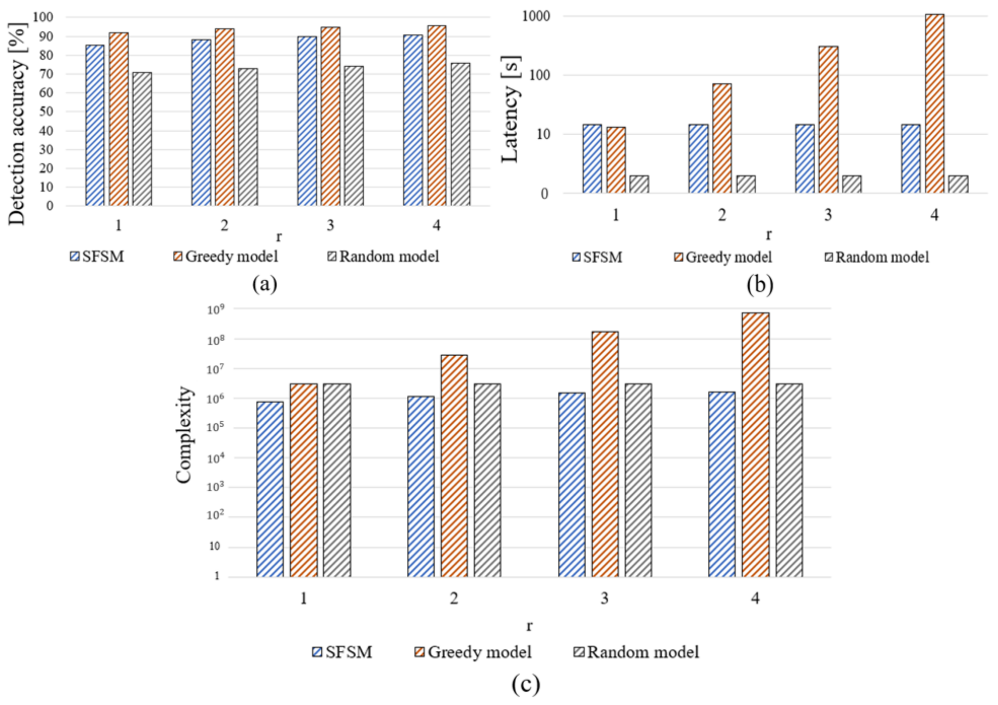

Figure 4: Performance and complexity evaluation results of SFSM compared to conventional models: (a) Detection accuracy; (b) Latency; (c) Complexity.

The detection accuracy indicated good results using the greedy, SFSM, and random models in order. Due to the nature of the greedy model, which conducts an exhaustive search across all features, the highest accuracy, in the range 90–95%, is obtained. In contrast, the random model, which randomly selects features without considering their importance, proved to be the least efficient in terms of accuracy. Furthermore, the greedy model is the least efficient in terms of latency and complexity, which are two crucial evaluation metrics for IoT environments. Specifically, the latency of the greedy model was approximately 69 times higher than that of the SFSM, whereas the complexity was approximately 313 times more inefficient than that of the SFSM.

Based on these findings, the SFSM demonstrated that it could minimize the tradeoff between these two metrics more efficiently compared to the greedy model. This was achieved by significantly reducing latency and complexity, which are essential considerations in the IoT environment, while incurring only a marginal loss in detection performance.

5 Conclusion

Machine-learning technology is widely used to identify and detect malicious traffic in IoT network environments. However, the presence of irrelevant and overlapping features result in an increase in the computational complexity and degrade machine-learning performance during the learning process. Therefore, to address the complexity problem and enhance the detection performance of machine learning, a feature selection approach that eliminates less relevant or overlapping features is crucial. In this study, we propose an SFSM, which reduces the complexity of the feature selection process for effective machine learning in IoT environments. During this process, the SFSM determines the number of features to be considered and identifies suboptimal sampling rate parameters. It utilizes an optimized feature set for learning through an exhaustive search. The effectiveness of the SFSM in detecting malicious traffic in resource-constrained IoT environments has been demonstrated. In this study, a single dataset with a fixed gradient value is used. Future works may showcase the performance of the SFSM with a more diverse range of datasets and optimize its capabilities by varying the gradients for selecting the k and s parameters.

Acknowledgments

This work was partly supported by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korean Government(MOTIE) (P0008703, The Competency Development Program for Industry Specialists), and MSIT under the ICAN (ICT Challenge and Advanced Network of HRD) program (No. IITP-2022-RS-2022-00156310) supervised by the Institute of Information & Communication Technology Planning and Evaluation (IITP).

References

[1] Insider Inc. The internet of things 2020: Here's what over 400 iot decision-makers say about the future of enterprise connectivity and how iot companies can use it to grow revenue. https://www.businessinsider.com/internet-of-things-report, 2020.

[2] Yun S. W., Park N. E., and Lee I. G. Wake-up security: Effective security improvement mechanism for low power internet of things. *Intelligent Automation Soft Computing*, 37:2897–2917, 2023.

[3] Bagaa M., Taleb T., Bernabe J. B., and Skarmeta A. A machine learning security framework for iot systems. *IEEE Access*, 8:114066–114077, 2020.

[4] Mohanta B. K., Jena D., Satapathy U., and Patnaik S. Survey on iot security: Challenges and solution using machine learning, artificial intelligence and blockchain technology. *Internet of Things*, 11, 2020.

[5] Ngo Q. D., Nguyen H. T., Le V. H., and Nguyen D. H. A survey of iot malware and detection methods based on static features. *ICT Express*, 6:280–286, 2020.

[6] Khraisat A., Gondal I., Vamplew P., and Kamruzzaman J. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 20, 2019.

[7] Al amri R., Murugesan R. K., Man M., Abdulateef A. F., Al-Sharaf M. A., and Alkahtani A. A. A review of machine learning and deep learning techniques for anomaly detection in iot data. *Applied Sciences*, 11, 2021.

[8] Mendonça R. V., Silva J. C., Rosa R. L., Saadi M., Rodriguez D. Z., and Farouk A. A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. *Expert Systems*, 39(5), 2022.

[9] Shafiq M., Tian Z., Bashir A. K., Du X., and Guizani M. Iot malicious traffic identification using wrapper-based feature selection mechanisms. *Computers Security*, 97, 2020.

[10] Sun G., Li J., Dai J., Song Z., and Lang F. Feature selection for iot based on maximal information coefficient. *Future Generation Computer Systems*, 89:606–616, 2020.

[11] Ferrara P., Mandal A. K., Cortesi A., and Spoto F. Static analysis for discovering iot vulnerabilities. *International Journal on Software Tools for Technology Transfer*, 23:71–88, 2020.

[12] Liu Z., Zhang L., Ni Q., Chen J., Wang R., Li Y., and He Y. An integrated architecture for iot malware analysis and detection. *IoT as a Service. 4th EAI International Conference (IoTaaS 2018)*, 271, 2018.

[13] Nazir A. and Khan R. A. A novel combinatorial optimization based feature selection method for network intrusion detection. *Computers Security*, 102, 2021.

[14] Abawajy J., Darem A., and Alhashmi A. A. Feature subset selection for malware detection in smart iot platforms. *Sensors*, 21(4), 2021.

[15] Alomari E. S., Nuiaa R. R., Alyasseri Z. A. A., Mohammed H. J., Sani N. S., Esa M. I., and Musawi B. A. Malware detection using deep learning and correlation-based feature selection. *Symmetry*, 15(1), 2023.

[16] Alhanahnah M., Lin Q., Yan Q., Zhang N., and Chen Z. Efficient signature generation for classifying cross-architecture iot malware. *2018 IEEE Conference on Communications and Network Security (CNS)*, 2018.

[17] Vijayanand R. and Devaraj D. A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network. *IEEE Access*, 8, 2020.

[18] Mahindru A. and Sangal A. L. Fsdroid:- a feature selection technique to detect malware from android using machine learning techniques. *Multimedia Tools and Applications*, 80:13271–13323, 2021.

[19] Taradeh M., Mafarja M., Heidari A. A., Faris H., Aljarah I., Mirjalili S., and Fujita H. An evolutionary gravitational search-based feature selection. *Information Sciences*, 497:219–239, 2019.

[20] Al-Tashi Q., Kadir S. J. A., Rais H. M., Mirjalili S., and Alhussian H. Binary optimization using hybrid grey wolf optimization for feature selection. *IEEE Access*, 7:39496–39508, 2019.

[21] Arora S., Singh H., Sharma M., Sharma S., and Anand P. A new hybrid algorithm based on grey wolf optimization and crow search algorithm for unconstrained function optimization and feature selection. *IEEE Access*, 7:26343–26361, 2019.

[22] Altmann A., Toloşi L., Sander O., and Lengauer T. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26:1340–1347, 2010.

[23] Moustafa N. and Slay J. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *2015 Military Communications and Information Systems Conference (MilCIS)*, page 1–6, 2015.

[24] Jeon S. E., Oh Y. S., Kil Y. S., Lee Y. J., and Lee I. G. Two-step feature selection technique for secure and lightweight internet of things. *ICCCN (The 32nd International Conference on Computer Communication and Networks)*, 2023.