A Comparison of Feature Selection and Feature Extraction in Network Intrusion Detection Systems

Tuan-Cuong Vuong, Hung Tran, Mai Xuan Trang, Vu-Duc Ngo, and Thien Van Luong

***Abstract*—Internet of Things (IoT) has been playing an important role in many sectors, such as smart cities, smart agriculture, smart healthcare, and smart manufacturing. However, IoT devices are vulnerable to cyber-attacks, which may result in security breaches and data leakages. To effectively prevent these attacks, a variety of machine learning-based network intrusion detection methods for IoT networks have been developed, which often rely on either feature extraction or feature selection techniques for reducing the dimension of input data before being fed to machine learning models. This aims to make the detection complexity low enough for real-time operations, which is particularly vital in intrusion detection systems. This paper provides a comprehensive comparison between these two methods in terms of various performance metrics, namely, precision rate, recall rate, detection accuracy, as well as runtime complexity, in the presence of UNSW-NB15 dataset. Note that such a comparison between feature selection and feature extraction methods has been overlooked in the literature. Furthermore, based on this comparison, we provide a useful guideline on selecting a suitable intrusion detection type for each specific scenario.**

***Index Terms*—Intrusion detection, UNSW-NB15, feature selection, feature extraction, PCA, machine learning, internet of things.**

I. Introduction

In recent years, the Internet of Things (IoT) has seen explosive expansion in industry-specific applications [1], [2] such as healthcare, smart homes, smart cities, smart energy, smart agriculture, and logistics. These IoT systems are made up of several interconnected sensors, actuators, and diverse network-enabled devices [3] that communicate various sorts of data through both public and private networks. By 2030, it is forecast that there will be about 500 billion IoT devices connected to the internet [3]. It is crucial to consider cyber-security seriously since the IoT has evolved into the engine of the current industrial revolution and the system for gathering real-time dependent data. As a result, a Network Intrusion Detection System (NIDS) that can identify present, potential, and future attacks is needed to protect IoT networks and systems built on them.

By examining the patterns of data traffic in the network, intrusion detection systems (IDS) can identify attacks. To lower the computational cost of processing raw data in IDS, feature extraction is necessary. It describes a method of converting a data space into a feature space that has the same (or a reduced) dimension than the original data space. It is well known that principal component analysis (PCA) is an essential technique in data compression and feature extraction, and it has been applied to the field of IDS [4]. Kuang *et al.* [5] proposed a novel intrusion a novel intrusion detection (ID) approach combining support vector machine (SVM) and kernel principal component analysis (KPCA) to enhance the detection precision for low-frequent attacks and detection stability. Pajouh *et al.* [6] proposed a network-based model for ID, hereafter referred to as Two-layer Dimension Reduction, including Linear Discriminant Analysis (LDA) combining PCA and a Two-tier Classification module utilizing Naive Bayes and K-Nearest Neighbor. Their proposed model is designed to detect and deter malicious activities such as User to Root (U2R) and Remote to Local (R2L) attacks, its results outperforming similar models of detection rates for both low-frequency and conventional attacks. With the strong development of Deep Learning, in order to maximize the efficiency of feature extraction and create a powerful intrusion detection system, Autoencoder was used [7]. Yan *et al.* [8] proposed using the sparse autoencoder (SSAE) to extract high-level feature representations of intrusive behavior information. Its results reach or even exceed the average detection level of conventional machine learning classifiers. However, it cannot effectively detect R2L and U2R low-frequency attack samples, that is, it cannot overcome the adverse effects caused by imbalanced data distribution. In addition, due to the high complexity of the algorithm, which leads to expensive training time, many methods seem to reduce the complexity of the algorithm. Charte *et al.* [9] proposed an autoencoder-based approach to complexity reduction, using class labels in order to inform the loss function about the adequacy of the generated variables. Dao *et al.* [10] proposed a thin-but-powerful NIDS scheme that combines a stacked autoencoder with a network pruning technique. The system's result demonstrates that our slender NIDS architecture is well-suited to edge devices with minimal memory and compute requirements. Currently, researchers are still trying to find ways to optimize the Autoencoder algorithm.

