PRUNING-BASED INTRUSION DETECTION FOR MAXIMIZING THE TRAFFIC MANAGEMENT IN INTERNET OF THINGS

Thi-Nga Dao[1], *Manh-Hung Tran* [1], *Huu-Noi Nguyen*[2]

Abstract

This work considers the problem of maximizing the number of packets to be classified by the network security system in programmable switches in Internet of Things. With the purpose of developing a lightweight security method for programming switches with limited computing resource, we present a neural-network-based intrusion detection model that combines with a neuron pruning method to achieve low model complexity without significant sacrifice in accuracy. Then, we formulate an integer linear programming (ILP) problem that maximizes the amount of monitored traffic by all switches under requirements of classification accuracy and computing resources. The optimization problem is considered in two cases: using and not using the neuron pruning (NP)-based models to show the benefits of the proposed lightweight architecture. The evaluation results show that NP-based models allow switches to manage more data traffic while satisfying given requirements of accuracy and computing resources.

Index terms

Traffic management, intrusion detection, neuron pruning

1. Introduction

With the recent development of technologies such as low-power devices, data analytics, and processors, more and more Internet of things (IoT) devices are allowed to be connected to form a network in a variety of applications (i.e., smart home, smart agriculture, smart transportation, and surveillance system) [1], [2], [3], [4]. However, it is challenging to monitor such high volume of data traffic under the fact that the number of network attacks tends to increase over time. Therefore, there is a need for a network intrusion detection and classification system that can quickly detect network threats and take an appropriate action for detected attacks. This work aims to address the problem of maximizing the amount of data traffic to be classified by the security model.

[1] Faculty of Radio-Electronic Engineering, Le Quy Don Technical University, Hanoi, Vietnam
[2] Faculty of Information Technology, Le Quy Don Technical University, Hanoi, Vietnam

Some network security models require to transmit traffic to external devices for management, which leads to high detection time [5], [6]. To achieve the low detection latency, the classification model is usually implemented on edge devices (e.g., switches) that are distributed near IoT devices. One big challenge of implementing the classification function on edge devices is that the classification model should have a lightweight architecture with low model complexity since the edge devices are usually equipped with limited computing and memory resources. Note that high accurate classification models are constructed based on advanced machine learning techniques (e.g., neural network) with high model complexity. Therefore, a simplification method is needed to reduce the model complexity of the classification models [7], [8], [9], [10].

In this work, we consider a simple neuron pruning method [8] to build a traffic classification model with low complexity, thus making it suitable for edge devices with constrained computing resources. There are three steps to construct the neuron pruning (NP)-based traffic classification model: training the whole model, removing unimportant connections, re-training the pruned model. Note that the programming language (i.e., P4) for data plane only supports a limited set of arithmetic operations. Therefore, we use only supported operations by P4 for implementing the classification model. To evaluate the NP-based model, we measure and compare the classification accuracy and detection delay of the NP-based model with the fully-connected (FC) architecture. The detection delay is measured on a programmable switch.

Then, we introduce the integer linear programming (ILP) problem to maximize the number of packets to be managed by all participating switches in the network. Two main constraints are considered including accuracy requirement and computing resource. Specifically, the average classification accuracy of all switches should be greater than a given threshold value and the time used for traffic classification by each switch should be less than a threshold value based on available resources. Each switch independently assigns a suitable detection model to specific packets such that all constraints are satisfied and the number of monitored packets is maximized. To highlight the benefits of NP models, we consider two cases: using NP models and not using NP models with different accuracy and computing resource threshold values. Note that since the incoming data rate at a switch tends to vary over time, the switch needs to re-determine packet assignment whenever the data rate changes with a large value.

The main contributions of our work is listed as below.

1) First, we introduce the neuron-pruning-based intrusion detection and classification model with low complexity that can achieve low detection delay.
2) Then, we formulate an optimization problem to assign the right detection model to a specific incoming packet for the traffic management maximization under accuracy and computing resource constraints.
3) We evaluate and compare the performance of the NP-based intrusion detection model with the FC architecture to prove the lightweight architecture.
4) The traffic management problem is considered under two cases (with and without NP models) given various requirements. The experimental results show that using

NP models always produce a better traffic management approach than not using NP models.

The rest of the paper is organized as below. Section 2 presents assumption of the network system and Section 3 introduces the detail architecture of the NP-based intrusion classification model. Then, the optimization problem is presented in Section 4. To prove the advantages of the NP-based model, Section 5 shows the performance evaluation of the NP-based model and counterpart algorithms on the data plane. Finally, we conclude our work in Section 6.

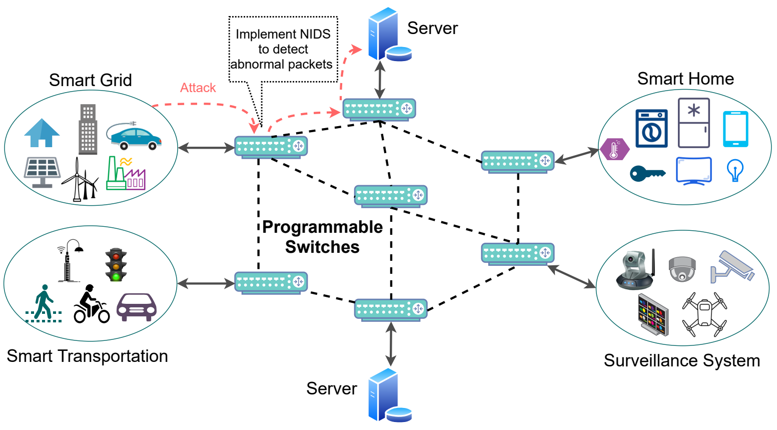

Fig. 1. The overview of network intrusion detection system for IoT

2. Network System

Figure 1 presents IoT networks consisting of IoT devices from multiple applications (e.g., smart grid, smart transportation, smart home, smart healthcare, and surveillance system), servers, and switches that connect IoT devices and servers. Data traffic generated from IoT devices can be transmitted to servers via switches. Participating switches are connected using an arbitrary network topology such as bus, star, tree, ring, mesh. For example, the mesh topology is shown in Figure 1. Besides the forwarding function, these switches are in charge of managing data traffic and make sure that only normal data should be forwarded. If there is an attack from an IoT devices in smart grid toward a server in Figure 1, switches equipped with the network intrusion detection system (NIDS) detect abnormal packets and make an appropriate actions to block these packets from accessing the server.

Assume that IoT devices generate data periodically or whenever an event (e.g., fire, gas leakage, and smoke) is detected. Hence, the data rate may vary over time. We demonstrate the changes of traffic rate in the IoT Korea dataset [11] in Figure 2. Specifically, we measure the number of incoming packets per second. If a period lasts for one second, there are around 1,000 periods. Generally, the data rate changes significantly over time. For example, there are around 500 incoming packets per second

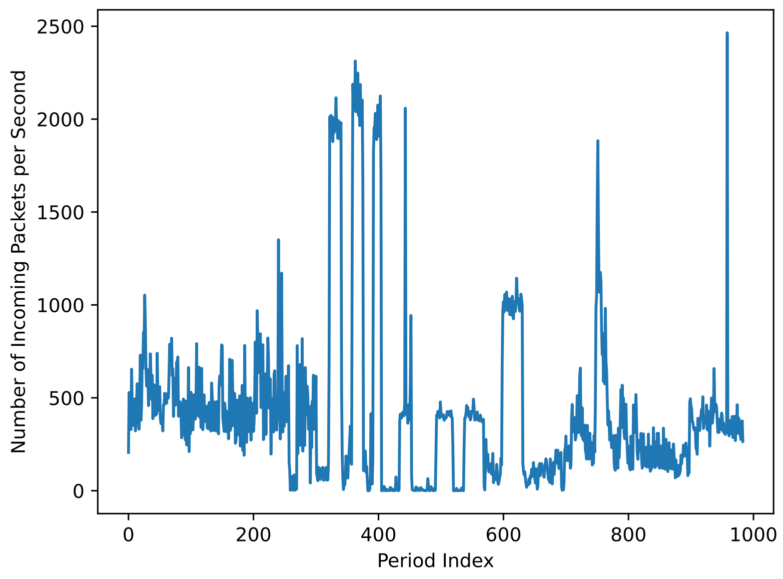

Fig. 2. Changes of traffic rate in the IoT Korea dataset

at the beginning and this number can increase up to 2,500 or reach nearly 0 at some points in the experimental duration. Since the incoming data rate at each switch varies over time, the detection model used by a switch should be adapted accordingly. For instance, when the data rate is low, the switch can use a complex detection model to achieve high accuracy. In contrast, a simpler prediction model with low complexity should be considered in cases when the data rate is high.

In order to quickly detect and respond to network threats, we execute the traffic classification model on programmable data plane. More specifically, data plane programmability allows us to easily add customized packet processing functions on edge devices, thus significantly reducing detection delay. There are multiple commercial programming edge devices including NetFPGA SUME developed by Digilent [12] and Intel Tofino2 [13]. When a data packet arrives at the input port, the switch makes a prediction on the traffic type of the packet. There are five different traffic labels: normal, reconnaissance, man-in-the-middle, denial-of-service, and botnet. Based on the prediction output, the switch can take an appropriate action for this packet, e.g., forwarding the packet, dropping the packet, or adding an alarm field in the packet header.

In our work, each programmable switch makes the decision on the detection model independently. There are multiple factors that should be taken into account: the incoming data rate, available computing resources, accuracy and detection time of each detection model. Note that there is a balance between the accuracy and detection time of the traffic classification model. Specifically, if using a model with high complexity, we can achieve high accuracy with a sacrifice in detection time and vice verse. Therefore, these factors should be considered carefully by participating switches.

In the following section, we introduce a lightweight detection and classification model with the support of a parameter trimming method. Then, an optimization problem for the traffic management maximization is proposed in Section 4 to select a suitable model given constraints on computing resources and average performance.

3. A Neuron Pruning-based Intrusion Detection and Classification Model

In this section, we introduce a timely and lightweight network intrusion detection architecture that can be suitable for programmable networking devices with limited computing resource. Recently, neural networks (NNs) have emerged as an advanced machine learning technique to learn a non-linear mapping from input features to output values. However, NNs suffer from the high model complexity, which leads to high detection delay.

To address the issue of large detection latency, we apply a neuron pruning technique that trims unnecessary connections of the model and only keeps salient weights. The reason of proposing pruning-based detection model is to provide the switch other options with low complexity for selection, especially for the cases of high data rate. The construction of the pruning-based network intrusion detection model is shown in Figure 3. The fully-connected architecture consists of three layers: input, hidden, and output. The ReLU activation function is used for the hidden layer since it only contains simple operations and can be easily implemented using the P4 programming language. Assume that z is the input of ReLU function, then the output of ReLU is given by:

$$ReLU(z) = max(0, z) \tag{1}$$

The softmax function is considered at the output layer. Now, the construction of the fully-connected model with one hidden layer is presented. We define x, h and y as the vectors for input, hidden and output units, respectively. Meanwhile, W_1 and b_1 denote the weight matrix and bias vector, respectively, that connect the input and hidden layers. Network parameters for the output layer are defined as W_2 and b_2. The fully-connected model is computed as below.

$$h = ReLU(W_1 x + b_1) \tag{2}$$

$$y = W_2 h + b_2 \tag{3}$$

There are three phases for the training procedure: learning the fully-connected model, pruning unnecessary connections, re-training the pruned network. In the first phase, parameters including weights and biases are trained by minimizing the entropy-based loss function L as follows.

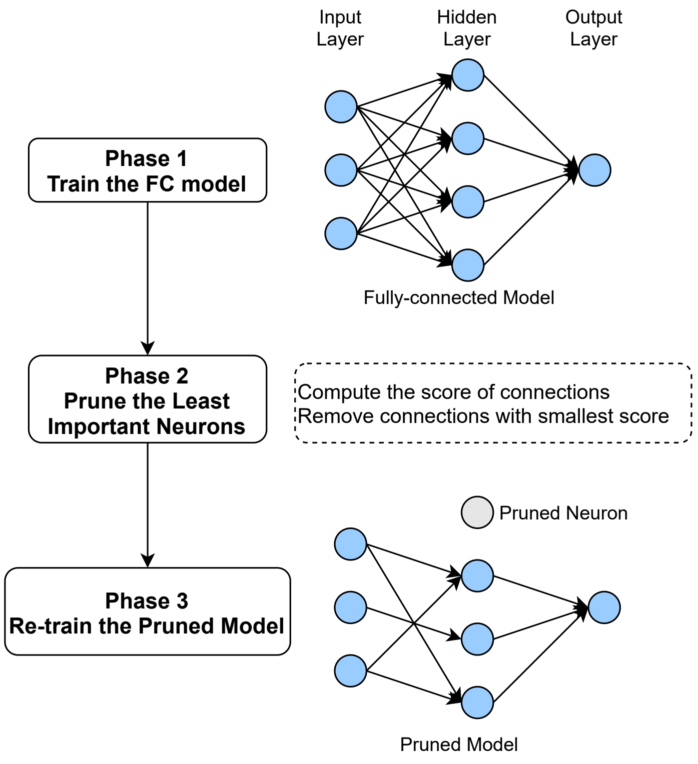
Input Layer
Hidden Layer
Output Layer
Phase 1 Train the FC model
Fully-connected Model
Phase 2 Prune the Least Important Neurons
Compute the score of connections
Remove connections with smallest score
Pruned Neuron
Phase 3 Re-train the Pruned Model
Pruned Model

Fig. 3. Construction of the network intrusion detection model incorporating with neuron pruning

$$L = -\frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n_y} t_i^{(j)}\log(y_i^{(j)}) \tag{4}$$

where m and n_y are the number of samples and the number of data classes, respectively, while $t_i^{(j)}$ and $y_i^{(j)}$ denote the i^{th} true label and predicted output of the j^{th} sample.

In the next phase, the trained weights with the least minimum scores are removed from the network. We use an absolute value to represent the weight score since the smaller weight value means less important for the network. We define the pruning rate p_{prune} $(0 \leq p_{prune} \leq 1)$ as the ratio of the number of removed connections to the total number of connections of the fully connected layer. The percentile p_w of the absolute weight values is calculated such that $p_{prune} \times 100\%$ of weight values are below or equal to p_w. For example, assume that a list of weight values is $\{2, 3, 4, 5\ \}$ and $p_{prune} = 0.5$, then $p_w = 3.5$. Note that, p_w is selected such that at most $p_{prune} \times 100\%$ of weight values are pruned from the fully connected network. Then, if a weight with the absolute value is greater than p_w, we keep this connection. Otherwise, the connection is trimmed from the network. We use a binary mask matrix M with the same size as the weight matrix to denote the pruning status of weights. For example, for the connection from the input to the hidden layer in Fig. 3, the binary mask is represented as $M = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$. Since the first neuron of the hidden layer is removed, the first column of M is set to 0, i.e, no connection is connected to the first neuron of the hidden layer.

In the final phase, the remaining connections of the pruned network are re-trained. We can call this step is fine-tuning. The entropy-based loss function is still used for re-training. The difference with the first phase is that we multiple the connections with the binary mask matrix so that the pruned weights are not trained in this phase. In the proposed method, the parameter p_{prune} selection depends on several factors including the number of incoming packets and the available computing resource of switches. More specifically, p_{prune} can be a high value if there are not many packets or the computing resource is high enough.

Note that since the P4 language only supports integer operations, we need to convert the trained network parameters into integer values. Assume that k bits are used to represent the fractional part of parameters. Recall that W_1 and b_1 denote the trained weight and bias float values of the hidden layer, respectively. Then, the integer hidden vector h_{int} is derived as below.

$$W_{1,int} = \text{int}(W_1 \times 2^k) \tag{5}$$

$$X_{int} = \text{int}(X \times 2^k) \tag{6}$$

$$b_{1,int} = \text{int}(b_1 \times 2^{2k}) \tag{7}$$

$$h_{int} = \begin{cases} (W_{int}X_{int} + b_{int})//2^k, & \text{if } W_{int}X_{int} + b_{int} \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

After doing the division (//), we get the integer part of the output of the division.

After applying the neuron pruning method, the model complexity can be considerably reduced. We now compare the number of connections between the fully-connected and pruned models in the case of a hidden layer. We define n_x and n_h as the number of input and hidden units, respectively. Then, the number of connections can be reduced by $(n_x n_h + n_h n_y)p_{prune}$. In our case, the number of input features and output units are 6 and 5, respectively. If the number of hidden features is 20 and $p_{prune} = 0.5$, the proposed pruning-based architecture can reduce 110 connections compared to the fully-connected model.

When fine-training the pruned network is done, the parameters are sent to programmable switches. Each switch computes the output values y that represent the probability of traffic classes for an incoming packet. Then, the packet is classified into label with argmax(y). Depending on the classified label, we can take different actions for this packet. For example, the packet can be forwarded normally or dropped at the switch.

4. The traffic management maximization strategy

We present an integer linear programming (ILP) problem that maximizes the amount of data to be managed by all participating switches given constraints of classification accuracy and computing resources. Table 1 summarizes main notations in the ILP. The system consists of S switches and there are M available detection models at each switch. Assume that switch i $(1 \leq i \leq S)$ receives N_i incoming packets per second. We define y_{ij} as variables that present the number of packets classified by switch i using model j per second. y_{ij} is a non-negative integer value or $y_{ij} \geq 0$. Note that the total number of packets processed by switch i should be less than the number of incoming packets per second, i.e., $\sum_{j}^{M} y_{ij} \leq N_i$.

We measure the evaluation performance for each detection model including classification accuracy and detection time. The measurement is conducted on programmable switches. Let A_j and T_j denote the accuracy and detection delay of model j $(1 \leq j \leq M)$. We consider two requirements on performance and available computing resource of switches. First, the average classification accuracy of the system should exceed a given threshold value, i.e., $\frac{1}{Y}\sum_{i}^{S}\sum_{j}^{M} y_{ij}A_j \geq A_{th}$ where $Y = \sum_{i}^{S}\sum_{j}^{M} y_{ij}$. Second, the total time during an one-time period for conducting the detection and classification function at each switch should not be greater than a threshold value T_{th}, i.e., $\sum_{j}^{M} y_{ij}T_j \leq T_{th}$ where $0 \leq T_{th} \leq 1$. Note that T_{th} selection depends on the resource availability because a switch may need to perform other tasks besides the packet management task.

Table 1. List of main notation in the ILP formulation

S	Number of switches
M	Number of traffic classification models
N_i	Number of incoming packets per second at switch i
y_{ij}	Number of packets classified by switch i using model j
Y	The total number of packets processed by all switches
A_j	Classification accuracy of model j
T_j	Detection time for a packet of model j
A_{th}	Threshold value for average classification accuracy
T_{th}	Threshold value for available computing resource

The ILP can be defined as follows.

$$\text{maximize} \quad \sum_{i}^{S}\sum_{j}^{M} y_{ij} \tag{9}$$

subject to

$$\frac{1}{Y}\sum_{i}^{S}\sum_{j}^{M} y_{ij}A_j \geq A_{th} \tag{10}$$

$$Y = \sum_{i}^{S}\sum_{j}^{M} y_{ij} \tag{11}$$

$$\sum_{j}^{M} y_{ij}T_j \leq T_{th}, \quad \forall i \tag{12}$$

$$0 \leq y_{ij} \leq N_i \tag{13}$$

$$\sum_{j}^{M} y_{ij} \leq N_i \tag{14}$$

The objective function in (9) indicates that we maximize the number of packets classified by all switches in the network. Constraints in (10) and (11) guarantees that the average classification accuracy should be greater than a threshold value, A_{th}. Then, constraint in (12) ensures that the total time spent for traffic classification in an one-second period should be less than T_{th} ($0 \leq T_{th} \leq 1$). Finally, the inequalities in (13) and (14) show the range of variables y_{ij}.

5. Experimental Results

5.1. Network Setup

To evaluate the NP-based model and compare with the fully-connected (FC) architectures, we consider the IoT dataset [11] with nearly 3 million data samples. The data is collected in a wireless network including smart home devices (i.e., intelligent speaker and Wi-Fi camera) and laptops as well as smart phones. The data distribution of this dataset with five different traffic classes is shown in Table 2. The normal and Botnet data are major classes with 58.82% and 34.76% of the whole dataset, respectively. The remaining three attack types are the minority labels. We randomly divide the whole

Table 2. Label distribution of the IoT network intrusion dataset

	No. of packets	Percentage (%)
Normal	1,756,276	58.82
Reconnaissance	25,210	0.84
MitM	101,885	3.41
DoS	64,646	2.16
Botnet	1,037,977	34.76
Total	**2,985,994**	**100**

dataset into the training and test sets. Network parameters are trained using the training data while the performance of the evaluated model is measured on the test set only.

To estimate the detection delay of detection models, we consider a network with two hosts and one programmable switch that connects these hosts as shown in Figure 4. More specficially, the host h_1 extracts data traffic from pcap traces of the IoT dataset and then sends data packets to the host h_2 via the switch. The switch is in charge of monitoring incoming data by classifying the data packets into one of five different traffic classes. A variety of classification models are implemented in the switch using the P4 programming language. Assume that we forward all packets from h_1 to h_2 to measure the average end-to-end (E2E) delay of detection models under consideration. In fact, the switch takes different actions for incoming packets: forwarding, dropping, adding an alarm header in the packet. The network emulator Mininet [14] is used to define the network topology including the number of hosts, switches and network parameters (e.g., bandwidth and link delay). The Python-based library Scapy is used for packets generation and transmission at h_1 as well as reception at h_2.

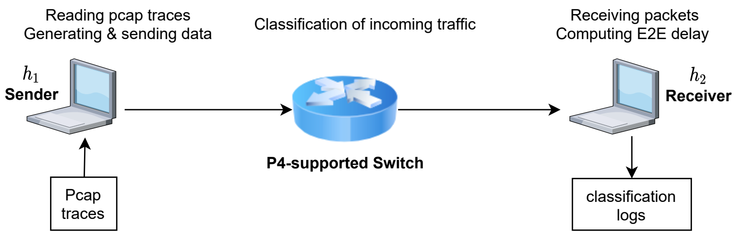

Fig. 4. Network topology used to collect E2E delay

5.2. Evaluation of the pruning-based model

First, we present learning curves of the NP-based detection model on both training and validation sets with pruning rate 0.8 in Figure 5. The number of hidden units is set to 10. There are two phases of parameter learning: training the fully-connected (FC) model and re-training the NP-based model after trimming unimportant connections. In the first phase, all network parameters are learned from the scratch by minimizing the

Table 3. Performance comparison between NP-based models and other approaches

Model	Classification Accuracy (%)	Detection Time (ms)
NP w/ $p_{prune} = 0.2$	94.3	0.822
NP w/ $p_{prune} = 0.4$	93.97	0.851
NP w/ $p_{prune} = 0.6$	93.34	0.77
NP w/ $p_{prune} = 0.8$	89.64	0.73
NB-based model	45.71	0.476
Linear SVM-based model	83.64	0.62
FC w/o hidden layer	88.68	0.578
FC w/ 1 hidden layer	94.17	0.849

loss function. Then, we prune 80% of the least significant connections and re-train the remaining parameters. Thus, the loss curve gradually goes down in both phases while the accuracy improves over epochs. In both phases, we stop network training when there is no improvement in classification accuracy on the validation set for the most recent 20 epochs.

After parameter training, the performance of the NP-based model is considerably lower than the FC model. This observation is attributed to the fact that the number of connections in NP-based model is only one-fifth of the FC architecture, which greatly affects traffic classification performance. Therefore, the classification accuracy of the pruned model (around 90%) is roughly 4.3% lower than that of the FC model (around 94.3%).

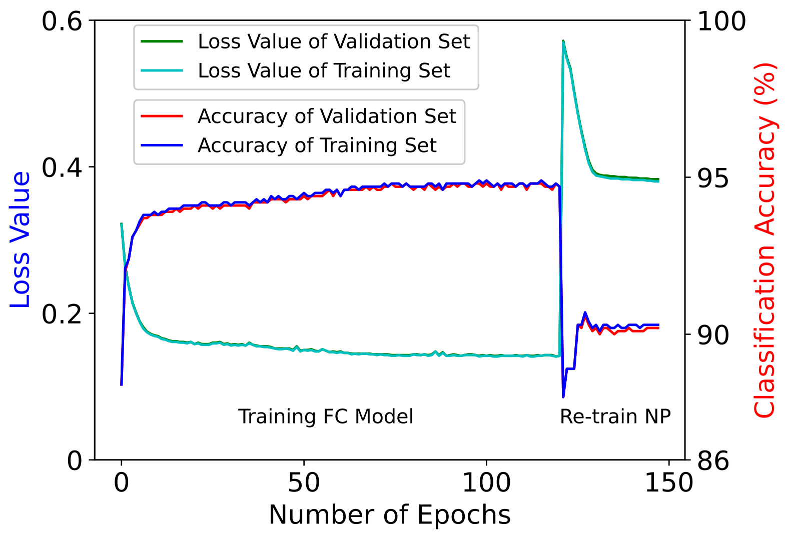

Fig. 5. Learning curves of the NP-based intrusion detection model with $p_{prune} = 0.8$

Then, we compare the performance of NP-based models with different pruning rates with other approaches including the FC architectures and Naive-Bayes (NB)-based model [15] and linear support vector machine (SVM)-based method [16] in Table 3.

Pruning rate changes between 0.2 and 0.8 and we consider the FC model without and with one hidden layer. Classification accuracy deteriorates and the average detection time per packet becomes smaller when we prune more connections (e.g., the pruning rate p_{prune} increases). For example, when the pruning rate changes from 0.2 to 0.8, the accuracy decreases nearly 5% while we can save almost 1 ms for the detection time. In cases of the FC model, using one hidden layer can produce better performance than not using hidden layer. More specifically, the difference in classification accuracy is around 5.5% between two versions of the FC model. However, the detection time of the FC without hidden layer is 0.27 ms lower that the FC with one hidden layer. The proposed NP-based models outperform NB-based and SVM-based methods in terms of classification accuracy. For example, the accuracy of NP with p_{prune} is 44% and 6% higher than that of NB-based and SVM-based approaches.

5.3. Evaluation of the traffic management maximization strategy

In this subsection, we assume that there are a switch ($S = 1$). We find the optimal solutions for ILP in two cases: with and without NP models. Note that since the proposed optimization problem is novel and is not presented elsewhere, there is no heuristic algorithms for this problem. Therefore, we compare the optimal solutions in two above-mentioned situations. Another reason for consideration of these two cases is that we aim to highlight the effectiveness of the pruning-based intrusion detection models, especially when the networking devices are equipped with constrained computing resources. The experimental results show that the NID system can inspect more incoming packets when using pruning-based models than the case of not using pruning methods. If using NP models, the total number of available models (M) is six including four NP models with $p_{prune} = 0.2, 0.4, 0.6, 0.8$ and two FC models without hidden layer and with a hidden layer. If NP models are not considered, $M = 2$ (two FC models). We use OR-Tools developed by Google to solve ILP. The number of packets arriving at the switch is a random variable with uniform distribution in the range [1000, 2500].

Table 4 presents the number of packets assigned to each model where the accuracy threshold changes from 90% to 94%. We use symbol p to indicate the pruning rate for short. In this example, the number of incoming packets is 1,319. If using NP models, we can assign more packets to the FC model with $n_h = 0$ when A_{th} is low. This is attributed to the fact that the FC model with $n_h = 0$ requires the lowest detection time among models under consideration. When A_{th} increases, fewer packets are assigned to the FC model with $n_h = 0$ since the FC model has low classification accuracy. For example, when $A_{th} = 93\%$, only 87 packets are assigned to the FC models while 1,232 packets are monitored by the NP models. When $A_{th} = 94\%$, all packets are managed by the NP models. Note that the NP model with $p_{prune} = 0.2$ has slightly higher classification accuracy and lower detection delay than the FC model with one hidden layer. Therefore, the switch prefers to select the NP model with $p_{prune} = 0.2$ when A_{th} is high.

In cases of not using NP models, the number of packets examined by the FC model with $n_h = 0$ gradually decreases with the increase of A_{th}. This is because the FC model

Table 4. Solutions of ILP with different accuracy requirements

A_{th} (%)	Models	NP $p = 0.2$	NP $p = 0.4$	NP $p = 0.6$	NP $p = 0.8$	FC $n_h = 0$	FC $n_h = 1$	Y
90	w/ NP	0	0	374	3	942	0	1,319
	w/o NP	-	-	-	-	1,001	317	1,318
91	w/ NP	0	0	739	0	580	0	1,319
	w/o NP	-	-	-	-	761	557	1,318
92	w/ NP	204	0	694	1	420	0	1,319
	w/o NP	-	-	-	-	521	798	1,319
93	w/ NP	0	0	1,219	13	83	4	1,319
	w/o NP	-	-	-	-	269	994	1,263
94	w/ NP	854	0	387	0	0	0	1,241
	w/o NP	-	-	-	-	36	1,153	1,189

with $n_h = 0$ has lower classification accuracy than the case of $n_h = 1$. Moreover, if not using the NP models, the total number of managed packets is smaller than the cases of NP models. The benefits of the NP models become more clear when the accuracy threshold increases. For example, the difference between the total number of monitored packets (Y) is only 1 packet when $A_{th} = 90\%$ and $Y = 56$ when $A_{th} = 93\%$.

We investigate the impacts of NP models under a variety of time constraints from 0.5 to 1 (second) in two cases: using and not using NP models. As can be seen in Figure 6, with a higher value of T_{th}, more packets can be managed in both cases. The increase in the total number of monitored packets Y seems to be linear. Y achieves the maximum value (1,322 and 1,263 in two cases) when $T_{th} = 1$ (the whole computing resource at the switch can be used for the task classification task). Note that using the NP models achieves higher performance than not using the NP models with all T_{th} values.

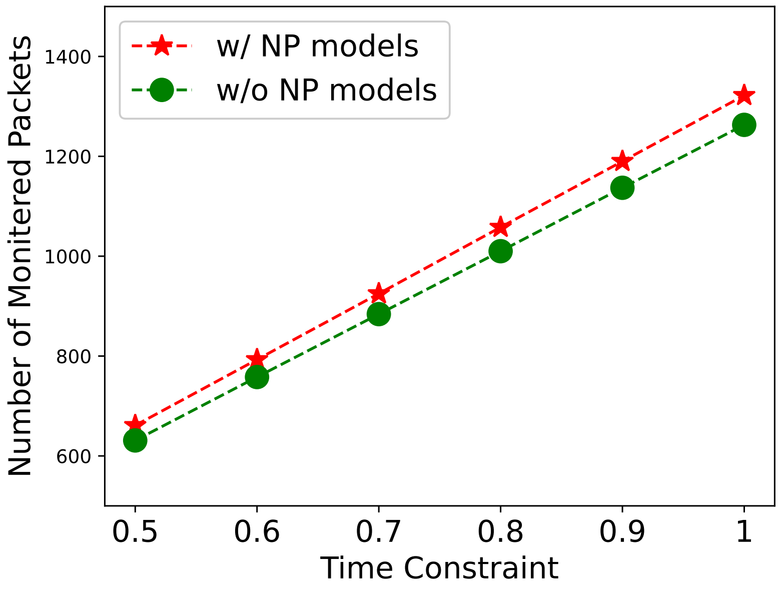

Fig. 6. Effects of NP-based models with different time constraints

6. Conclusion

This paper aims to maximize the amount of data traffic to be managed by programmable switches in Internet of Things (IoT). Since the edge-devices usually lack of computing resources, we introduce a traffic classification model that cooperates with a network simplification method. The proposed architecture can reduce model complexity by trimming the least important connections from the classification model. To evaluate the neuron-pruning (NP)-based model, we introduce an optimization problem that maximizes the number of packets to be classified by all switches in the network. Two requirements on classification accuracy and available computing resource are considered to make the problem to be more practical. The evaluation results illustrate that we can achieve a better traffic management strategy for the network security system in IoT when using NP models. As future work, we will improve the NP-based intrusion detection and classification method to achieve higher performance (e.g., reduce the execution time). Moreover, since finding an optimal solution for the considered optimization problem may not be easy, especially in large-scale networks, heuristic algorithms should be proposed and applied in large-scale networks.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2020.06

References

[1] J. Qi, P. Yang, G. Min, O. Amft, F. Dong, and L. Xu, “Advanced internet of things for personalised healthcare systems: A survey,” *Pervasive and Mobile Computing*, vol. 41, pp. 132 – 149, 2017.

[2] D. Glaroudis, A. Iossifides, and P. Chatzimisios, “Survey, comparison and research challenges of iot application protocols for smart farming,” *Computer Networks*, vol. 168, p. 107037, 2020.

[3] R. Li, T. Song, N. Capurso, J. Yu, J. Couture, and X. Cheng, “Iot applications on secure smart shopping system,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1945–1954, 2017.

[4] L. D. 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.

[5] F. Erlacher and F. Dressler, “On high-speed flow-based intrusion detection using snort-compatible signatures,” *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[6] M. F. Umer, M. Sher, and Y. Bi, “Flow-based intrusion detection: Techniques and challenges,” *Computers And Security*, vol. 70, pp. 238–254, 2017.

[7] T.-N. Dao and H. J. Lee, “Stacked autoencoder-based probabilistic feature extraction for on-device network intrusion detection,” *IEEE Internet of Things Journal*, 2021.

[8] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Advances in neural information processing systems*, pp. 1135–1143, 2015.

[9] B. Hassibi and D. G. Stork, “Second order derivatives for network pruning: Optimal brain surgeon,” in *Advances in Neural Information Processing Systems 5* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), pp. 164–171, Morgan-Kaufmann, 1993.

[10] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient transfer learning,” *CoRR*, vol. abs/1611.06440, 2016.

[11] K. Hyunjae, A. Dong Hyun, L. Gyung Min, Y. Jeong Do, P. Kyung Ho, and K. Huy Kang, “Iot network intrusion dataset,” 2019.

[12] F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi, "P4 edge node enabling stateful traffic engineering and cyber security," *Journal of Optical Communications and Networking*, vol. 11, no. 1, pp. A84–A95, 2019.

[13] A. Agrawal and C. Kim, "Intel tofino2–a 12.9 tbps p4-programmable ethernet switch," in *2020 IEEE Hot Chips 32 Symposium (HCS)*, pp. 1–32, IEEE Computer Society, 2020.

[14] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, (New York, NY, USA), Association for Computing Machinery, 2010.

[15] G. K. Ndonda and R. Sadre, "A two-level intrusion detection system for industrial control system networks using p4," in *5th International Symposium for ICS & SCADA Cyber Security Research 2018 5*, pp. 31–40, 2018.

[16] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted ddos attack detection with p4 language," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

Manuscript received 27-7-2021; Accepted 17-12-2021.

■

Thi-Nga Dao received a B.S. degree in Electrical and Communication Engineering from the Le Quy Don Technical University, Vietnam in 2013, an M.S. degree in Computer Engineering from University of Ulsan in 2016, and a Ph.D. degree in Computer Engihneering from University of Ulsan, South Korea in 2019. From July 2019, she has been working as a lecturer in Faculty of Radio-Electronic Engineering, Le Quy Don Technical University, Hanoi, Vietnam. Her research interests include machine learning-based applications in network security, network intrusion detection and prevention systems, human mobility prediction and mobile crowdsensing. . E-mail: daothinga.mta@gmail.com

Manh-Hung Tran received the B.S. degree in Electrical and Communication Engineering and the M.S. degree in Electrical Engineering from the Le Quy Don Technical University, Vietnam in 1998 and 2003, respectively. His current research interests include Machine Learning, Network Security. Email: trmhung@gmail.com

Huu-Noi Nguyen received the B.Sc. degree in applied mathematics and informatics from Lipetsk State University, Lipetsk, Russia. He currently studying the Ph.D program in Computer Science at Le Quy Don Technical University. His current research interests include Machine Learning, Anomaly Detection, IoT and Information Security. Email: noi.nguyen@lqdtu.edu.vn

PHÁT HIỆN XÂM NHẬP TRONG MẠNG DỰA TRÊN VIỆC LOẠI BỎ BỚT NƠ-RON ĐỂ TỐI ĐA HOÁ VIỆC QUẢN LÝ LƯU LƯỢNG TRONG MẠNG KẾT NỐI VẠN VẬT

Đào Thị Ngà, Trần Mạnh Hùng, Nguyễn Hữu Nội

Tóm tắt

Bài báo này xem xét vấn đề tối đa hóa số lượng gói tin được phân loại bởi hệ thống an ninh mạng trong các thiết bị chuyển mạch có thể lập trình được trong mạng kết nối vạn vật. Với mục đích phát triển một phương pháp bảo mật gọn nhẹ cho các thiết bị chuyển mạch lập trình được với tài nguyên tính toán hạn chế, chúng tôi giới thiệu mô hình phát hiện xâm nhập dựa trên mạng nơ-ron kết hợp với phương pháp loại bỏ bớt nơ-ron để giảm bớt độ phức tạp của mô hình mà không ảnh hưởng nhiều đến độ chính xác của mô hình. Sau đó, chúng tôi xây dựng một bài toán tối ưu nhằm tối đa hóa lượng lưu lượng mạng được giám sát bởi tất cả các thiết bị chuyển mạch với các yêu cầu về độ chính xác phân loại và giới hạn về tài nguyên tính toán. Bài toán tối ưu được xem xét trong hai trường hợp: sử dụng và không sử dụng các mô hình dựa trên việc loại bỏ bớt nơ-ron (NP) để chỉ ra những lợi ích của kiến trúc gọn nhẹ được đề xuất. Kết quả đánh giá cho thấy rằng các mô hình dựa trên NP cho phép các bộ chuyển mạch quản lý nhiều lưu lượng mạng hơn trong khi vẫn đáp ứng các yêu cầu nhất định về độ chính xác và giới hạn tài nguyên tính toán.