TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 09/15 pp484–495
DOI: 10.26599/TST.2020.9010022
Volume 26, Number 4, August 2021

Intrusion Detection System Using Voting-Based Neural Network

Mohammad Hashem Haghighat* and Jun Li*

Abstract: Several security solutions have been proposed to detect network abnormal behavior. However, successful attacks is still a big concern in computer society. Lots of security breaches, like Distributed Denial of Service (DDoS), botnets, spam, phishing, and so on, are reported every day, while the number of attacks are still increasing. In this paper, a novel voting-based deep learning framework, called VNN, is proposed to take the advantage of any kinds of deep learning structures. Considering several models created by different aspects of data and various deep learning structures, VNN provides the ability to aggregate the best models in order to create more accurate and robust results. Therefore, VNN helps the security specialists to detect more complicated attacks. Experimental results over KDDCUP'99 and CTU-13, as two well known and more widely employed datasets in computer network area, revealed the voting procedure was highly effective to increase the system performance, where the false alarms were reduced up to 75% in comparison with the original deep learning models, including Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).

Key words: deep learning; Voting-based Neural Network (VNN); network security; Pearson correlation coefficient

1 Introduction

Computer network plays an important role nowadays. Various internet-based services, like voice over IP, internet banking, Point to Point (P2P) file sharing, online gaming, and so on, having been used every day. However, the number of network malicious activities are increasing dramatically[1]. According to McAfee, "ransomware attacks", as a type of malware aiming at blocking the access of a user to its computer until specific amount of money is paid, have been increased by 118% during 2019[2].

Dozens of behavior-based detection techniques have been proposed to protect networks from such attacks. The key challenge of these methods is to lower the false alarms using machine learning algorithms[3–14].

Nowadays, deep learning provides a suitable infrastructure to automatically learn features from raw data. This advantage enables the scientists to employ deep learning techniques in different areas, like natural language processing, image and voice recognition, and computer networks.

Generally, various types of deep learning models have been developed, including Deep Neural Network (DNN), Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Boltzmann Machine (BM), and Stacked Auto-Encoder (SAE).

RNNs enable previous outputs to be used for the input of the next step as depicted in Fig. 1. Since RNNs are suitable for time series data, they are widely utilized in network anomaly-based detection techniques in the literature.

J. Kim and H. Kim[16] applied RNN to Intrusion Dection System (IDS) and achieved magnificent results on KDDCUP'99. They improved their method by employing Long Short-Term Memory (LSTM) as the learning engine which the performance test showed the system was suitable for IDSes[17]. Yin et al.[18] and

• Mohammad Hashem Haghighat and Jun Li are with the Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: l-a16@mails.tsinghua.edu.cn; junl@tsinghua.edu.cn.

∗ To whom correspondence should be addressed.

Manuscript received: 2020-03-24; revised: 2020-07-02; accepted: 2020-07-10

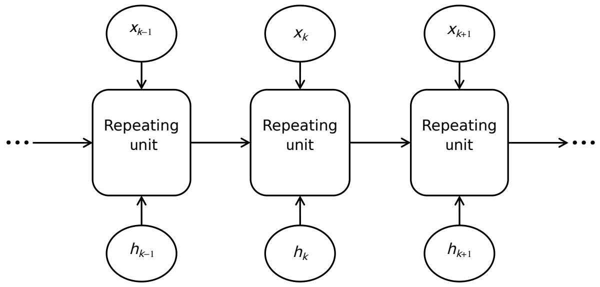


Fig. 1 RNN architecture[15].

Althubiti et al.[19] compared the performance of RNN with traditional machine learning methods, including naive Bayes, random forest, and Support Vector Machine (SVM), using KDDCUP'99 in both multi-class and binary classifiers and revealed RNN overwhelmed all the traditional methods well. A Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) was proposed by Tang et al.[20] with the performance of 89% on KDDCUP'99 using only 6 raw features.

CNN is a special deep learning architecture firstly developed for image recognition problem. However, Yao et al.[21] proposed a CNN-based method to detect time-delayed attacks, and reported that the method was highly accurate for DARPA'98 dataset. Wu et al.[22] employed CNN in order to select traffic properties automatically from raw dataset. They evaluated the method by KDDCUP'99 and argued that the method performs better in terms of performance and false alarm rate compared to the conventional standard algorithms.

SAE is a specific type of neural network with exactly the same size output of its input. The main goal of SAE is to reconstitute of the output from the input. Figure 2 depicts the SAE architecture where the input is compressed and then decompressed to compute the output.

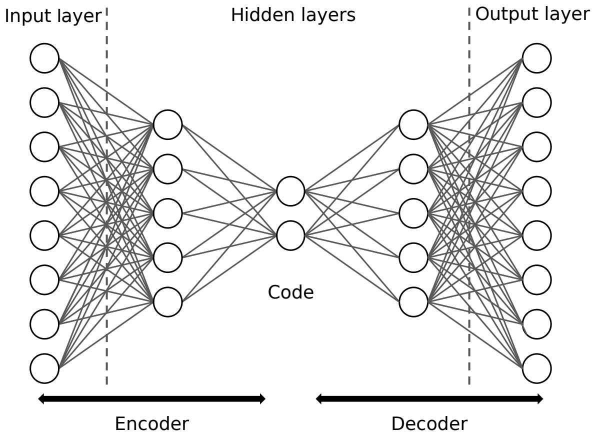


Fig. 2 SAE architecture.

Aminanto and Kim[23] applied SAE as a classifier on KDDcup'99 dataset and presented four different IDSes: application layer IDS, transport layer IDS, network layer IDS, and data link layer IDS. Javaid et al.[24] used SAE to learn features from NSLKDD.

Farahnakian and Heikkonen[25] proposed Deep Auto Encoder (DAE) to extract features from high dimensional data. They achieved more than 97% detection precision in case of using 10% KDDCUP'99 dataset as test case.

BM is a type of stochastic RNN to make decisions concerning being either on or off. BM provides the ability to simply learn systems and interesting features from datasets having binary labels[26].

A multi-layer Denial of Service (DoS) attack detection technique based on Deep Boltzmann Machine (DBM) was provided by Gao et al.[27] The authors argued that their method gained better precision on KDDCUP'99 compared to SVM and simple Artificial Neural Network (ANN). Zhang and Chen[28] sped up the training time by combining SVN, BM, and Deep Belief Network (DBN). Alrawashdeh and Purdy[29] achieved 97.9% precision on 10% KDDCUP'99 dataset as the test case. Recently Vinayakumar et al.[30–34] provided a comprehensive study of various CNN, LSTM, CNN-LSTM, CNN-GRU, and DNN to select the optimal network architecture using KDDCUP'99 and NSLKDD datasets.

Haghighat et al.[35] also developed a sliding window-based deep learning technique (called SAWANT) which achieved 99.952% accuracy on CTU-13 dataset. The authors used only 1%–10% CTU-13 dataset as training to conduct their tests.

The aforementioned methods took the advantage of deep learning to detect network malicious activities. Although their performance was considerable, aggregating different deep learning models provides the capability to utilize the strength of each model and detect attacks incredibly more efficient.

In this paper we propose "Voting-based Neural Network (VNN)" as a general infrastructure voting-based mechanism to aggregate and take the advantages of any kinds of deep learning algorithms. In other words, several deep learning-based models can be created by the state-of-the-art techniques with different performance. Giving test data, VNN provides a procedure to perform a weighted voting function on the most suitable models to achieve higher accurate results. Due to only selecting and aggregating the best models for each test sample, VNN incredibly boosted the system accuracy. Experimental results proved our argument, as the false alarms were reduced up to 75%.

Table 1 summarized all the relevant acronyms employed throughout the paper.

The paper is structured as follows. In Section 2, an overview of VNN is explained. Then, VNN is deeply studied by two well-known KDDCUP'99 and CTU-13 datasets in Sections 3 and 4, using two different configurations: high and low accuracies, respectively. Finally, in Section 5, the paper is concluded and future research plans are explained.

2 Voting-Based Neural Network

Voting-based neural network is a general infrastructure to create several models using different aspects of data or various types of deep learning architectures, and merging them, aiming at increasing the system performance.

As illustrated in VNN architecture (Fig. 3), several inputs are extracted from the original data to be modeled by various kinds of deep learning techniques, like DNN, CNN, RNN, SAE, and so on. As a result, in the prediction phase, a heuristic function, called "voting engine", processes all the models to select the best candidates in a way to minimize the errors. The chosen models perform voting procedure in order to predict test data label. Algorithm 1 describes the whole VNN procedure in detail.

In the next two sections different case studies on well known KDDCUP'99 and CTU-13 datasets are presented to make the voting procedure clearer.

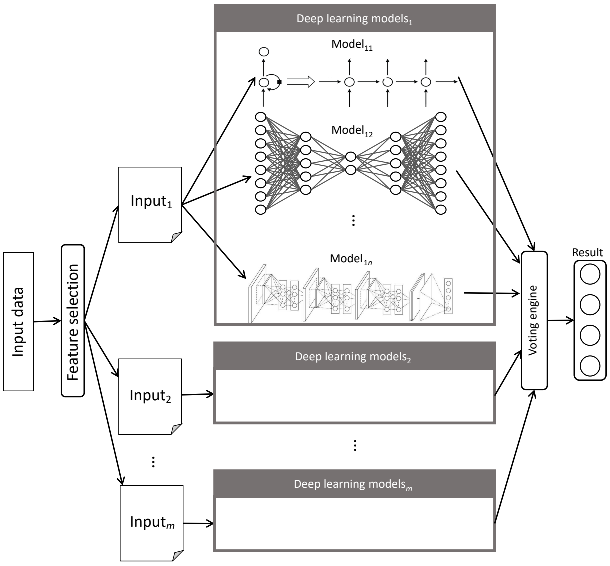


Fig. 3 VNN architecture.

Algorithm 1 VNN whole procedure

```
input1: $\text{train}_{\text{data}} = \{F_1, F_2, \ldots, F_l\}$ //input train data
input2: $\text{test}_{\text{data}} = \{F'_1, F'_2, \ldots, F'_l\}$ //input test data
        where $F_i = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ //k different attributes
input3: $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ //n different models
output: prediction result
//initialization
$\leftarrow$ {} //empty set as training data
$\omega \leftarrow$ {} //empty set as testing data
$\Delta \leftarrow$ {} //empty set as prediction results
$\Xi \leftarrow$ {} //empty set as voting candidates
//selecting n different train and test features
vectors with randomely chosen attibutes
for $i \leftarrow$ range(1, n)
    $A \leftarrow$randomely select $i$ attributes
    $\Psi_i \leftarrow$selectattributes(train, $A$)
    $\Omega_i \leftarrow$selectattributes(test, $A$)
end
for each models $\theta_i$, train data$_j$, and test data $\omega_j$
    model$_{ij} \leftarrow$train($\theta_i$, $\psi_j$) //train model
    $\Delta_{ij} \leftarrow$predict(model$_{ij}$, $\omega_j$) //prediction result
end
$\Delta' \leftarrow$select best voting candidates
result $\leftarrow$vote($\Delta'$)
return result
```

3 Case Study 1: KDDCUP'99

KDDCUP'99[36] is the mostly used dataset to evaluate anomaly-based detection systems[37]. The dataset was built based on DARPA'98 project[38] and contains about 4.9 million records, including 41 different features with normal and four attack types (denial of service, user to root, remote to local, and probing) labels. Hereafter, Tavallaee et al.[39] removed the duplicated records of KDDCUP'99 to create NSLKDD dataset. Figure 4 shows the evolution of NSLKDD dataset[40].

Table 1 Acronyms used through the paper.

Acronym	Expression
VNN	Voting-based Neural Network
DDoS	Distributed Denial of Service
ANN	Artificial Neural Network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
BM	Boltzmann Machine
SAE	Stacked Auto-Encoder
SVM	Support Vector Machine
P2P	Point to Point

3.1 Voting procedure

The main idea behind VNN is to make a general infrastructure to create several models using different

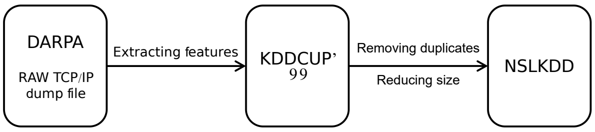


Fig. 4 Evolution of NSLKDD dataset.

deep learning approaches or data aspects. Then, given a test sample, select those models who likely more suitable to find the accurate label.

Definition 1 Let n be the number of models. Uncertainty factor γ_i of the i-th model is defined according to the following equation:

$$\gamma_i = 1 - \rho_i \tag{1}$$

where ρ_i is the probability of the output layer achieved by the i-th model.

The below procedure is defined to select k best candidate models of the voting procedure.

- Considering ζ_i as the accuracy of i-th model reported by the system training procedure, normalize all ζ values according to "normal distribution equation" provided by Eq. (2) in the following[41]:

$$f(x, \mu, \delta) = \frac{1}{\sqrt{2\pi\delta}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\delta^2}} \tag{2}$$

- Assuming λ as "Unsatisfied Models Threshold (UMT)", remove all the models whose normalized accuracy are less than λ.
- Consider "Total Uncertainty (TU)" threshold as ϵ.
- Sort all the remained models based on their uncertainty factors in ascending order.
- Select the models until the total sum of uncertainty factor (γ_i) is less than ϵ.
- Perform the majority voting mechanism on the selected models.

Algorithm 2 describes the procedure in detail.

Algorithm 2 Multi-class output best model selection

```
input1: $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ //uncertainety factors of $n$ models
input2: $Z = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$ //accuracy of the models
input3: $\epsilon$ //total uncertainty threshold
input4: $\lambda$ //unsatisfied model threshold
output: $\Delta$, as set of $k$ best models
//initializing the voting parameters
$E \leftarrow 0$ //total sum of uncertainty factors
$\Delta \leftarrow \{\}$ //inittializing the output
$M \leftarrow \{\}$ //inittializing the set of satisfies models
for each $\zeta_i \in Z$
    $n_i \leftarrow$ normalize($\zeta_i$)
    if $n_i > \lambda$
        //adding corresponding uncertainty factor to $M$
        $M \leftarrow$ add($\gamma_i$)
    end
end
$M_{\text{sorted}} \leftarrow$ sort($M$) //sorting the models
while true
    $E \leftarrow E +$ pop($M_{\text{sorted}}$)
    if $E > \epsilon$
        break
    else
        add corresponding model to $\Delta$
    end
end
return $\Delta$
```

3.2 Experimental result

Several test cases were conducted on KDDCUP'99 using different deep learning architectures, including CNN, LSTM, GRU, CNN-LSTM, and DNN models. In order to highlight the efficiency of voting mechanism, we configured the hyper parameters of these deep learning techniques using two different approaches to see the impact of the voting procedure on different situations:

(1) Achieving highly accurate results (performance> 99%).

(2) Having lots of false alarms (55%<performance< 80%).

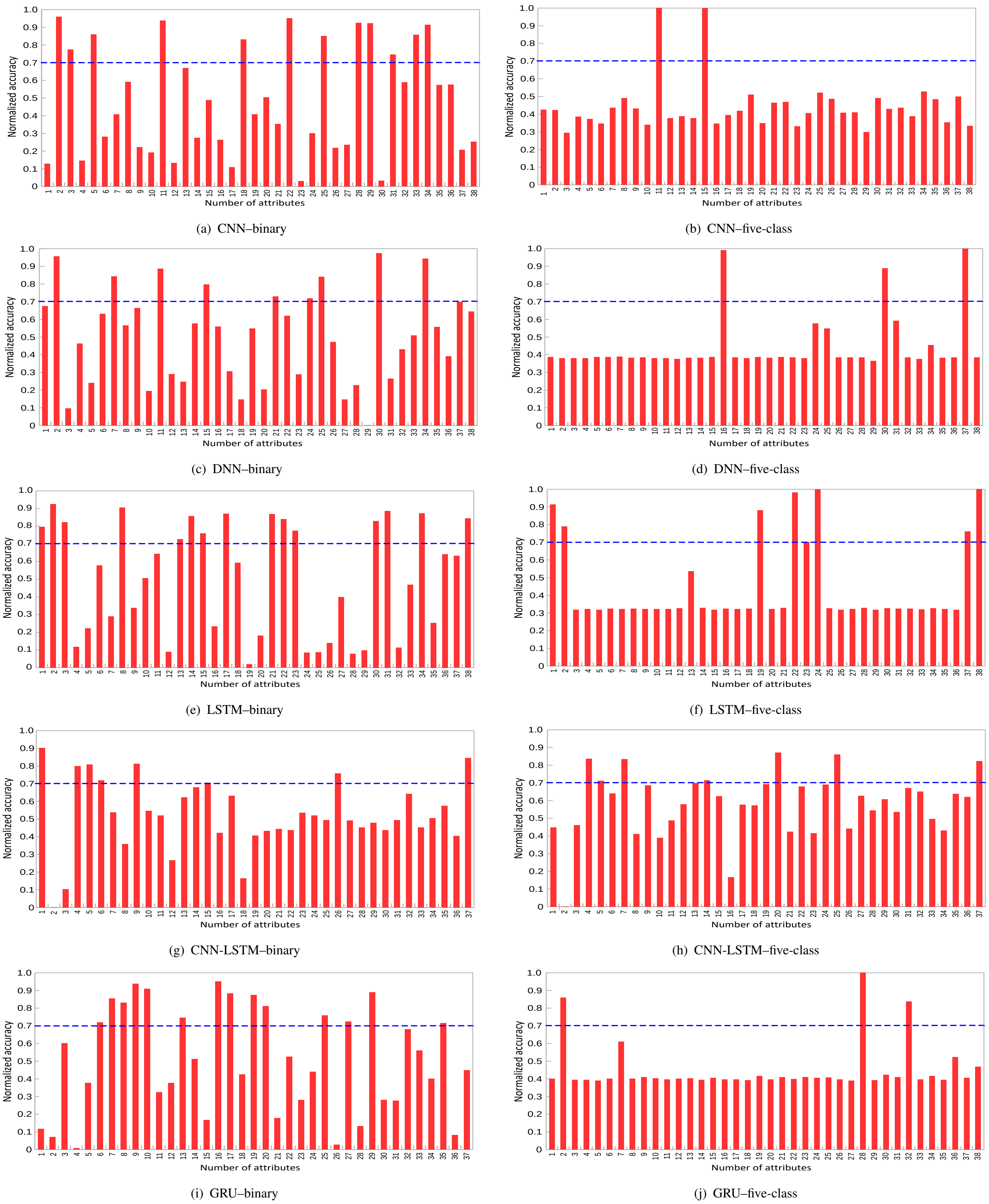
Table 2 describes the models' hyper parameters configuration in detail.

Table 2 Hyper parameters used to test KDDCUP'99.

Hyper parameter	Value
Train size	90%
Test size	10%
Dropout	0.5
Batch input	On
Activation function	Relu
Layers number of CNN	4
Layers number of LSTM	2
Layers number of CNN-LSTM	4
Layers number of DNN	2
Layers number of GRU	2
Number of input attributes	37
Number of input subsets	38
Output	Binary and five-class
UMT	0.7
TU	0.5

Generally, 90% of KDDCUP'99 datasets were chosen to train the models, while the rest of 10% were used for testing. In addition, 38 different training and testing datasets were generated from the input data, in which each dataset includes 37 random KDDCUP'99 attributes. We conducted binary classification as our highly accurate test, while the less accurate tests were performed based on five-class classifier. Figure 5 depicts the accuracy reported by the system during the training phase.

As illustrated in Fig. 5, 0.7 was chosen for UMT where all the models with less normalized accuracy values than



(a) CNN–binary

(b) CNN–five-class

(c) DNN–binary

(d) DNN–five-class

(e) LSTM–binary

(f) LSTM–five-class

(g) CNN-LSTM–binary

(h) CNN-LSTM–five-class

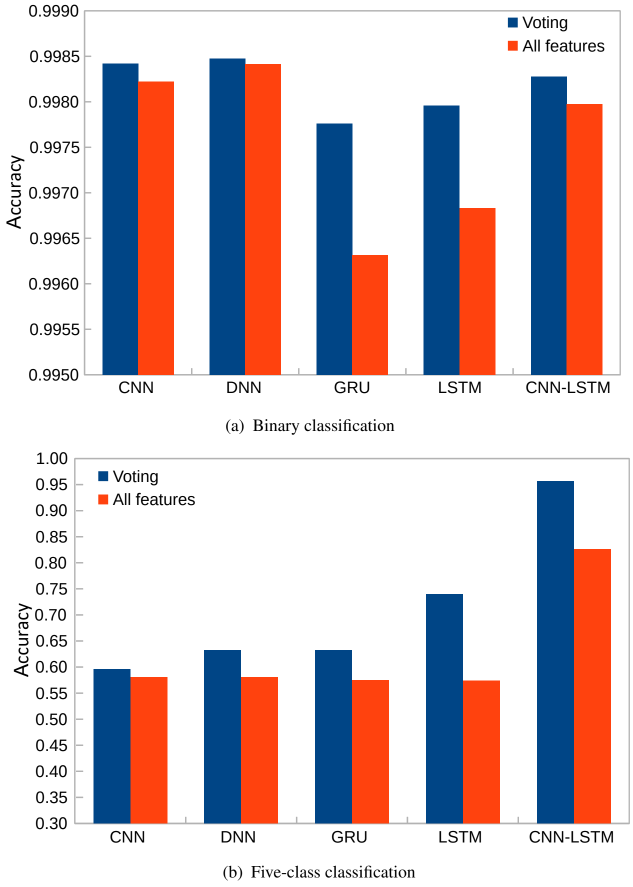
(i) GRU–binary

(j) GRU–five-class

Fig. 5 Normalized form of model accuracy (The blue dashed lines show UMT).

UMT were removed.

The voting procedure was conducted over the remained models and the result was depicted by Fig. 6. The results proved that VNN increased the true responses magnificently in both higher and less accurate deep learning structures. VNN resolved 708 errors out of 1804 (more than 39%) for binary classification-based GRU architecture, and 63 675 false alarms out of about 85 000



(a) Binary classification

(b) Five-class classification

Fig. 6 System accuracy: voting-based vs. normal-based using KDDCUP'99 dataset.

(around 75%) for five-class classification-based CNN-LSMT models. The detailed number of false alarms and their correction rates were explained in Table 3.

We also performed the voting procedure over all the models created by any deep architectures, in which the performance result is summarized in Tables 4 and 5.

Different measurements of the experiment, including False Positive Rate (FPR), False Negative Rate (FNR), Accuracy, Precision, Recall, and F_Score are computed in Table 6. These values were achieved by Eqs. (3) – (8) in the following:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3}$$

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{All Data}} \tag{7}$$

$$\text{F\_Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

where FP, FN, TP, and TN denote False Positive, False Negative, True Positive, and True Negative, respectively.

The result proved that VNN achieved higher accuracy compared to any deep learning structures for both binary and five-class classifiers efficiently. Figure 7 compares VNN with DNN, CNN, LSTM, CNN-LSTM, and GRU methods.

Table 3 KDDCUP'99 error correction.

	Method	Number of errors	Number of corrections	Correction rate (%)
Binary	DNN	777	29	3.73
	CNN	872	97	11.12
	LSTM	1551	551	35.53
	CNN-LSTM	993	148	14.90
	GRU	1804	708	39.25
Five-class	DNN	205 439	25 497	12.41
	CNN	205 306	7463	3.64
	LSTM	208 849	81 263	38.90
	CNN-LSTM	85 068	63 675	74.85
	GRU	208 513	28 374	13.61

Table 4 KDDCUP'99 binary classification confusion matrix.

		Predicted		
		Normal	Malicious	Total
Actual	Normal	301 031	203	301 234
	Malicious	488	188 121	188 609
	Total	301 519	188 324	489 843

Table 5 KDDCUP'99 five-class classification confusion matrix.

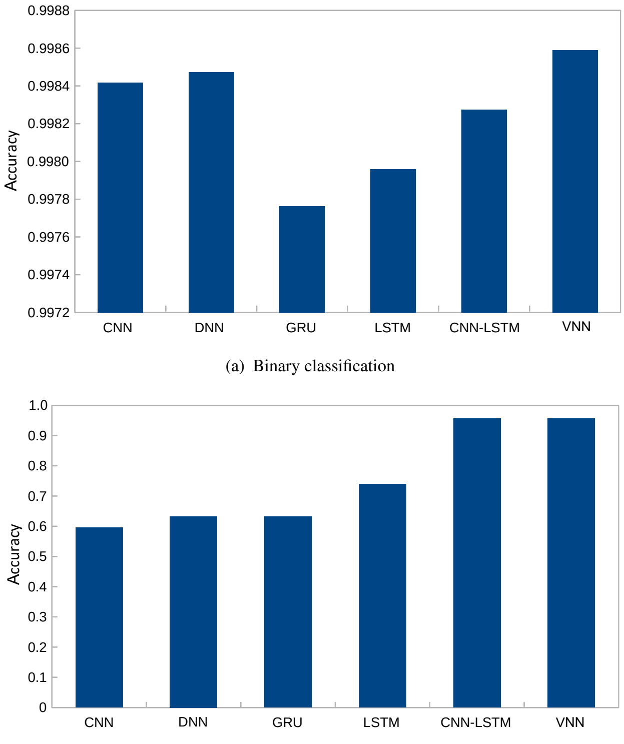
		Predicted					
		Normal	DoS	R2L	U2R	Probing	Total
Actual	Normal	277 269	219	20 608	0	0	298 096
	DoS	490	188 107	5	7	0	188 609
	R2L	0	62	3060	0	0	3122
	U2R	0	1	0	0	0	1
	Probing	0	14	1	0	0	15
	Total	277 759	188 403	23 674	7	0	489 843

4 Case Study 2: CTU-13

CTU-13 contains thirteen days labeled traffic, captured by CTU University, Czech Republic in 2011[42]. It has about twenty million netflow records, including Internet

Table 6 Measurement result of KDDCUP'99 study.

	FPR	FNR	Accuracy	Precision	Recall	F_Score
Binary classification	0.0011	0.0016	0.9986	0.9993	0.9984	0.9989
Five-class classification	0.0982	0.0021	0.9563	0.9302	0.9979	0.9628



(a) Binary classification

(b) Five-class classification

Fig. 7 VNN vs. other deep learning architectures.

Relay Chat (IRC), P2P, HTTP, fast flux, spam, click fraud, port scan, and DDoS traffic. The goal of CTU-13 is to collect a large real botnet traffic mixed with the normal user activities in the network. Table 7 describes the distribution of labels in the netflow traffic per day.

4.1 Deep learning models

Netflow traffic contains high level network activities information, including source IP/port numbers, destination IP/port numbers, protocol, Transmission Control Protocol (TCP) flags, flow duration, flow size, number of packets, input and output Simple Network Management Protocol (SNMP) interface, and next hop router. These attributes are too simple to be used in a deep learning method to detect network attacks. As a result, Haghighat et al.[35] developed a sliding window-based technique, called SmArt Window-based Anomaly detection using Netflow Traffic (SAWANT), which aggregates netflow records and extracts several meaningful attributes using sliding window algorithm.

Table 7 CTU13 label distribution.

Day	Number of flows (million)	Botnet (%)	Normal (%)	Command and control (%)	Background (%)
1	2.82	1.41	1.07	0.030	97.47
2	1.81	1.04	0.50	0.110	98.33
3	4.71	0.56	2.48	0.001	96.94
4	1.21	0.15	2.25	0.004	97.58
5	0.13	0.53	3.6	1.150	95.70
6	0.56	0.79	1.34	0.030	97.83
7	0.11	0.03	1.47	0.020	98.47
8	2.95	0.17	2.46	2.400	97.32
9	2.75	6.50	1.57	0.180	91.70
10	1.31	8.11	1.20	0.002	90.67
11	0.11	7.60	2.53	0.002	89.85
12	0.33	0.65	2.34	0.007	96.99
13	1.93	2.01	1.65	0.060	96.26

Using training small subset of netflow records (one to ten percent), SAWANT was able to achieve high accurate models, which is its main contribution. As illustrated in Fig. 8, new feature vectors were extracted from netflow traffic according to the following procedure. In addition, the label of each vector was called malicious rate, describing how many the aggregated vectors were abnormal.

(1) Slide a window of size w through the netflow records.

(2) For each position of the window, calculate these attributes:

- Number of unique values of source IP/port, destination IP/port, duration, source bytes, number of packets, and flow size per incoming and outgoing flows.
- Entropy values of source IP/port, destination IP/port, duration, source bytes, number of packets, and flow size per incoming and outgoing flows.
- Minimum, maximum, average, sum, and variance of duration, source bytes, number of packets, and flow size per incoming, outgoing, and total flows.
- Calculate malicious rate (ρ) as the label of each vector based on Eq. (9) in the following:

$$\rho = \frac{\text{Number of malicious netflow records}}{\text{Window size}} \tag{9}$$

The new feature vectors were used to train ANN model as depicted in Fig. 9, where the output layer expressed the malicious rate.

The results of the test dataset were compared with the actual malicious rate values using “Pearson correlation

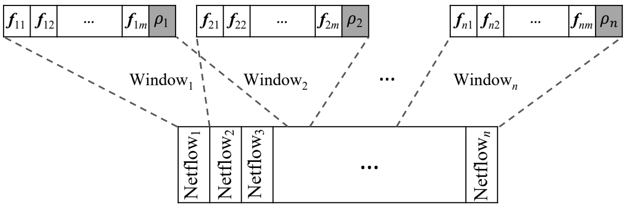


Fig. 8 SAWANT window-based feature extraction procedure.

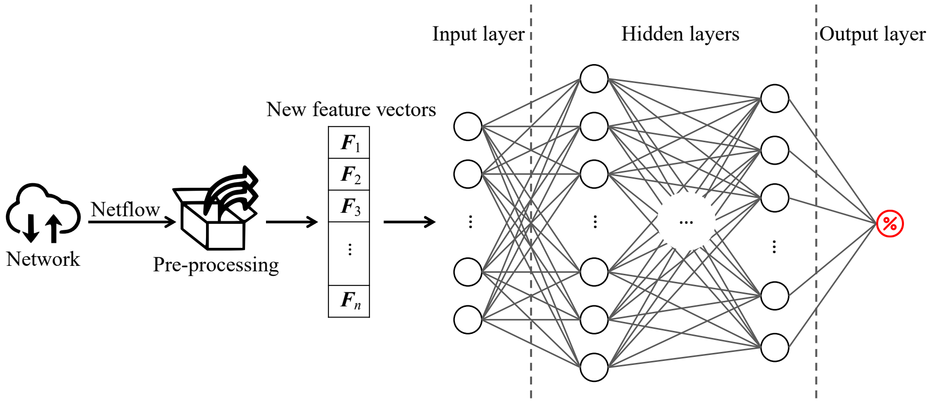


Fig. 9 SAWANT architecture.

coefficient” function, described by Eq. (10) in the following:

$$r_{X,Y}=\frac{E[XY]-E[X]E[Y]}{\sqrt{E[X^2]-E[X]^2}\sqrt{E[Y^2]-E[Y]^2}}=\frac{\sum\limits_{i=1}^{n}x_iy_i-n\bar{x}\bar{y}}{\sqrt{\sum\limits_{i=1}^{n}x_i^2-n\bar{x}^2}\sqrt{\sum\limits_{i=1}^{n}y_i^2-n\bar{y}^2}}\tag{10}$$

where X and Y were two different variable sets.

Definition 2 Let X and Y be two different data series. X and Y are positively correlated ($r=1$), and

$$\forall x_i\in X, y_i\in Y \mid y_i=\alpha x_i+\beta,$$

where α and β are two arbitrary numbers.

4.2 Voting procedure

As described in the previous section, a ranking mechanism is defined in order to select a subset of more probable models to achieve more accurate results in the voting procedure. The more decisive models were selected in the classification environment (like Case Study 1 with “malicious” and “benign” classes), the more likely it has more accuracy. However, the main challenge of SAWANT is its predicted malicious rate which is numerical (not categorical). In fact, the SAWANT predicted results were not equal to the actual values, meaning finding more decisive models impossible. Therefore, the aforementioned majority voting procedure explained in Section 3.1 is not practical here. As a result, we developed a new heuristic procedure to rank and select better models for any arbitrary test case as t.

- Normalize the accuracy of all the models according to Eq. (2) and remove less accurate models based on UMT.
- Compute the sum of Pearson correlation coefficient (r) of each predicted model with all the others.
- Sort the models based on the computed value and remove the last 50% models.
- For each two remaining predicted sets i and j:
  - Compute α as the Pearson correlation coefficient of S_i and S_j, $r(S_i,S_j)$.
  - Remove t from both S_i and S_j, and compute β as the Pearson correlation coefficient of the two sets, $r(S_i-\{t\},S_j-\{t\})$.
  - Compare the Pearson correlation coefficient calculated from the above steps.
  - Mark S_i and S_j as being similar for test case t if α is greater than β.
- Put similar models into a single set.
- Return the largest set as the voting candidate.
- Compute the result based on the majority voting schema over the parties inside the selected set.

Algorithm 3 describes the model selection procedure in detail.

4.3 Experimental results

We chose DNN, CNN, LSTM, and GRU as the deep learning structure of SAWANT and performed the voting procedure to evaluate VNN. The SAWANT pre-processed data contains 92 different attributes. We extracted 73 unique subsets, each containing 72 features. