Received: 27 March 2019 | Revised: 28 June 2019 | Accepted: 29 July 2019

DOI: 10.4218/etrij.2019-0156

SPECIAL ISSUE

ETRI Journal WILEY

Supervised learning-based DDoS attacks detection: Tuning hyperparameters

Meejoung Kim

Research Institute for Information and Communication Technology, Korea University, Seoul, Rep. of Korea

Correspondence
Meejoung Kim, Research Institute for Information and Communication Technology, Korea University, Seoul, Rep. of Korea.
Email: meejkim@korea.ac.kr

Funding information
This research was supported by the Mid-career Research Program and Basic Science Research Program through NRF grant funded by the MEST (NRF-2019R1A2C1002706, NRF-2016R1D1A1B03931037) and supported by the Korea University Grant.

Two supervised learning algorithms, a basic neural network and a long short-term memory recurrent neural network, are applied to traffic including DDoS attacks. The joint effects of preprocessing methods and hyperparameters for machine learning on performance are investigated. Values representing attack characteristics are extracted from datasets and preprocessed by two methods. Binary classification and two optimizers are used. Some hyperparameters are obtained exhaustively for fast and accurate detection, while others are fixed with constants to account for performance and data characteristics. An experiment is performed via TensorFlow on three traffic datasets. Three scenarios are considered to investigate the effects of learning former traffic on sequential traffic analysis and the effects of learning one dataset on application to another dataset, and determine whether the algorithms can be used for recent attack traffic. Experimental results show that the used preprocessing methods, neural network architectures and hyperparameters, and the optimizers are appropriate for DDoS attack detection. The obtained results provide a criterion for the detection accuracy of attacks.

KEYWORDS
accuracy of detection, DDoS attack, long short-term memory, machine learning, tensorflow

1 | INTRODUCTION

The evolution of networks and the development of smart mobile devices have made it more convenient and easier than ever to obtain information and interact with others. Such convenience, however, leads to new types of network security problems that threaten our lives, such as personal information leaks or even national security issues. Network security consists of several factors, such as policies and practices adopted to prevent and detect the behavior of malicious nodes. Attack prevention and detection techniques in networks have been studied steadily over the past two decades with various approaches, such as stochastic modeling, decision theory, and game theory [1–4]. Recently, machine learning (ML) techniques, such as multilayer perceptron (MLP), have been applied to network attack detection [5–23]. In addition, as social media outlets, such as Facebook and Twitter, are regarded as possible vehicles for the next large cybercrime [24], research on the prediction of cyberattacks based on social media data has been studied [20].

Some attacks by malicious nodes prevent access to network resources and cause serious threats and damage. For instance, a Denial of Service (DoS) attack deprives legitimate end users of network resources, as malicious nodes overwhelm the target system by transmitting extensively and eventually paralyzing the system. There are several types of DoS attacks such as TCP/SYN Flood, Ping Flood, UDP Flood, and Distributed Denial of Service (DDoS) [25]. In a DDoS attack, multiple compromised malicious nodes attack a single target. In a typical DDoS attack, the attacker makes

a DDoS master (botmaster), using the vulnerability in one computer system. The botmaster then identifies vulnerable systems and infects them with malware. Eventually, it instructs the infected or controlled computers (zombie army or botnet) to launch an attack against a specified target. A Smurf attack is an example of a DDoS attack. In the attack, large numbers of Internet Control Message Protocol (ICMP) echo request packets are broadcasted to a network using an IP address that is a spoofed source IP of the intended victim. The nodes associated with the network respond to this request by sending replies to the IP address of the victim. Consequently, the victim computer is flooded with traffic [25].

There are various ways to prevent and detect DDoS attacks, such as the trace back scheme and traffic filtering autonomous system [26–29]. The trace back scheme finds the locations of sources for attacks, whereas a traffic filtering autonomous system uses a traffic filter to drop traffic that does not originate from the network or is destined to the network. Another method is traffic aggregation with traffic classification. Signature-based detection and anomaly-based detection are the two different approaches for this method. The former monitors packets on a specified network and compares the packets with a set of signatures from known malicious threats [30], whereas the latter depends on the network behavior as the system distinguishes attack data from traffic data based on a training process. The output obtained by the training process can be updated when new data are added while preserving the previously acquired knowledge. ML techniques belong to this category.

Considerable research has recently been conducted on the prevention and detection of cyberattacks, including DoS and DDoS attacks [5–35], especially in cloud computing environments [16,17], [25], [32–34] and software defined networking (SDN) architecture [17–19], [32]. Several ML techniques ([5,6], [8–22]) and the autoregressive integrated moving average (ARIMA) time series model [31] were considered for the detection of DoS attacks including DDoS. In addition, [20] used social media data to predict attacks, whereas [35] developed DDoS Testbed (DDoSTB) to generate a variety of attack scenarios. Many studies using ML techniques in detecting attacks have mainly applied different ML feature extraction algorithms or modified algorithms to improve performance. Details of recent studies are presented in the following section.

There are several hyperparameters associated with ML techniques. Hyperparameters are the parameters that control the learning process. They are the higher level concepts of ML techniques, represented as variables, which determine the complexity of network structure and the ability to learn. For instance, the numbers of hidden units and layers, and the learning rate are examples of hyperparameters. Because appropriately chosen values of hyperparameters may resolve overfitting and underfitting problems and reduce training time and costs that lead to performance improvement, hyperparameter tuning is a critical step in the training process of an ML model [36]. Recently, these issues have been studied [37,38]. In addition, preprocessing of data is also an important step in the ML technique because it derives useful information from raw data and transforms the derived information into a format that increases the learning ability of the model [39].

In this study, we analyze DDoS attacks by ML techniques. Datasets derived from three different sources of traffic including DDoS attacks are used for training and testing after preprocessing by proper transformation. In the training process, two supervised learning algorithms, a basic neural network (BNN) and a long short-term memory recurrent neural network (LSTM RNN), are considered. ML is performed via TensorFlow. The motivation of this study is as follows: (a) How do the different preprocessing methods and various values of hyperparameters affect the performance of the ML techniques? (b) What are the suboptimal values of hyperparameters that enable quick and accurate detection for feature extraction algorithms? (c) Do learning former traffic and learning one dataset affect the learning of sequential traffic and another dataset, respectively, in a DDoS attack? (d) Are the ML algorithms suitable for detecting attacks on older data applicable for detecting attacks on recent data? Based on this motivation, two preprocessing methods, three training scenarios, and several different environments are considered. Different environments are represented by various hyperparameters, such as different optimizers and network architectures in the experiment. The learning rates and iteration numbers are obtained by exhaustive search based on the grid search method [36] in such a way as to rapidly decrease the cost function for BNN, and one of the determined learning rates is used in LSTM RNN. Other hyperparameters, including the numbers of layers and hidden nodes, are fixed with constants. The constants are determined by considering the detection accuracy and observed characteristics of data during the experiment. Feature extraction algorithms are applied to recent traffic, including attacks, to determine whether they are applicable to new attack characteristics. There are several studies that have considered the optimal hyperparameters in an ML model, including [37]. To the best of our knowledge, however, this is the first attempt to investigate the joint effect of preprocessing methods and hyperparameters on the performance of DDoS attack detection using ML techniques. The contribution of this study is as follows: (a) The joint effect of preprocessing methods and hyperparameters on the performance of ML techniques is investigated. (b) The effect of learning former traffic on the analysis of sequential traffic and the effect of learning one dataset on application to another dataset are studied. (c) The applicability of existing ML technologies to detect attacks with new attack characteristics is investigated. (d) Two

optimizers commonly used in ML models are compared for DDoS detection using TensorFlow.

This paper includes the following: Section 2 presents existing studies of detection of cyberattacks including DDoS attacks; Section 3 describes the ML process; Section 4 and Section 5 present performance measures and the experimental results, respectively; and Section 6 provides the conclusion.

2 | RELATED WORKS: DETECTION OF CYBERATTACKS INCLUDING DDOS

There are many studies addressing the prevention and detection of cyberattacks including DDoS attacks [5–35] and many of them are based on ML techniques [5–16], [18–23]. Examples of techniques are decision trees [5], [16]; support vector machine (SVM) [6], [16]; two-level hybrid solution consisting of anomaly and misuse detection [7]; classification techniques such as MLP, Naïve Bayes, and random forest (RF) [8], [16]; K-mean clustering algorithms [9], [16]; genetic algorithm (GA) [10]; ensemble of neuro-fuzzy and genetic fuzzy systems [11]; Lyapunov exponent based on entropy [12], [31]; convolutional neural network (CNN) [14], [20]; RNN [14]; LSTM RNN [14], [18–20]; gated recurrent unit (GRU) RNN [14]; hybrid heterogeneous multi-classifier ensemble learning [22]; and deep-feature extraction and selection method [23]. Details of the existing studies are as follows.