Feature selection is a form of search in the training data. In order to increase the classification accuracy of a learning classifier, it chooses a subset of input features from a total of original input features in the training data. Wrapper and filter approaches can be used to classify feature selection algorithms [11]. Filter methods select features that are independent of one another and are heavily dependent on the

Tuan-Cuong Vuong, Hung Tran, Mai Xuan Trang, and Thien Van Luong are with the Faculty of Computer Science, Phenikaa University, Hanoi 12116, Vietnam (e-mail: 21011490@st.phenikaa-uni.edu.vn, {hung.tran, trang.maixuan, thien.luongvan}@phenikaa-uni.edu.vn).

V.-D. Ngo is with the School of Electrical and Electronics Engineering, Hanoi University of Science and Technology, Hanoi 11657, Vietnam, (email: duc.ngovu@hust.edu.vn).

output, while wrapper techniques strive to maximize some specified criteria with respect to the feature set as part of the selection process. The entire feature space is searched for and potential subsets are evaluated using a search algorithm in feature selection methods. They need a feature goodness measure that assigns grades to any feature subset in order to assess these subsets. Overall, a feature is useful if it contributes to the output but does not duplicate other relevant features. The relationship between two features may serve as a measure of feature goodness. Amiri *et al.* proposed two methods: linear correlation-based feature selection (LCFS) and modified mutual information-based feature selection (MMIFS) to build a lightweight IDS [12]. In [13], Ambusaidi *et al.* proposed a new filter-based feature selection method, namely Flexible Mutual Information Feature Selection (FMIFS). FMIFS is an improvement over MMIFS.

Although both feature selection and feature extraction methods have been widely used for reducing data dimensionality in NIDS, a comprehensive comparison between them has been overlooked in the literature. Our paper appears to address this gap. In particular, we first provide an overview of NIDS, with a focus on the phase of feature reduction, where feature extraction with PCA and feature selection with correlation matrix are two promising candidates for realistic low-latency operations of NIDS. Then, using the UNSW-15 dataset, we thoroughly compare the detection performance (precision, recall, F1-score) as well as runtime complexity (training time and inference time) of these two methods when the same number of features are selected or extracted. Thanks to this comparison, we provide a useful guideline on choosing a suitable intrusion detection technique for each specific scenario.

The rest of this paper is organized as follows. Section II discusses machine learning-based network intrusion detection methods for IoT networks. The overview of UNSW-NB15 dataset and data pre-processing are explained in Section III. Section IV provides the experimental results and discussion. Finally, Section V concludes this paper.

II. Machine Learning-based Network Intrusion Detection Methods

In this section, we describe an overview of a network intrusion detection system (NIDS) based on machine learning, followed by details on the two major ML-based detection methods, namely feature selection and feature extraction.

A. Overview of NIDS

An NIDS consists of three major components, namely data pre-processing, feature reduction, and attack classification, as illustrated in Fig. 1. In particular, in the first phase, the raw data is denoted as the dataframe $\mathbf{Z}$, whose features may include unexpected or non-numeric values, such as null or nominal. $\mathbf{Z}$ is pre-processed in order to either replace these unexpected values with valid ones or transform them to the numeric format using one-hot encoding. Several features that do not affect detection performance, such as the source IP address and the source port number, are dropped. Furthermore, depending on the classifier we use for identifying attacks, we may use the

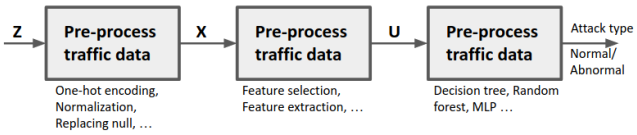

Fig. 1. Block diagram of an network intrusion detection system.

normalization technique, for example, to constrain the values of all features, i.e., the elements of the output vector of the first phase $\mathbf{X}$ in Fig. 1 to vary only from 0 to 1. We will discuss this in detail in Section III, where we present UNSW-NB15 dataset.

As such, after the first phase, the pre-processed data $\mathbf{X} \in \mathbb{R}^{D\times N}$ is likely to have much more features than the original data $\mathbf{Z}$, particularly due to the use of one-hot encoding, where D is the number of dimensions, or equivalently, the number of features of $\mathbf{X}$, and N is the number of data samples. For example, when UNSW-NB15 dataset is used, the dimension of data increases from 45 to 200, which is too large for classification techniques to quickly recognize the attack type. In order to address this fundamental issue, in the second phase, we need to reduce the number of features that will be used for the attack classification phase (the last phase in Fig 1). For this, two feature reduction methods called feature selection and feature extraction are widely used to either selected or extracted a small number of most important features from pre-processed traffic data. This procedure also helps to remove a large amount of unnecessary features, which not also increase the complexity of NIDS, but also degrade its detection performance, as will be illustrated in experimental results in Section IV. Herein, the output data of the feature reduction block is denoted as vector $\mathbf{U} \in \mathbb{R}^{K\times N}$ in Fig. 1, which is expected to achieve much lower dimensions than $\mathbf{X}$, i.e., $K \ll D$, while retaining its most important information.

Finally, in the third phase of NIDS, a number of multiclass classification approaches based on machine learning, such as decision tree, random forest and multilayer perception neural networks, are employed to detect the attack type. Relying on attack detection results, the system administrators can promptly make a decision to prevent malicious activities, ensuring the security of IoT networks. Here, note that the detection performance and latency of a NIDS strongly depend on which classifier and which feature reduction method it employs. Therefore, in this contribution, we comprehensively investigate detection performance (in terms of recall, precision, F1-score) and latency (in terms of training time and inference time) of different detection methods in presence of both feature selection and feature extraction as well as different machine learning classifiers. We also focus more on the comparison between these two feature reduction methods, which will be described in detail in the following subsections.

B. Feature selection

There are a number of feature selection techniques used in intrusion detection, namely information gain (IG) [14] and feature correlation [15], [16]. In this work, we focus on using feature correlation for selecting important features, since this

method has been shown to achieve competitive performance and complexity compared to its IG counterpart. Using this correlation-based method, we aim to select features that are most correlated to other features based on the correlation matrix calculated from the training dataset. More specifically, the correlation coefficient between feature Ω_1 and Ω_2 is calculated based on the numeric pre-processed training dataset $\mathbf{X}$ as follows:

$$\mathcal{C}_{\Omega_1,\Omega_2} = \frac{\sum_{i=1}^{N}(\alpha_i - E_{\Omega_1})(\beta_i - E_{\Omega_2})}{\sqrt{\sum_{i=1}^{N}(\alpha_i - E_{\Omega_1})^2}\cdot\sqrt{\sum_{i=1}^{N}(\beta_i - E_{\Omega_2})^2}}, \quad (1)$$

where α_i and β_i are the values of these two features, $E_{\Omega_1} = \sum_{i=1}^{N}\alpha_i/N$ and $E_{\Omega_2} = \sum_{i=1}^{N}\beta_i/N$ are their means over N training data samples. By doing this, we obtain a $D \times D$ correlation matrix $\mathbf{C}$, whose elements are given by $c_{ij} = \mathcal{C}_{\Omega_i,\Omega_j}$ for $i, j = 1, 2, ..., D$. The average correlation of feature Ω_i to other features is computed as follows:

$$C_i = \frac{\sum_{j=1}^{D} c_{ij}}{D}, \quad (2)$$

where $c_{ii} = 1$ for $j = i$ and $c_{ij} \in [-1; 1]$ for $j \neq i$. Note that the self-correlation coefficient c_{ii} does not affect selection results, since it contributes the same amount to all C_i for $i = 1, 2, ..., D$. Then, using a suitable threshold, as will be detailed in Section IV, we are able to select K most important features corresponding to K largest elements C_i.

It is worth noting that we only need to calculate such feature correlation in the training phase, while in the testing phase, we simply pick up K features from the high-dimensional data $\mathbf{X}$ to form the reduced-dimensional data $\mathbf{U}$ in Fig. 1. This does not require much computational resource when compared with the feature extraction method, which is presented next.

C. Feature extraction

Principal component analysis (PCA) and autoencoder (AE) [17] are the two major feature extraction methods used in the NIDS. Different from feature selection, whose selected features are identical to those appearing in the original data, these feature extraction techniques compress the high-dimensional data $\mathbf{X}$ into the low-dimensional data $\mathbf{U}$ using either a projection matrix or an AE-based neural network learned from training dataset. Since the AE approach usually suffers from high computational complexity of a deep neural network (DNN), leading to higher latency than the PCA, in this work, we concentrate on the PCA-based feature extraction approach in order to fulfill a strict requirement on the latency of the NIDS for promptly preventing severe cyber attacks.

In what follows, we introduce the procedure of producing the $D \times K$ projection matrix $\mathbf{W}$ in the training phase, and how to utilize this matrix in the testing phase. In particular, based on the pre-processed training data $\mathbf{X}$ of N samples, we normalize it by subtracting all samples of $\mathbf{X}$ by its mean over all training samples, i.e., the normalized data is given as follows: $\hat{\mathbf{X}} = \mathbf{X} - \bar{\mathbf{X}}$, where $\bar{\mathbf{X}}$ is the mean vector. Then, we compute the $D \times D$ covariance matrix of training data as follows: $\mathbf{R} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$. Based on this, we determine its eigenvalues and eigenvectors, from which, we select K eigenvectors corresponding to K largest eigenvalues for constructing the $D \times K$ projection matrix $\mathbf{W}$. Herein, these K eigenvectors are regarded as the principal components that create a subspace, which is expected to be significantly close to the normalized high-dimensional data $\hat{\mathbf{X}}$. Finally, the compressed data is determined by $\mathbf{U} = \mathbf{W}^T\hat{\mathbf{X}}$, which now has the size of $K \times N$ instead of $D \times N$ of the original data.

In the testing phase, for each new data point $\mathbf{x}_i \in \mathbb{R}^D$, its dimensions are reduced using PCA according to $\mathbf{u}_i = \mathbf{W}^T(\mathbf{x}_i - \bar{\mathbf{X}})$. This indicates that the output of the training phase of PCA includes both the projection matrix $\mathbf{W}$ and the mean vector of all training samples $\bar{\mathbf{X}}$. It should be noted that such projection matrix calculation would be computationally expensive, particularly when D and K are large.

III. Overview of UNSW-NB15 dataset

We now present some key information about UNSW-NB15 dataset, which is used in our experiments in Section IV to compare between feature selection and feature extraction. Then, the data pre-processing for this dataset is also discussed.

A. Key information of UNSW-NB15 dataset

UNSW-NB15 dataset was first introduced in [18], which offers better real modern normal and abnormal synthetical network traffic compared with the previous NIDS dataset such as NSLKDD and KDDCUP99 [19]. A total of 2.5 million records of data are included in the UNSW-NB15 dataset, which includes one normal class and nine attack classes: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms. Flow features, basic features, content features, time features, additional generated features, and labeled features are six feature groups, which consist of a total of 49 features in the original data. Additionally, in this work, we use a 10% cleaned dataset of UNSW-NB15, which includes a training set of 175,341 records and a test set of 82,332 records. There are a few minority classes with proportions of less than 2%, including Analysis, Backdoor, Shellcode, and Worms (see Fig. 2 and Fig. 3). In the 10% dataset, some unrelevant features were removed (see Subsection III-B), and the number of features was reduced to 45, including 41 numerical features and 4 nominal features.

B. Pre-processing dataset

Initially, the dataset has 45 features, including 41 numerical features and 4 nominal features. We remove the *id* feature in numerical features, since its does not affect the detection performance. The remaining 4 nominal features include ['attack_cat', 'proto', 'service', 'state']. The *attack_cat* feature contains the names of attack categories so we will remove this feature. For the *service* feature, if it contain nulls, we treats those null values as 'other' type of service.

One-hot encoding is used for transforming nominal features to numerical values. As a result, the number of features $\mathbf{Z}$ in the training set will increase from 45 features to 200 features $\mathbf{U}$. Therefore, it is necessary to reduce such a large

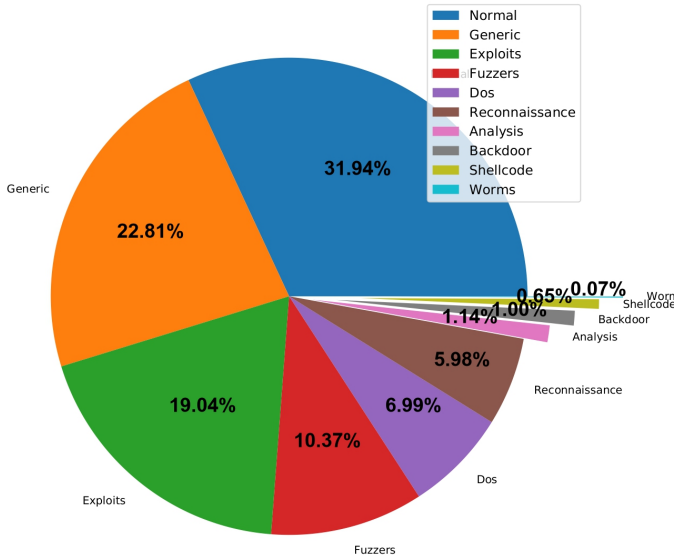

Fig. 2. Proportions of 10 classes in training dataset of UNSW-NB15.

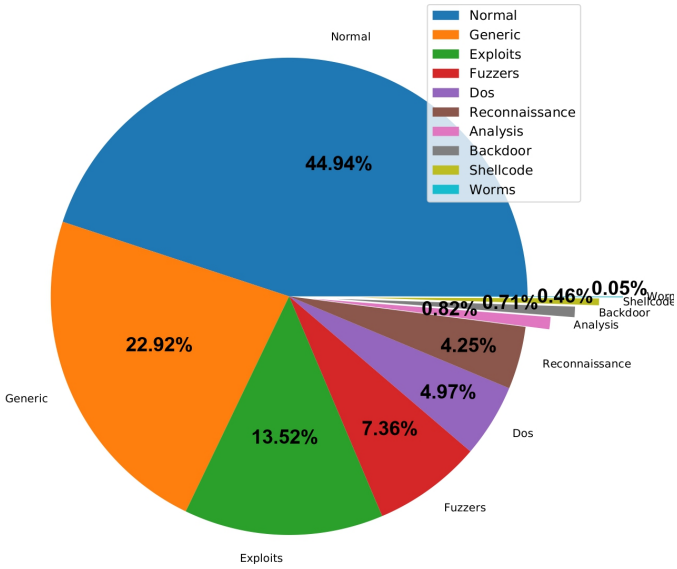

Fig. 3. Proportions of 10 classes in testing dataset of UNSW-NB15.

number of features to a few most important features, which helps to reduce the complexity of machine learning models in the classification phase. When feature extraction is used, we normalize the input feature with the minimum-maximum method [20] to improve the classification accuracy, while we do not use data normalization for feature selection since it does not improve performance.

IV. Experimental Results and Discussion

We now present extensive experimental results for investigating the performance of the NIDS using both feature selection and feature extraction methods described in Section II, in combination with a range of machine learning-based classification models. More particularly, the performance metrics used for comparison include recall (R), precision (P), F1-score, training time and inference time, which will be explained in detail in Subsection IV-A. We also provide a number of confusion matrices in order to provide an insight into different intrusion detection methods considered in this work. Last but not least, based on our extensive investigation and comparison between feature selection and feature extraction, we provide a helpful guideline on how to choose an appropriate intrusion detection technique for each specific scenario.

TABLE I
Hardware and Environment Specification

Unit	Description
Processor	Intel Core i5-10400F (2.66 Hz, 6 cores 12 threats, 12MB Cache, 65W)
RAM	16GB
GPU	GiGabyte GTX 1650 OC-4G
Operating System	Ubuntu 20.04.4 LTS
Packages	Numpy, Matplotlib, Pandas, Scipy, Scikit-learn, Scikit-plot and Time

TABLE II
Threshold Setting and Features Selected

Threshold	Number	Features Selected
0.0137	8	'dur', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'sloss', 'dloss','ct_state_ttl'
0.011	16	'dur', 'spkts', 'dpkts', 'sbytes', 'dbytes', 'sloss', 'dloss', 'dinpkt', 'sjit', 'djit', 'tcprtt', 'synack', 'ackdat', 'response_body_len', 'ct_state_ttl', 'proto_icmp'

A. Implementation setting

1) Computer configuration: The configuration of our computer, its operation system as well as a range of Python packages used for implementing intrusion detection algorithms in this paper are detailed in Tab. I.

2) Evaluation Metrics: We consider the following performance metrics: recall, precision, F1-score, as well as training time and inference time, which are defined as follows.

As shown in Fig. 1, two methods go through the same preprocessing data, so the training time consists of the training time of classification models and the time duration consumed by Feature Reduction in training (FR_train), as follows:

$$\text{Training time} = \text{time}_{train} + \text{time}_{FR\_train}, \quad (3)$$

The inference time consists of the prediction time of classification models and the testing time of classification models.

$$\text{Inference time} = \text{time}_{predict} + \text{time}_{test}, \quad (4)$$

3) Classification models: We use 5 machine learning models: Decision Tree, Random Forest having max_depth = 5, K-nearest Neighbors (n_neighbors = 5), Multi-layer Perceptron (MLP) (max_iter = 100, hidden_layer_sizes = 200), and Bernouli Naive Bayes to do the multiclass classification task. These models are all imported from the Scikit-learn library.

For feature selection, in Tab. II, we provide the lists of 8 and 16 selected features, as well as the corresponding thresholds of the average correlation used to achieve that desired numbers of selected features.

TABLE III
DETECTION PERFORMANCE COMPARISON BETWEEN FEATURE SELECTION AND FEATURE EXTRACTION: 8 FEATURES

Models	Feature Extraction					Feature Selection				
	P	R	F1	training (s)	inference (μs)	P	R	F1	training (s)	inference (μs)
Decision Tree	76.43	69.36	72.72	27.49	**4.42**	**80.18**	**76.62**	**78.36**	14.65	**0.13**
Random Forest	78.32	66.74	72.07	34.13	20.66	78.82	68.65	73.38	18.38	5.48
KNeighbors	77.95	**72.86**	**75.32**	23.5	41.78	80.27	73.9	76.96	14.8	50.15
MLP	**79.83**	69.78	74.47	1180.75	47.77	65.14	68.96	66.99	591.4	42.79
Naive Bayes	65.73	51.77	57.92	**22.77**	4.52	43.47	59.67	50.3	**14.57**	1.08

TABLE IV
DETECTION PERFORMANCE COMPARISON BETWEEN FEATURE SELECTION AND FEATURE EXTRACTION: 16 FEATURES

Models	Feature Extraction					Feature Selection				
	P	R	F1	training (s)	inference (μs)	P	R	F1	training (s)	inference (μs)
Decision Tree	77.33	70.11	73.55	37.68	**5.04**	**79.59**	**75.78**	**77.64**	15.17	**0.19**
Random Forest	78.36	**66.71**	72.07	53.12	21.27	80.03	68.02	73.54	22.17	5.69
KNeighbors	77.56	72.03	74.69	34.94	1405.43	78.79	63.91	70.58	14.72	1396.85
MLP	**79.44**	71.97	**75.52**	1428.97	47.51	64.42	69.01	66.63	895.37	49.51
Naive Bayes	74.57	60.59	66.87	**34.75**	6.34	47.28	59.75	52.79	**14.47**	1.09

B. Detection and Runtime Performance

We investigate the detection performance and runtime of feature selection and feature extraction when using multiclass classification in Tab. III and IV for 8 and 16 selected/extracted features, respectively. In these tables, the best values (i.e. the maximum values for precision, recall, and F1-score, and the minimum values for training and inference times at each column of the tables) are highlighted in bold, especially the best values for both feature selection and feature extraction methods are highlighted both in bold and red color. The training time is measured in second (s), while the inference time (for each sample) is measured in millisecond (μs).

In terms of detection performance, it is shown from Tab. III and IV that when the number of reduced features (i.e. extracted or selected) K increases, the detection performance of feature extraction generally improves, while that of feature selection does not improve when we increase K from 8 to 16. In fact, the precision, recall and F1-score of feature selection even slightly degrade from Tab. III to Tab. IV. This phenomenon is understandable due to the fact that if the number of selected features gets larger, it is likely to have more noisy or unimportant features, which deteriorate the detection performance. For example, when Decision Tree is employed in Tab. III to achieve the lowest inference time, the F1-score of feature selection is 78.36%, which is higher than that of feature extraction with 75.32%. It is also shown from Tab. III and IV that when using feature selection, the Decision Tree classification always provides the best precision, recall as well as F1-score. Meanwhile, the feature extraction enjoys the KNeighbors classifier when $K = 8$, while Decision Tree is only its best classifier when K becomes larger, i.e., $K = 16$.

As for the runtime performance, Tab. III and IV demonstrate that both the training time and inference time of feature selection is lower than that of feature extraction. This is because of the fact that the feature extraction method requires additional computational resources when compressing the high-dimensional data into low-dimensional data, as explained in Section II-C, while the feature selection almost does require any computing resources when just picking up K out of D features. Actually, in Tab. III, the best inference time of feature selection is 0.13 μs, which is 34 times lower than that of feature extraction with 4.42 μs, where the Decision Tree classifier is the best choice for minimizing the inference time for both feature reduction methods. Again, Decision Tree is one of the best classifier for minimizing both training and inference times, in addition to the Naive Bayes classifier, which however does not achieve a good accuracy.

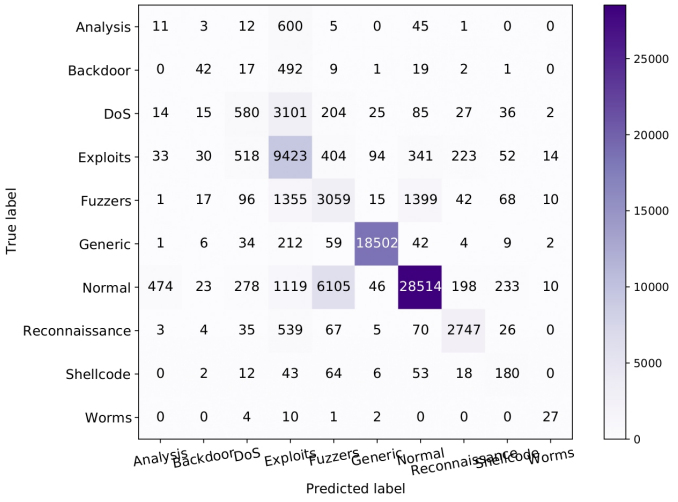

Fig. 4. Confusion Matrix based Decision Tree with Feature Selection (8 feature selected)

Based on the Decision Tree model, we now evaluate the multi-class classification ability of feature extraction and feature selection, using confusion matrix, as shown in Figs. 4, 5 and Figs. 6, 7 for 8 and 16 reduced features, respectively. In both cases where we reduce 8 features or 16 features, the results based on four figures show that Decision Tree model based on the feature selection method can correctly detect a

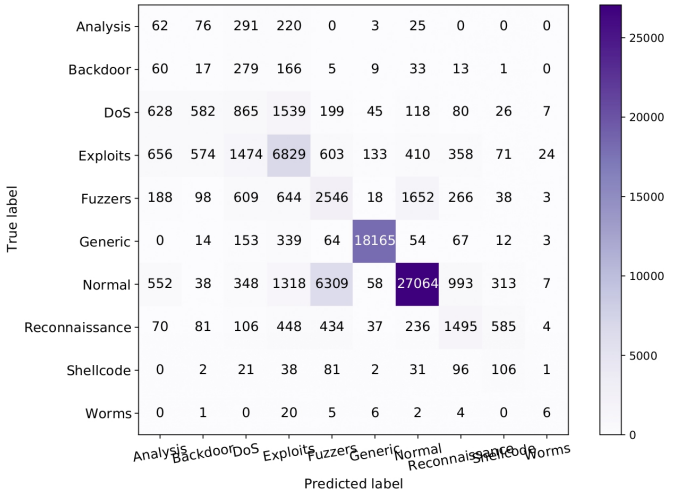

Fig. 5. Confusion Matrix based Decision Tree with Feature Extraction (8 feature extracted)

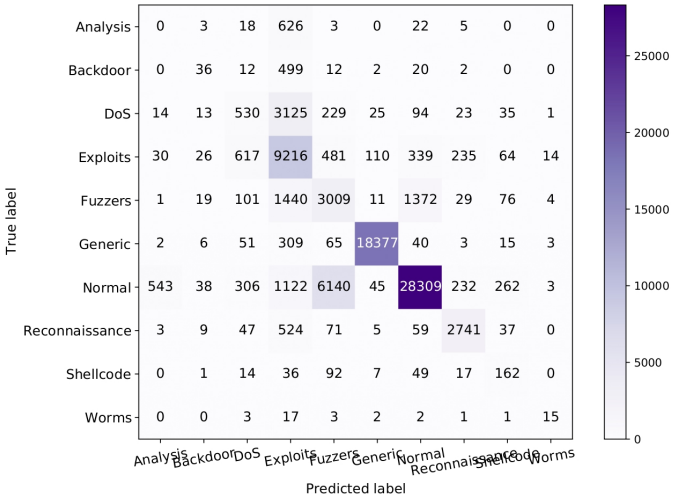

Fig. 6. Confusion Matrix based Decision Tree with Feature Selection (16 feature selected)

greater number of attacks compared with the feature extraction method. When the number of selected features increases, for the feature selection method, shown in Figs. 4 and Fig. 6, the number of detections of attack types is reduced slightly. In contrast, for the feature extraction method, shown in Figs. 5 and Fig. 7, the number of attack types detected has been increased. It is shown via Fig. 6 and Fig. 7 that when the Decision tree model based on feature selection method cannot detect the type of Analysis attack, but the Decision tree model based on feature extraction method can detect it. The reason for this problem lies in the fact that the number of features we choose is relatively large, but the number of Analysis attack types is only less than 1% (see Fig. 3), leading to the model tending to detect the remaining attack types.

In summary, considering multiclass classification for the NIDS, the feature selection method not only provides better detection performance but also lower training and inference time compared to its feature extraction counterpart, especially when the number of reduced features K is not too large. However, feature selection is not capable of detecting attacks with as little data as feature extraction can yield.

V. Conclusions

We have compared two intrusion detection methods, namely, feature selection and feature extraction, based on the UNSW-NB15 dataset. Feature selection not only achieves higher detection accuracy but also requires fewer training and inference times than feature extraction, especially when the number of reduced features is not too large. However, feature selection is not capable of detecting attacks as diverse as feature extraction can yield. In addition, the detection accuracy of feature selection significantly degrades when substantially increasing the number of reduced features, while that of feature extraction is slightly improved.

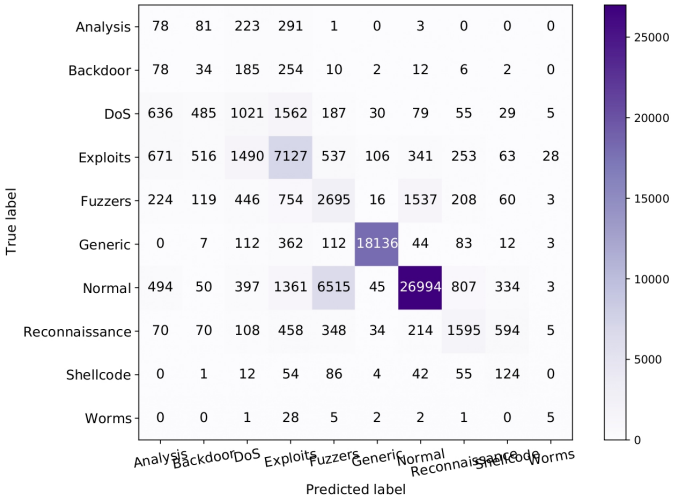

Fig. 7. Confusion Matrix based Decision Tree with Feature Extraction (16 feature extracted)

References

[1] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on industrial informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[3] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the internet of things (IoT) forensics: Challenges, approaches, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1191–1221, 2020.

[4] G. Liu, Z. Yi, and S. Yang, "A hierarchical intrusion detection model based on the pca neural networks," *Neurocomputing*, vol. 70, no. 7-9, pp. 1561–1568, 2007.

[5] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid kpca and svm with ga model for intrusion detection," *Applied Soft Computing*, vol. 18, pp. 178–184, 2014.

[6] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K.-K. R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 2, pp. 314–323, 2016.

[7] Y. N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, Firdaus, and Jasmir, "Automatic features extraction using autoencoder in intrusion detection system," in *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2018, pp. 219–224.
[8] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41 238–41 248, 2018.
[9] D. Charte, F. Charte, and F. Herrera, "Reducing data complexity using autoencoders with class-informed loss functions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
[10] T.-N. Dao and H. Lee, "Stacked autoencoder-based probabilistic feature extraction for on-device network intrusion detection," *IEEE Internet of Things Journal*, 2021.
[11] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.
[12] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1184–1199, 2011.
[13] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE transactions on computers*, vol. 65, no. 10, pp. 2986–2998, 2016.
[14] C. Lee and G. G. Lee, "Information gain and divergence-based feature selection for machine learning-based text categorization," *Information processing & management*, vol. 42, no. 1, pp. 155–165, 2006.
[15] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.
[16] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 856–863.
[17] T.-N. Dao and H. Lee, "Stacked autoencoder-based probabilistic feature extraction for on-device network intrusion detection," *IEEE Internet of Things Journal*, pp. 1–1, 2021.
[18] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.
[19] J. Gehrke, P. Ginsparg, and J. Kleinberg, "Overview of the 2003 kdd cup," *Acm Sigkdd Explorations Newsletter*, vol. 5, no. 2, pp. 149–151, 2003.
[20] S. B. Kotsiantis and et al., "Data preprocessing for supervised learning," 2006.