A DDoS attacks detection system designed based on decision tree and traffic-flow pattern-matching was used to trace back the locations of attackers [5]. The study of [6] focused on the generation and detection of DDoS attack data by using enhanced SVM. A two-level hybrid approach consisting of two anomaly detection components and one misuse detection component was considered [7]. A new dataset containing modern DDoS attacks, such as SIDDoS and HTTP Flood, was collected in different network layers, and MLP, Naïve Bayes, and RF were applied to classify them [8]. An experiment was conducted to observe the discriminating capabilities of hierarchical and K-mean clustering algorithms for botnet trace in the presence of background internet traffic [9]. A GA based approach [10] and adaptive and hybrid neuro-fuzzy systems as subsystems of an ensemble [11] were considered in DoS and DDoS detection. A variation of the Lyapunov exponent was proposed to detect anomalies in network traffic based on entropy [12].

Several deep learning (DL) models, such as CNN, RNN, LSTM, and GRU neural network (NN) [14], and the PCA RNN framework [15] were considered to identify DDoS attacks. The performances of SVM, decision tree, Naïve Bayes, RF, K-means, and Gaussian-mixture model were compared [16] in terms of expectation-maximization. A defense mechanism mitigating DDoS attacks applicable to the fog environment as well as the cloud environment was considered [18], whereas a DDoS detection model and defense system based on LSTM was proposed for SDNs [19]. Classifiers based on CNN and LSTM were presented to predict the likelihood of cyberattack-related words in large volumes of social networking texts [20]. Hybrid heterogeneous multi-classifier ensemble learning and a heuristic detection algorithm were proposed to construct a detection system [22]. The combination of stacked feature extraction and weighted feature selection was proposed for the detection of an impersonating Wi-Fi network [23]. A real-time DDoS attack detection method using a correlation measure was proposed [26], whereas a mechanism to detect botnets using their fundamental characteristics, such as group activity, was considered [27]. Probabilistic and deterministic packet marking techniques to detect DDoS attacks [28] and a highly distributed DDoS attack blocking mechanism [29] were proposed. A signature detection technique was proposed to investigate the ability of various routing protocols that facilitate intrusion detection when the attack signatures are completely known [30]. Furthermore, a time series based on the number of packets in DDoS attack traffic was considered; meanwhile, the ARIMA model was used for prediction, and the chaotic behavior of prediction error was considered by using the Lyapunov exponent [31]. An experiment of a real-time DDoS attack was conducted to study attack absorption and mitigation for various target services in the presence of dynamic cloud resource scaling [33]. The impact of DDoS attacks on web services was measured using several metrics such as throughput, average latency, and round-trip time [35]. Many studies that focused on passive defenses have analyzed the traffic on the destination side after attack occurred, while [16] and [18] considered it on the source side.

In addition, there are several survey papers dealing with DDoS attacks. The evaluation and ranking of several supervised ML algorithms were discussed with the aim of increasing precision and recall while maintaining detection accuracy and reducing type I and type II errors [13]. Different scenarios of DDoS attacks were surveyed [17], focusing on different methods of classification, detection, and defense of DDoS attack, and the eligibility of ML approach to DDoS attack defense was evaluated. A systematic review of ML techniques [21] and defense methods [25] for DoS attack detection were provided. Defense mechanisms against DDoS attacks in cloud computing environments using SDNs were also provided [32]. Furthermore, the main features of existing DDoS mitigation strategies and their functionalities were discussed [34].

3 | PROCESS OF MACHINE LEARNING

3.1 | Architecture of neural network

Artificial neural networks consist of three parts: input layer, hidden layer, and output layer. A dataset is fed into the input layer and preprocessed for efficient learning. The learning process of the training set occurs in the hidden layer. During the learning process, the features of training data are extracted and saved as "learning parameters," which are presented as weight matrices **W** and biases **b**. In the output layer, tests are performed using these matrices to verify the results of learning. Figure 1 illustrates the general ML process.

In this study, we consider two neural networks: a BNN and a LSTM RNN. BNN is a model using a basic logistic regression classifier, whereas LSTM RNN is a model using LSTM cells in hidden layers. LSTM RNN is a network model that considers the vanishing gradient problem of RNN, which is an NN dealing with time series. A LSTM cell is illustrated in Figure 2.

The objective of an ML process is to detect attacks from a traffic set with high accuracy as fast as possible. In general, the accuracy becomes high if the cost becomes small, and a smaller number of iterations (epochs, eps) of learning enable faster detection. As the cost is represented as a function of learning parameters **W** and **b**, the objective function of ML can be formulated as follows:

$$\textbf{(Objective)} \qquad \min_{k} \left\{ \min_{\mathbf{W},\mathbf{b}} C(\mathbf{W}^{(k)},\mathbf{b}^{(k)}) \right\}.$$

In the (Objective), C is a cost function, and k is the number of iterations. The notations in (Objective) will be explained in Algorithm 1, which is a description of BNN.

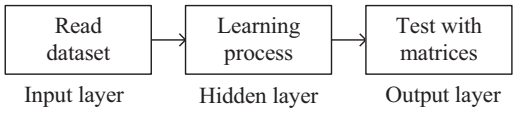

FIGURE 1 Machine learning process

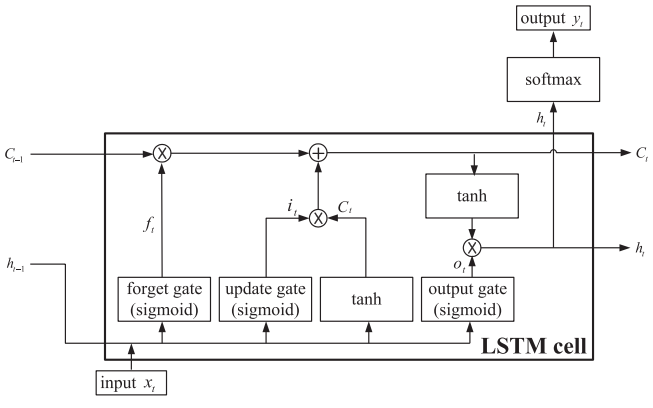

FIGURE 2 Learning process of LSTM

3.2 | Preprocessing of datasets

A dataset has to be preprocessed for efficient learning. To determine the preprocessing methods, traffic characteristics, such as distributions of transmitted packets and accumulated packets in a unit time interval, have to be investigated.

In this study, three datasets are used for ML processing: "DDoS Attack 2007" obtained from CAIDA (Center for Applied Internet Data Analysis) [40], "DDoS attack 1998" of DARPA (Defense Advanced Research Projects Agency) obtained from MIT Lab [41], and "recent DDoS attacks dataset" obtained from [42]. The third dataset was collected in 2016 by the authors of [8]. It contains four types of recent DDoS attacks—HTTP Flood, SQL Injection Distributed Denial of Service (SIDDOS), UDP Flood, and Smurf.

The Wireshark, a traffic analyzer, is used to examine traffic characteristics. The dataset from CAIDA contains packets transmitted "from attackers to a victim (to victim)" and "from a victim to attackers (from victim)" separately. Figure 3 shows the number of packets transmitted "from victim" and "to victim" over the course of 5 minutes when the DDoS attack begins for CAIDA. It shows that the transmitted packets "from victim" and "to victim" are maximally below 800 bytes and above 150 000 bytes, respectively, for 1 second. As seen from Figure 3, there are large differences between the number of packets occurring in the normal state (before 80 seconds) and that in the attack state (after 80 seconds). That is, the distributions of transmitted packets in the successive time intervals of these two states are far from normal. To deal with such data, preprocessing of the dataset is necessary for efficient training.

The Box-Cox transformation (BCT) and min-max transformation (MMT) are well known preprocessing methods of datasets.

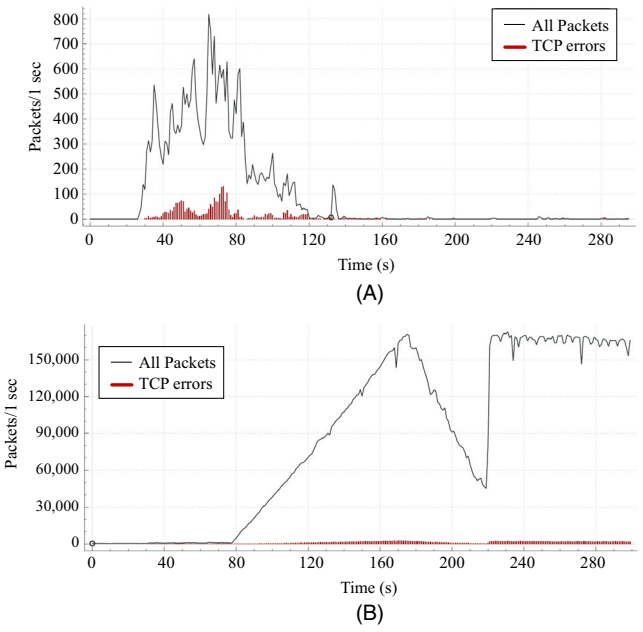

FIGURE 3 Number of packets transmitted in 5 min: (A) from victim (B) to victim

BCT converts a dataset that is not normally distributed into a rather normally distributed dataset [43], whereas MMT is a commonly used method in ML, taking values of [0, 1]. These are defined by

$$\mathrm{BCT}(x_i)=\begin{cases}(x_i^{\lambda}-1)/\lambda, & \lambda\neq 0,\\ \log(x_i), & \lambda=0,\end{cases}$$

and

$$\mathrm{MMT}(x_i)=\frac{x_i-\min(\mathbf{x})}{\max(\mathbf{x})-\min(\mathbf{x})}, \tag{1}$$

respectively, where λ is power parameter and $\mathbf{x}=(x_1,\ \dots,x_n)$.

3.3 | Learning and testing process

In the following algorithm, supervised learning is considered with the logistic classification, and training is implemented using the back-propagation algorithm.

Let $\{\mathbf{X}_j\}_{j=1}^{N}$ and $\{y_j\}_{j=1}^{N}$ be the preprocessed dataset and corresponding label set, respectively, where N is the number of data elements. For the label, binary classification is applied. That is, zero or one is assigned to $y_{j,\bullet}$ depending on the traffic state—normal or attack. Divide $\{\mathbf{X}_j\}_{j=1}^{N}$ and $\{y_j\}_{j=1}^{N}$ into a training (testing) set $\{\mathbf{X}_{j,\mathrm{tr}}\}_{j=1}^{N_1}$ ($\{\mathbf{X}_{j,\mathrm{te}}\}_{j=1}^{N_2}$) and corresponding label set $\{y_{j,\mathrm{tr}}\}_{j=1}^{N_1}$ ($\{y_{j,\mathrm{te}}\}_{j=1}^{N_2}$), respectively, where N_i, $i=1, 2$, are the numbers of training and testing data elements satisfying $N_1+N_2=N$. Each data $\mathbf{X}_{j,\bullet}$ is a vector given by $\mathbf{X}_{j,\bullet}=(x_{1,j},\ \dots,x_{m,j})_{\bullet}$, where each component $x_{i,j}$ is a value representing the characteristic of traffic, $\bullet$ is either "tr" or "te," and m is an input dimension. The process of BNN is as follows:

Algorithm 1 Process of Machine Learning (BNN)

1: Read a dataset and pre-process the dataset: $\{\mathbf{X}_j\}_{j=1}^{N}$ and $\{y_j\}_{j=1}^{N}$.

2: Divide the pre-processed dataset $\{\mathbf{X}_j\}_{j=1}^{N}$ and corresponding label set $\{y_j\}_{j=1}^{N}$ into a training set and testing set:

$\{\mathbf{X}_j\}_{j=1}^{N}=\{\{\mathbf{X}_{j,\mathrm{tr}}\}_{j=1}^{N_1},\{\mathbf{X}_{j,\mathrm{te}}\}_{j=1}^{N_2}\}$ and $\{y_j\}_{j=1}^{N}=\{\{y_{j,\mathrm{tr}}\}_{j=1}^{N_1},\ \{y_{j,\mathrm{te}}\}_{j=1}^{N_2}\}$.

3: Initialize variables: weight $\mathbf{W}_p^{(1)}$ and bias $\mathbf{b}_p^{(1)}$, $p=1,\cdots,l$.

4: Training and verifying processes: for $i=1:k$

4.1 Training: Compute $S(L(\mathbf{X}_{j,\mathrm{tr}}))_p^{(i)}=1/\{1+e^{-(\mathbf{X}_{j,\mathrm{tr}}\mathbf{W}_p^{(i)}+\mathbf{b}_p^{(i)})}\}$, $j=1,\cdots,N_1,\ p=1,\cdots,l$.

Compute the cost function $C^{(i)}\equiv C(\mathbf{W}_l^{(i)},\mathbf{b}_l^{(i)})$.

Train by using gradient decent optimizer (or Adam optimizer) to minimize the cost function by updating $\mathbf{W}_p^{(i)}$ and $\mathbf{b}_p^{(i)}$, $p=1,\cdots,l$.

4.2 Verifying: Compute the accuracy of $\{y_{j,\mathrm{tr}}\}_{j=1}^{N_1}$ by using the obtained $\mathbf{W}_p^{(i)}$ and $\mathbf{b}_p^{(i)}$ for each i.

5: Testing process: Applying $\mathbf{W}_p^{(k)}$ and $\mathbf{b}_p^{(k)}$ to $\{\mathbf{X}_{j,\mathrm{te}}\}_{j=1}^{N_2}$, predict $\{y_{j,\mathrm{te,pred}}\}_{j=1}^{N_2}$. Compare $\{y_{j,\mathrm{te,pred}}\}_{j=1}^{N_2}$ and $\{y_{j,\mathrm{te}}\}_{j=1}^{N_2}$.

6: End

In step 3, the initial values of weights and biases are given by $\mathbf{W}_p^{(1)}$ and $\mathbf{b}_p^{(1)}$, and l is the number of hidden layers. Step 4 consists of training and verifying processes. In 4.1, the training process, the following linear regression L and sigmoid function S are used:

$$L(\mathbf{X}_{j,\mathrm{tr}})_p^{(i)}=\mathbf{X}_j\mathbf{W}_p^{(i)}+\mathbf{b}_p^{(i)}$$

and

$$S\{L(\mathbf{X}_{j,\mathrm{tr}})\}_p^{(i)}=\frac{1}{1+e^{-L(\mathbf{X}_{j,\mathrm{tr}})_p^{(i)}}},\quad p=1,\ \dots,l. \tag{2}$$

In the algorithm, $\mathbf{W}_p^{(i)}$ and $\mathbf{b}_p^{(i)}$, the i-th updated values of $\mathbf{W}_p^{(1)}$ and $\mathbf{b}_p^{(1)}$, are determined during the training process in such a way as to minimize the cost function. Note that $S(L(\mathbf{X}_{j,\mathrm{tr}}))_p^{(i)}$ is the calculated value of y_j by training $\mathbf{X}_{j,\mathrm{tr}}$ in the p-th layer and i-th iteration.

The cost function is defined as follows:

$$C(\mathbf{W}_l^{(i)},\mathbf{b}_l^{(i)})=\frac{1}{N_1}\sum_{j=1}^{N_1}C(S(L(\mathbf{X}_{j,\mathrm{tr}}))_l^{(i)},y_{j,\mathrm{tr}}). \tag{3}$$

In (3), $C(S(x),y)$ is the negative log cost function defined by

$$C(S(x),y)=\begin{cases}-\log(S(x)), & y=1,\\ -\log(1-S(x)), & y=0.\end{cases} \tag{4}$$

By substituting (4) into (3), the average cost function computed after the i-th learning can be written as

$$C(\mathbf{W}_l^{(i)},\mathbf{b}_l^{(i)})=\frac{1}{N_1}\sum_{j=1}^{N_1}\begin{bmatrix}y_{j,\mathrm{tr}}\log S(L(X_{j,\mathrm{tr}}))_l^{(i)}\\ +(1-y_{j,\mathrm{tr}})\log(1-S(L(\mathbf{X}_{j,\mathrm{tr}}))_l^{(i)})\end{bmatrix}. \tag{5}$$

Equation (5) evaluates the closeness of $y_{j,\mathrm{tr}}$ and $S(L(\mathbf{X}_{j,\mathrm{tr}}))_p^{(i)}$, which equals the Kullback-Leibler (KL) divergence between them. If the gradient decent (GD) algorithm is used, $\mathbf{W}_p^{(i)}$ and $\mathbf{b}_p^{(i)}$ are updated by

$$W_{j,p}^{(i)}\leftarrow W_{j,p}^{(i-1)}-\alpha\partial C(\mathbf{W}_l^{(i-1)},\mathbf{b}_l^{(i-1)})/\partial W_{j,p},\quad j=1,\ \dots,m,$$

and

$$b_{j,p}^{(i)}\leftarrow b_{j,p}^{(i-1)}-\alpha\partial C(\mathbf{W}_l^{(i-1)},\mathbf{b}_l^{(i-1)})/\partial b_{j,p},\quad j=1,\ \dots,n, \tag{6}$$

where n is the output dimension and α is the learning rate. The GD algorithm in this step can be replaced by modified algorithms, such as the adaptive moment estimation (Adam: AD) [44]. In 4.2, the verifying process, $\{\mathbf{X}_{j,\mathrm{tr}}\}_{j=1}^{N_1}$ is tested for each training step by using $\mathbf{W}_p^{(i)}$ and $\mathbf{b}_p^{(i)}$ obtained in 4.1, and the accuracy for the training data is computed in each iteration. We call the accuracy for the training data "training accuracy" and

denote it by "Acc_{tr}." Acc_{tr} is computed by using $y_{j,\text{tr,pred}}$, the prediction of $y_{j,\text{tr}}$, which is determined as follows:

$$y_{j,\text{tr,pred}} = \begin{cases} 0, & S(L(\mathbf{X}_{j,\text{tr}}))_l^{(i)} < 0.5, \\ 1, & S(L(\mathbf{X}_{j,\text{tr}}))_l^{(i)} \geq 0.5, \end{cases} \quad j = 1, \ldots, N_1. \quad (7)$$

The iteration number of training, k, is determined in this step in such a way as to obtain a high Acc_{tr}. Now, using the matrices $\mathbf{W}_p^{(k)}$ and $\mathbf{b}_p^{(k)}$, $\{\mathbf{X}_{j,\text{te}}\}_{j=1}^{N_2}$ is tested in step 5. $S(L(\mathbf{X}_{j,\text{te}}))_l^{(k)}$ is computed and this determines $y_{j,\text{te,pred}}$, the prediction of $y_{j,\text{te}}$, for the testing set according to (7). We call the accuracy of the testing set "testing accuracy" and denote it by "Acc_{te}."

For LSTM RNN, the functions of gates in Figure 2 are given by

$$\begin{aligned} &f_t = S(\mathbf{W}_f \cdot [h_{t-1}, \mathbf{X}_t] + \mathbf{b}_f), i_t = S(\mathbf{W}_i \cdot [h_{t-1}, \mathbf{X}_t] + \mathbf{b}_i), \\ &\widetilde{C}_t = \tanh(\mathbf{W}_C \cdot [h_{t-1}, \mathbf{X}_t] + \mathbf{b}_C), o_t = S(\mathbf{W}_o \cdot [h_{t-1}, \mathbf{X}_t] + \mathbf{b}_o), \\ &C_t = f_t \otimes C_{t-1} \oplus i_t \otimes \widetilde{C}_t, \text{and } h_t = o_t \otimes \tanh(C_t), \end{aligned} \quad (8)$$

where $\otimes$ and $\oplus$ represent entry-wise multiplication (Hadamard product) and entry-wise addition (direct sum), respectively. The subscripts of the matrices in (8) only denote the gates. The notations describing the layer and iteration number are deleted in the matrices for notational simplicity. The detailed explanation of LSTM and the explicit formulas of the functions in (8) can be found in several articles including [45]. Table 1 summarizes the notations used in this paper.

4 | PERFORMANCE MEASURES

The two accuracies, Acc_{tr} and Acc_{te}, true positive rate (TPR, recall), positive predictive value (PPV, precision), F1 score, and accuracy score are considered as the performance measures. They are defined by

$$\text{Acc}_{\text{tr}} = \frac{1}{N_1} \sum_{j=1}^{N_1} (y_{j,\text{tr}} - y_{j,\text{tr,pred}}), \text{Acc}_{\text{te}} = \frac{1}{N_2} \sum_{j=1}^{N_2} (y_{j,\text{te}} - y_{j,\text{te,pred}}), \quad (9)$$

$$\begin{aligned} &\text{TPR} = \text{TP}/(\text{TP} + \text{FN}), \text{PPV} = \text{TP}/(\text{TP} + \text{FP}), \\ &\text{F1 score} = 2(\text{PPV} \times \text{TPR})/(\text{PPV} + \text{TPR}), \\ &\text{and Accuracy score} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}), \end{aligned} \quad (10)$$

where TP, FP, TN, and FN are true positive, false positive, true negative, and false negative, respectively. Here, attack is positive class and normal is negative class. In addition, the execution times for training and testing, T_{tr} and T_{te}, are considered.

TABLE 1 Summary of notations

Notation	Description
$\mathbf{X}_j$ ($\mathbf{X}_{j,\text{tr}}$, $\mathbf{X}_{j,\text{ te}}$)	j-th input data (training, testing)
y_j ($y_{j,\text{tr}}$, $y_{j,\text{te}}$)	Label corresponding to $\mathbf{X}_j$ ($\mathbf{X}_{j,\text{tr}}$, $\mathbf{X}_{j,\text{ te}}$)
λ	Power parameter
m	Dimension of input data
n	Dimension of output data
$\mathbf{A}^{(i)}$	i-th updated value of a vector initially given by $\mathbf{A}^{(1)}$
N (N_1, N_2)	Number of data elements (training, testing)
α	Learning rate
l	Number of hidden layers
k	Number of iterated learning

5 | EXPERIMENTAL RESULTS

In this section, the experimental results are presented. The experiment is performed by using Python 3.6 and TensorFlow v.1.7.0 on Intel Core i7, 16 GB RAM. The datasets of CAIDA and DARPA contain approximately 2 hours of traffic of size 23.5 GB, consisting of attack traffic and normal traffic. Specifically, the CAIDA dataset covers from 20:50:08 UTC (Coordinated Universal Time) to 21:56:16 UTC (actual date and time: 20070804, 13:49:36–14:54:36) and DARPA 1998 covers 19980629, 20:57:14–19980630, 04:59:14. The attacks in CAIDA are identified at approximately 21:13 UTC with the rapid increase in network load within a few minutes, from approximately 200 Kbits/sec to approximately 90 MB/sec, whereas the attacks in DARPA are identified at approximately 04:33 with the increase in network load from approximately 20 Kbits/sec to approximately 11 MB/sec. The dataset is divided into 10-s periods; consequently, 707 small subsets (CAIDA: 396, DARPA: 311) are obtained. Normal and attack traffic are mixed in each dataset, and labels are assigned to distinguish them. Then, the CAIDA (DARPA) dataset is divided into 158 (156) and 238 (155) small sets of normal traffic and attack traffic, respectively. Meanwhile, 300 sets of normal traffic and 300 of UDP Flood attack are extracted from the third dataset to confirm the applicability of the ML algorithms to recent attack traffic.

The datasets are analyzed to determine the components of traffic that enable the representation of attack characteristics. Figure 4 shows the information provided by CAIDA dataset. As the "to victim" dataset is divided into 10-s periods, Figure 4 is obtained by capturing the last few lines of two sets among 396 small subsets. As shown, it contains the number of transmissions, transmission times, source/destination addresses, and number of cumulated packets (in bytes) in

No.	Time	Source	Destination	Protocol	Length	Info	Cummulative Bytes
3709	9.985956	195.198.120.238	71.126.222.64	ICMP	60	Echo (ping) request id=0xee51, …	221652
3710	9.985959	195.198.120.238	71.126.222.64	ICMP	60	Echo (ping) request id=0xee51, …	221712
3711	9.988069	202.1.175.252	71.126.222.64	ICMP	60	Echo (ping) request id=0xce1d, …	221772
3712	9.988945	51.173.229.255	71.126.222.64	ICMP	60	Echo (ping) request id=0x0200, …	221832
3713	9.991742	192.95.27.190	71.126.222.64	ICMP	60	Echo (ping) request id=0xc6fa, …	221892
3714	9.992277	40.75.89.172	71.126.222.64	ICMP	60	Echo (ping) request id=0x4ba4, …	221952
3715	9.993425	192.120.148.227	71.126.222.64	ICMP	60	Echo (ping) request id=0x0200, …	222012

(A)

No.	Time	Source	Destination	Protocol	Length	Info	Cummulative Bytes
1337680	9.999933	203.255.196.135	71.126.222.64	ICMP	60	Echo (ping) request id=0x0200, …	78926994
1337681	9.999936	98.238.182.252	71.126.222.64	TCP	48	3702→1182 [SYN] Seq=0 Win=64512 …	78927042
1337682	9.999941	39.37.152.41	71.126.222.64	ICMP	60	Echo (ping) request id=0x0200, …	78927102
1337683	9.999975	194.28.128.232	71.126.222.64	TCP	48	28347→25601 [SYN] Seq=0 Win=6451…	78927150
1337684	9.999980	194.27.34.115	71.126.222.64	ICMP	60	Echo (ping) request id=0x0200, …	78927210
1337685	9.999984	200.184.115.179	71.126.222.64	ICMP	60	Echo (ping) request id=0x0200, …	78927270
1337686	9.999987	192.145.245.96	71.126.222.64	ICMP	60	Echo (ping) request id=0x0200, …	78927330

(B)

FIGURE 4 Information about packet transmission to victim: (A) not attack (B) attack

(a) normal state and (b) attack state. According to the figure, the accumulated packets transmitted to a victim are 222 012 bytes and 78 927 330 bytes in normal and attack states, respectively. A similar trend can be found in the "from victim" dataset in CAIDA and DARPA traffic. Based on the analysis of datasets, three values that could be the evidence of attack occurrence are extracted from the three datasets (ie, $m = 3$), which are the number of source IP addresses, number of transmitted packets by IPs, and number of accumulated packets for CAIDA and DARPA. However, because recent traffic provides different traffic information, different components are selected from the third dataset: the number of bytes, packet delay for node, and packet delay. Detailed explanation of the values can be found in [8]. Figure 5 shows a portion of the three values extracted from the CAIDA dataset, which is used in ML.

We consider three scenarios depending on the datasets used for training and testing: I. The datasets of CAIDA and DARPA are mixed and then divided into training and testing sets. II. CAIDA is used for training, whereas DARPA is used for testing. III. Recent dataset is divided into training and testing sets.

# source IP add.	# pkts by the IPs	accum. pkts	# source IP add.	# pkts by the IPs	accm. pkts
14	68	7908	6411	6414	6829094
5	19	1676	7074	10781	11488854
9	65	5740	7107	10801	11509384
11	227	189908	6897	10799	11504953
5	21	1708	7070	10726	11428356
6	19	1640	6794	10740	11444990
3	16	1514	7078	10823	11522233
4	16	1400	6950	10835	11542650
6	45	4790	7064	10749	11455718
9	27	2344	7086	10818	11526527
7	51	3542	6755	9638	10243526
11	252	21645	7067	9574	10175817
7	240	16888	7067	10519	11202618
15	282	20813	6515	9585	10191804

(A) (B)

FIGURE 5 A portion of the values used for ML: (A) Normal state (B) Attack state

The data are not shuffled for BNN, while shuffled and unshuffled data are used in LSTM. The purpose of considering scenarios I and II in BNN is to investigate the effects of learning former traffic on the analysis of sequential traffic and the effects of learning one dataset on application to another dataset, respectively. The ratio of training and testing sets may affect detection accuracy. During the experiment, we observed that using 80% of dataset for training seems relevant. Therefore, in the following figures for BNN and LSTM (no shuffle), for each dataset, the first 80% and the subsequent 20% of normal and attack traffic are used for training and testing, respectively. Meanwhile, 80% and 20% of total dataset are used for training and testing, respectively, for LSTM (shuffle).

For both learning algorithms, BCT with $\lambda = 0$ and MMT are applied for the extracted values from datasets, as shown in Figure 5. Then, the preprocessed data $\{\mathbf{X}_j\}_{j=1}^{N}$ and corresponding label y_j of $\mathbf{X}_j$ are obtained. $\mathbf{W}_*^{(1)}$ and $\mathbf{b}_*^{(1)}$ are given by random normal, initially, for all *. As binary classification is considered, "*tf.sigmoid*" and "*sigmoid_cross_entropy _with_logits*" are used for BNN and LSTM, respectively, which correspond to (2), and "*tf.reduce_mean*" is used for (3). Two optimizers, gradient decent algorithm and Adam, are used. In addition, SVM with radial basis function (RBF) kernel is executed to compare the performance. In the following tables, B and M are used instead of BCT and MMT, and S and PM are used instead for scenario and performance measure, for simplicity.

5.1 | BNN

Two values of l, 1 and 2, are considered to investigate the effect of the number of layers in the hidden layer for BNN, denoted as Layer1 and Layer2, respectively. It is observed that more layers do not provide performance improvement, whereas they require more training time. This seems to be because the range of

values used, such as the number of accumulated packets, clearly distinguishes the normal state and the attack state. Therefore, we considered only two layers. Learning rates of 10^{-2} and 10^{-1} are used for Layer1 and Layer2, respectively, for any optimizer. Iteration numbers vary depending on the environment. The learning rates and iteration numbers are determined by exhaustive grid search to decrease the cost function rapidly and according to the convergence rates of learning, respectively.

Henceforth, the training environments are distinguished as [scenario, layer, algorithm, preprocessing method], for example, [I, Layer1, GD, BCT]. From the numerical results, the following is observed for scenarios I–II; i) GD (AD) is better than AD (GD) for Layer1 (Layer2) in terms of cost function if the same iteration numbers are used. ii) BCT is better than MMT in terms of accuracies. iii) BCT is better than MMT for Layer1, whereas MMT is better than BCT for Layer2 in terms of training time with similar accuracies. Table 2 presents $\mathrm{Acc}_{\mathrm{tr}}$, $\mathrm{Acc}_{\mathrm{te}}$, T_{tr}, and T_{te} for different environments. It shows that the accuracies of scenarios I–II are higher than those of scenario III. Since MMT has values in the interval [0, 1], the differences between preprocessed data values for MMT are smaller than those for BCT. MMT may require more training time, as shown in Layer1, whereas MMT requires less training time in Layer2 with GD. The results for scenario III show that this method is applicable to recent traffic, and MMT is better than BCT in general for the measures, owing to the time it takes BCT to capture the characteristics of the traffic. This is because the number of bytes is relatively large compared with packet delay for node and packet delay. Table 3 compares two measures, TPR and PPV, for BNN and LSTM (no shuffle). According to the table, BCT is better than MMT for any layer and algorithm in scenarios I–II, whereas MMT seems better in many cases for scenario III, when BNN is applied. Even though *TPR*s are the same in different environments, the predicted values computed using the obtained $\mathbf{W}$ and $\mathbf{b}$ are different for each scenario. Figure 6 shows $S(L(\mathbf{X}_{j,\mathrm{te}}))_1^{(20)}$ using the obtained learning parameters and the corresponding $y_{j,\mathrm{te,pred}}$ of 50 sets of normal traffic and 50 sets of attack traffic from the testing set for [I–II, Layer1, GD, BCT]. It shows that the two scenarios have the same $\{y_{j,\mathrm{te,pred}}\}_{j=1}^{100}$, but different $\{S(L(\mathbf{X}_{j,\mathrm{te}}))_1^{(20)}\}_{j=1}^{100}$ values. That is, even though the *TPR*s are the same for the two scenarios, scenario I detects attack more accurately than scenario II does. The obtained learning parameters that are used for the calculation of $S(L(\mathbf{X}_{j,\mathrm{te}}))_1^{(20)}$ for scenarios I–II are as follows:

$$\mathbf{W}_1^{(20)} = [0.9753, 0.8679,]^{\mathrm{T}}, \mathbf{b}_1^{(20)} = [0.4714] \text{ for I}$$

and

$$\begin{aligned} \mathbf{W}_1^{(20)} &= [0.9390, 0.8762, -0.9756]^{\mathrm{T}}, \\ \mathbf{b}_1^{(20)} &= [0.4696] \text{ for II}, \end{aligned} \tag{11}$$

where "T" represents the transpose of a matrix.

Figures 7 and 8 compare the cost, training accuracy, and test accuracy of [I–III, Layer1, GD/AD, BCT] and [I–II, Layer2, AD, BCT/MMT], respectively, for varying iterations. According to Figure 7, the cost for GD converges rapidly during the first two iterations and approximates zero when four iterations are performed. However, that for AD

TABLE 2 Results for different environments: Accuracies (probability), training and testing times (time unit: second)

Layer	Layer1				Layer2			
Optimizer	GD		AD		GD		AD	
PM	B	M	B	M	B	M	B	M
Scenarios I								
$\mathrm{Acc}_{\mathrm{tr}}$	1.000	0.970	1.000	0.990	1.000	0.990	1.000	0.990
$\mathrm{Acc}_{\mathrm{te}}$	1.000	0.970	1.000	0.990	1.000	0.990	1.000	0.990
T_{tr}	0.211	2.849	0.257	1.697	3.246	0.775	0.359	0.364
T_{te}	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Scenarios II								
$\mathrm{Acc}_{\mathrm{tr}}$	1.000	0.980	1.000	0.980	1.000	0.990	1.000	0.990
$\mathrm{Acc}_{\mathrm{te}}$	1.000	0.840	1.000	0.850	1.000	0.910	1.000	0.910
T_{tr}	0.207	8.169	0.268	0.753	27.320	0.775	0.397	0.332
T_{te}	0.001	0.003	0.001	0.001	0.001	0.001	0.001	0.001
Scenarios III								
$\mathrm{Acc}_{\mathrm{tr}}$	0.800	0.930	0.930	0.930	0.710	0.940	0.930	0.940
$\mathrm{Acc}_{\mathrm{te}}$	0.810	0.950	0.960	0.960	0.770	0.970	0.960	0.970
T_{tr}	3.019	1.187	2.341	2.822	3.641	2.682	3.581	2.858
T_{te}	0.007	0.007	0.005	0.005	0.007	0.007	0.006	0.007

TABLE 3 Results for different environments: TPR, PPV (probability)

Algorithm	**BNN**							
Layer	**Layer1**				**Layer2**			
Optimizer	**GD**		**AD**		**GD**		**AD**	
PM	**B**	**M**	**B**	**M**	**B**	**M**	**B**	**M**
Scenario I								
TPR	1.000	0.988	1.000	0.988	1.000	1.000	1.000	0.988
PPV	1.000	0.975	1.000	1.000	1.000	0.852	1.000	1.000
Scenario II								
TPR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PPV	1.000	0.756	1.000	0.767	1.000	0.852	1.000	0.847
Scenario III								
TPR	0.960	0.940	0.940	0.960	1.000	0.940	0.940	0.940
PPV	0.738	0.903	0.979	0.738	0.505	0.959	0.958	0.959
Algorithm	**LSTM (no shuffle, epochs = 60)**							
Layer	**H1**				**H2**			
Optimizer	**GD**		**AD**		**GD**		**AD**	
PM	**B**	**M**	**B**	**M**	**B**	**M**	**B**	**M**
Scenario I								
TPR	1.000	0.987	1.000	0.987	1.000	0.987	1.000	0.987
PPV	0.560	1.000	1.000	1.000	0.940	1.000	1.000	1.000
Scenario II								
TPR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PPV	0.500	0.509	1.000	0.510	1.000	0.532	1.000	0.510
Scenario III								
TPR	1.000	1.000	0.940	0.940	1.000	1.000	1.000	1.000
PPV	0.505	0.505	0.959	0.979	0.505	0.505	0.505	0.694

converges slowly and approximates zero when more than 20 iterations are performed. As costs decrease, the training accuracies for GD converge fast to one while those for AD require more iterations to achieve the same accuracies. For [I–II, Layer2, AD, BCT/MMT], even though BCT is better than MMT in terms of measures, as seen in Tables 2 and 3, costs for MMT are less than those for BCT, as shown in Figure 7. This is because of the different ranges of preprocessed datasets.

5.2 | LSTM RNN

The number of layers, number of hidden nodes, and epochs are fixed as constants to investigate their effect on performance. Although the constants are not optimal values, those are determined by considering the detection accuracy and observed characteristics of the data during the experiment. To conduct this evaluation, 1 and 10 hidden nodes are considered in one layer and denoted as H1 and H2, respectively, 10 and 60 epochs are used for all environments, respectively, and the BNN value of 10^{-2} is used as the learning rate.

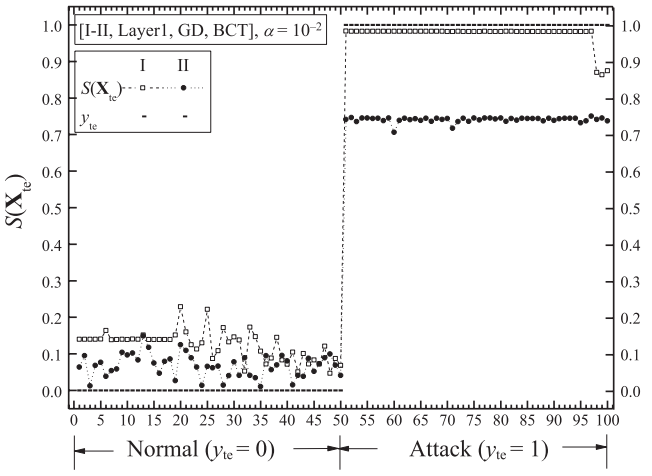

FIGURE 6 Comparison of learning results $S(L(\mathbf{X}_{j,te}))_1^{(20)}$: [I–II, Layer1, GD, BCT]

As shown in Section 5.1, Table 3 indicates that LSTM (no shuffle) with 60 epochs gives similar or even better results for TPRs than those of BNN, which uses an optimal number of iterations; however, the values of PPV are not acceptable. Table 4 compares TPR and PPV for LSTM (shuffle) and LSTM (no shuffle) with 10 hidden nodes and 10 epochs. In this setting, LSTM (shuffle) seems better than LSTM (no shuffle) except TPR for scenario III with MMT. Although some measurement values, such as PPV in scenario III, are not sufficient to accept, they could be improved by increasing the number of epochs. Similar to BNN, BCT seems better than MMT for any number of hidden nodes and algorithms for these measures. Table 5 compares T_{tr} and T_{te} in the same settings as those of Table 4. It shows that LSTM (shuffle) requires more time for training than LSTM (no shuffle) for scenario I, whereas the differences are negligible for other two scenarios. Testing times are almost the same in all scenarios, preprocessing methods, and optimizers, regardless of shuffle. It also shows that LSTM requires more time for training and testing than BNN, as shown in Table 2. Figure 9 compares the

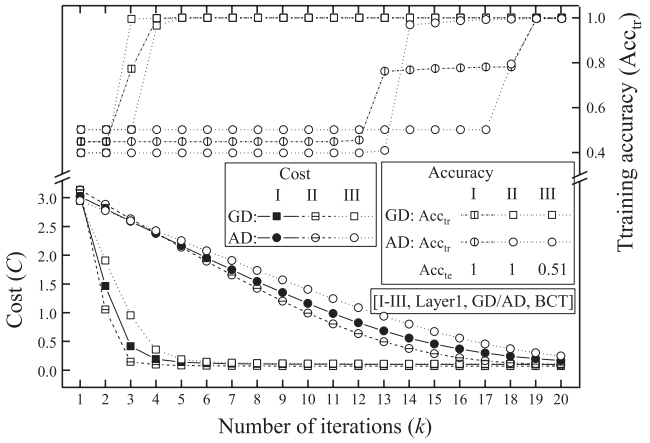

FIGURE 7 Comparison of costs and accuracies: [I–III, Layer1, GD/AD, BCT]

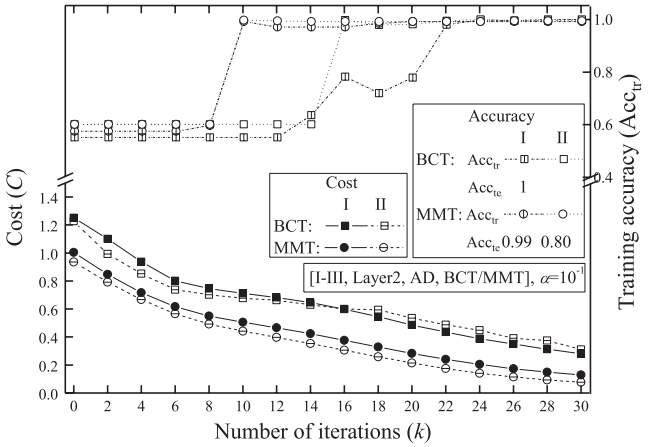

FIGURE 8 Comparison of costs and accuracies: [I–II, Layer2, AD, BCT/MMT]

cost of [I–III, H2, AD, BCT/MMT] for varying epochs when LSTM (shuffle) is applied, showing that the cost of scenario I converges rapidly during the first few epochs for both BCT and MMT, and scenario I converges faster than do the other two scenarios. Figure 10 shows the F1 scores and accuracy scores of [I & III, H2, GD/AD, MMT], for varying training set ratios. Although a small number of epochs (10 epochs) are executed, using more than 50% of dataset for training gives acceptable scores except in the case of [III, H2, GD, MMT].

The obtained results can be compared with the existing results. Table 6 summarizes the comparison of performance measures for scenario I with GD and SVM, scenario III with AD, and the existing results. A detailed explanation of each proposed algorithm can be found in the corresponding article. As shown in the table, scenario I with GD (BNN and LSTM (shuffle) both) is better than the existing results for the measures. Because the dataset used in scenario III was provided by the authors of [8], the results of scenario III can be compared with their results, and the performance of LSTM can be compared with those of [14,15], [18], and [20]. For instance, the measure values in the "[8] (2016)" row in the table are computed by using the values of the normal and UDP Flood in the confusion matrix in Table 6 in [8], and the normal and UDP Flood values in the confusion matrix in Table 6. Although the MLP setting is different from ours, using 16 nodes, a maximum of 500 epochs, and a learning rate of 0.3, and a small portion of their dataset is used in our experiment, similar results are obtained with smaller epochs. The testing accuracy and TPR of [14] are only better than LSTM (shuffle) of scenario III. This may be due to using different hyperparameters and datasets. For instance, [14] used four layers and 64 neurons for LSTM with ISCX 2012, which contains data from 2010. In addition, the training and testing times can be compared with the results in [11]. For instance, those values are 1.42 and 0.35 second in the reference while our results are 0.211 and 0.001 second for [II, Layer1, GD, BCT], respectively, as shown in Table 2. More comparison of existing training algorithm performances, such as those from decision tree, Naïve Bayes, random forest, and K-means, can be found in several articles (e.g. [13,16]). It is difficult to directly compare our results with other recent results because the experimental settings and data used are different. Based on the comparison, however, the obtained values are believed to be reliable, and the preprocessing methods, hyperparameters, and feature extraction algorithms used in this study are appropriate for DDoS attack detection, including attack traffic with recent characteristics.

From the experimental process, the following is obtained: For BNN, (a) single layer with appropriate learning rate is better than multi-layer in terms of training time and cost, (b) learning rate depends on several factors, such as the optimizer used, and has a crucial role for rate of convergence in the training process and performance measure values. (c)

TABLE 4 Results for different environments with 10 hidden nodes and 10 epochs: TPR, PPV (probability)

Algorithm	LSTM (shuffle, H2)				LSTM (no shuffle, H2)			
Optimizer	GD		AD		GD		AD	
PM	B	M	B	M	B	M	B	M
Scenario I								
TPR	1.000	0.987	1.000	0.987	1.000	0.987	1.000	0.987
PPV	1.000	0.987	1.000	0.987	0.560	0.987	0.987	0.987
Scenario II								
TPR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PPV	1.000	0.745	1.000	0.721	0.500	0.745	1.000	0.538
Scenario III								
TPR	1.000	0.940	1.000	0.94	1.000	1.000	1.000	1.000
PPV	0.505	0.734	0.685	0.959	0.505	0.505	0.505	0.505

TABLE 5 Results for different environments: T_{tr} and T_{te}

Algorithm	LSTM (shuffle, H2)				LSTM (no shuffle, H2)			
Optimizer	GD		AD		GD		AD	
PM	B	M	B	M	B	M	B	M
Scenario I								
T_{tr}	11.268	11.702	12.062	11.725	9.751	10.235	10.765	11.116
T_{te}	0.020	0.010	0.012	0.011	0.011	0.010	0.011	0.012
Scenario II								
T_{tr}	8.645	8.035	8.397	8.166	8.036	8.038	8.608	8.530
T_{te}	0.028	0.010	0.009	0.010	0.011	0.009	0.020	0.008
Scenario III								
T_{tr}	9.533	10.808	9.355	10.548	9.238	9.611	10.021	9.315
T_{te}	0.011	0.011	0.011	0.009	0.011	0.014	0.009	0.012

some measures, such as TPR for LSTM RNN, are better than those for BNN for recent datasets, whereas LSTM RNN requires a longer time than BNN for attack detection. Based on the results, it can be concluded that DDoS attacks can be detected fast with high accuracy when the learning rate and learning algorithms are chosen appropriately.

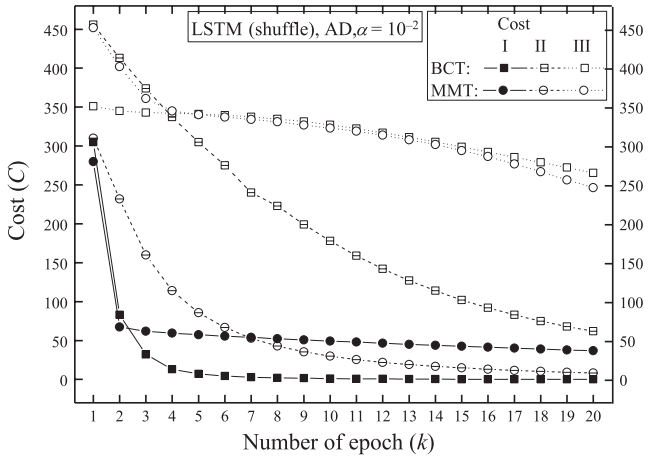

FIGURE 9 Comparison of costs: [I–III, H2, AD, BCT/MMT]

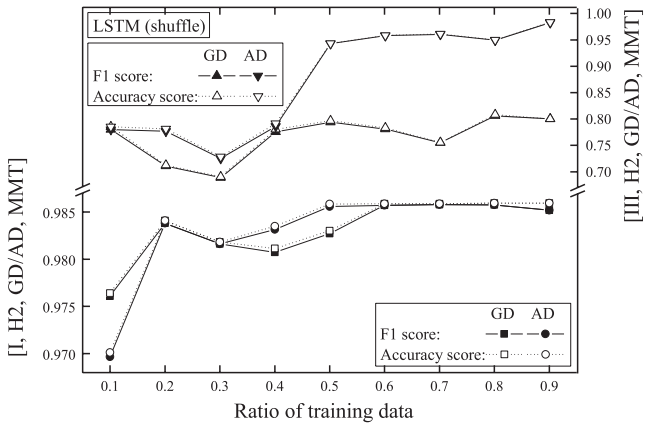

FIGURE 10 Comparison of F1 and accuracy scores: [I & III, H2, GD/AD, MMT]

TABLE 6 Comparison with existing results

Comparison	Data/Preprocess	Algorithm	Acc_{te}	TPR	PPV
I	CAIDA + DARPA /BCT	BNN(GD) (opt. iteration)	1.0000	1.0000	1.0000
		SVM(RBF)	1.0000	0.9750	1.0000
I	CAIDA + DARPA /BCT	LSTM (GD, shuffle) (10 eps)	1.0000	1.0000	1.0000
III	UDP/MMT	BNN(AD) (opt. iteration)	0.9700	0.9400	1.0000
III	UDP/MMT	LSTM (AD, shuffle) (10 eps)	0.9400 (acc. score)	0.9400	0.9600
[11] (2013)	Mixed(CAIDA) /MMT	NFBoost +cost min.	0.9880	N/A	N/A
[13] (2015)	CAIDA /MMT	Adaboost +RF	0.9989	N/A	N/A
[7] (2016)	KDD'99	proposed	0.9329	0.9186	N/A
[8] (2016)	UDP	MLP (500 max. eps)	0.9900 (acc. score)	0.9000	1.0000
[26] (2017)	CAIDA	proposed	1.0000	0.9995	N/A
	DARPA		1.0000	1.0000	
[22] (2017)	KDD'99 /Normalization	proposed	0.9998	0.9940	0.9984
[14] (2017)	ISCX 2012/ Normalization	LSTM	0.9799	0.9788	0.9810
		CNN LSTM	0.9589	0.9420	0.9753
		GRU	0.9679	0.9501	0.9841
[15] (2018)	KDD'99	RNN	0.9855	N/A	N/A
		PCA RNN	0.9859		
		PCA SVM	0.9853		
		LSTM	0.9850		
[23] (2018)	Collected /MMT	SAE	0.7721 – 0.9703	0.6465 – 0.9535	N/A
[18] (2019)	CTU-13 Botnet & ISCX 2012 IDS/ binary string	LSTM (hidden layer: 0–2)	0.8067 – 0.9367	N/A	N/A
[20] (2019)	AzScienceNet /Normalization	Improved CNN (500 eps)	0.7744	0.8455	0.8923
		Improved LSTM (140 eps)	0.6974	0.7522	0.8865

6 | CONCLUSIONS

In this study, supervised learning algorithms are applied to three different traffic sets including DDoS attacks. BNN and LSTM RNN are considered for feature extraction via TensorFlow. The preprocessing methods, optimizers, and network architectures that are appropriate detecting DDoS attacks are investigated. For different environments, the hyperparameters are appropriately determined for fast learning convergence, and the corresponding learning parameters providing high accuracy of detection are obtained. As the data accumulate, DDoS attacks can be detected by updating the learning parameters. The obtained performance measures provide a criterion in which the detection equipment is implemented.

This method could be applied to detect other abnormal behavior and extend to detect several types of attacks from a mixture of traffic, including new types of attacks that result from network evolution. The selections of traffic components that characterize the new types of attacks have to be studied

in advance. Finding the optimal values of hyperparameters related to ML techniques needs further study. In addition, more advanced technologies of ML that pertains to time series data and newly developed statistical models for time series need to be compared.

ACKNOWLEDGMENTS

Support for CAIDA's Internet Traces is provided by the National Science Foundation, the US Department of Homeland Security, and CAIDA Members. The author also appreciates the sharing of datasets used in this paper by MIT Lab and the authors of [8].

ORCID


Meejoung Kim https://orcid.org/0000-0002-8081-0489

REFERENCES

1. S. Abraham and S. Nair, *Cyber security analytics: a stochastic model for security quantification using absorbing markov chains*, J. Commun. **9** (2014), no. 12, 899–907.
2. X. Liang and Y. Xiao, *Game theory for network security*, IEEE Commun. Survey and Tuts. **15** (2013), no. 1, 472–486.
3. A. Fielder et al., *Decision support approaches for cyber security investment*, Decis Support Syst. **86** (2016), 13–23.
4. M. Kim, *Game theoretic approach of eavesdropping attack in millimeter-wave-based WPANs with directional antennas*, Wireless Netw. **25** (2019), no. 6, 3205–3222.
5. Y.-C. Wu et al., *DDoS detection and traceback with decision tree and grey relational analysis*, Int. J. Ad Hoc Ubiquitous Comput. **7** (2011), no. 2, 306–314.
6. T. Subbulakshmi et al., *Detection of DDoS attacks using Enhanced Support Vector Machines with real time generated dataset*, in Proc. Int. Conf. Advance Comput., Chennai, India, Dec. 2011, pp. 17–22.
7. C. Guo et al., *A two-level hybrid approach for intrusion detection*, Neurocomput. **214** (2016), 391–400.
8. M. Alkasassbeh et al., *Detecting distributed denial of service attacks using data mining techniques*, Int. J. Adv. Comput. Sci. Applicat. **7** (2016), no. 1, 436–445.
9. X. Zanget al., Botnet detection through fine flow classification, CSE Dept Technical Report, no. CSE11–001, 2011.
10. P. Salunkhe and M. Shishupal, *Denial-of -service attack detection using KDD*, Int. J. Applicat. Innovation Eng. Manag. **4** (2015), no. 3, 1–5.
11. P. A. R. Kumar and S. Selvakumar, *Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems*, Comput. Commun. **36** (2013), 303–319.
12. X. Ma and Y. Chen, *DDoS detection method based on chaos analysis of network traffic entropy*, IEEE Commun. Lett. **18** (2014), no. 1, 114–117.
13. R. Robinson and C. Ciza Thomas, *Thomas, Ranking of machine learning algorithms based on the performance in classifying DDoS attacks*, in Proc. IEEE Recent Adv. Intell. Computat. Syst., Trivandrum, India, Dec. 2015, pp. 10–12.
14. X. Yuan, C. Li, and X. Li, *Deepdefense: identifying ddos attack via deep learning*, in Proc. IEEE SMARTCOMP, Hong Kong, China, 2017, pp. 1–8.
15. Q. Li et al., *DDoS Attacks Detection using Machine Learning Algorithms*, in: G. Zhai, J. Zhou, P. An, X. Yang (eds) Digital TV and Multimedia Communication. IFTC 2018. Communications in Computer and Information Science, vol. **1009**. Springer, Singapore, pp 205–216.
16. Z. He, T. Zhang, and R. B. Lee. *Machine learning based DDoS attack detection from source side in cloud*, in Proc. IEEE ICCSCC, New York, NY, 2017, pp. 114–120.
17. A. Verma, M. Arif, and M. S. Husain, *Analysis of DDoS attack detection and prevention in cloud environment: A review*, Int. J. Adv. Research Comput. Sci. **9** (2018), 107–113.
18. R. Priyadarshini and R. K. Barik. *A deep learning based intelligent framework to mitigate DDoS attack in fog environment*, J. King Saud Univ.–Comput. Inf. Sci. (2019), published on line. https://doi.org/10.1016/j.jksuci.2019.04.010
19. C. Li et al., *Detection and defense of ddos attack–based on deep learning in openflow-based sdn*, Int J. Commun. Syst. **31** (2018), 1–15.
20. R. M. Alguliyev, R. M. Aliguliyev, and F. J. Abdullayeva, *The improved LSTM and CNN Models for DDoS attacks prediction in social media*, Int. J. Cyber Warfare Terrorism. **9** (2019), no. 1, 1–16.
21. N. Sharma, A. Mahajan, and V. Mansotra, *Machine learning techniques used in detection of DoS attacks: a literature review*, Int. J. Adv. Research Comput. Sci. Softw. Eng. **6** (2016), no. 3, 100–105.
22. B. Jia et al., *A DDoS attack detection method based on hybrid heterogeneous multiclassifier ensemble learning*, Hindawi J. Elect. Comput. Eng. **2017** (2017), 4975343:1–9.
23. M. E. Aminanto et al., *Deep abstraction and weighted feature selection for Wi-Fi impersonation detection*, IEEE Trans. Inf. Forensics Secur. **13** (2018), no. 3, 621–635.
24. T. George, The next big cybercrime vector: Social media, Security Week (2014) Retrieved from https://www.securityweek.com/next-big-cybercrime-vector-social-media
25. T. Peng, C. Leckie, and K. Ramamohanarao. *Survey of network-based defense mechanisms countering the DoS and DDoS problems*, ACM Comput. Surveys. **39** (2007), no. 1, 3:1–42.
26. N. Hoque, H. Kashyap, and D. K. Bhattacharyya, *Real-time DDoS attack detection using FPGA*, Comput. Commun. **110** (2017), no. C, 48–58.
27. H. Choi and H. Lee, *Identifying botnets by capturing group activities in DNS traffic*, Comput. Netw. **56** (2012), 20–33.
28. S. Suresh and N. S. Ram, *A review on various DPM trace back schemes to detect DDoS attacks*, Indian J. Sci. Technol. **9** (2016), no. 47, 1–8.
29. J. Katerina, K. Argyraki, and D. R. Cheriton, *Active internet traffic filtering: real-time response to denial-of-service attacks*, IEEE/ACM Trans. Netw. **17** (2009), no. 4, 1284–1297.
30. F. Anjum, D. Subhadrabandhu, and S. Sarkar, *Signature based Intrusion Detection for Wireless Ad-Hoc Networks: A Comparative study of various routing protocols*, in Proc. Veh. Technol. Conf., Orlando, FL, USA, Oct. 2003, pp. 2152–2156.
31. S. M. T. Nezhad, M. Nazari, and E. A. Gharavol, *A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks*, IEEE Commun. Lett. **20** (2016), no. 4, 700–703.
32. Q. Yan and F. R. Yu, *Distributed denial of service attacks in software-defined networking with cloud computing*, IEEE Commun. Mag. **53** (2015), no. 4, 52–59.

33. G. Somani et al., *Scale inside-out: rapid mitigation of cloud DDoS attacks*, IEEE Trans. Dependable Secure Comput. **15** (2018), no. 6, 1–14.
34. S.-S. Alireza et al., *Taxonomy of distributed denial of service mitigation approaches for cloud computing*, J. Netw. Comput. Applicat. **58** (2015), 165–179.
35. S. Behal and K. Kumar, *Measuring the impact of DDoS attacks on Web Services - A realtime experimentation*, Int. J. Comput. Sci. Inf. Security. **14** (2016), no. 9, 323–330.
36. P. Probst, A.-L. Boulesteix, and B. Bisch, *Tunability: importance of hyperparameters of machine learning algorithms*, J. Mach. Learn. Research. **20** (2019), 1–32.
37. J. Kimet al., CHOPT: *automated hyperparameter optimization framework for cloud-based machine learning platforms*, 2018, arXiv: 1810.03527v2.
38. J. Wu et al, *Hyperparameter optimization for machine learning models based on bayesian optimization*, J. Electron. Sci. Technol. **17** (2019), no 1, 26–40.
39. D. H. Deshmukh, T. Ghorpade, and P. Padiya. *Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset*, in Proc. Int. Conf. Commun. Inf. Ccomput. Technol., Mumbai, India, Jan. 2015, pp. 1–6.
40. CAIDA: Index of/datasets/security/ddos-20070804 [Online] Available from: https://data.caida.org/datasets/security/ddos-20070804/
41. MIT Lincoln Lab. Available from: https://www.ll.mit.edu/ideval/data/1998data.html [last accessed March 22, 2019].
42. Dataset (used for submain) final dataset.rar Available from: https://www.researchgate.net/publication/292967044_Dataset_Detecting_Distributed_Denial_of_Service_Attacks_Using_Data_Mining_Techniques
43. J. W. Osborne, *Improving your data transformations: Applying the Box-Cox transformation, Practical Assessment*, Research Evaluation **15** (2010), no. 12, 1–9.
44. D. P. Kingma and J. L. Ba, *ADAM: A method for stochastic optimization*, in Proc. Int. Conf. Learn. Representations, San Diego, USA, 2015, 1–15.
45. A. Azzouni and G. Pujolle. *A long short-term memory recurrent neural network framework for network traffic matrix prediction*, arxiv 1705.05690, v3 Thu, 8 Jun 2017.

AUTHOR BIOGRAPHY

Meejoung Kim received her BS in mathematics from Korea University, Seoul, Rep. of Korea, in 1986; an MS in mathematics from both Korea University and the University of Minnesota, Twin Cities, Minneapolis, MN, in 1988 and 1993, respectively; and her PhD in mathematics from Korea University in 1996. From 1993 to 1999, she worked as a lecturer and research fellow in the Department of Mathematics at Korea University. From 2000 to 2004, she worked as a research fellow and an assistant professor with Brain Korea 21 Information Technology. Since 2004, she has been a professor in the Research Institute for Information and Communication Technology, Korea University, Seoul, Rep. of Korea. She teaches probability theory and complex analysis. Her research interests include mm-wave WPANs, wireless communication systems, wireless security, white noise analysis, and machine learning.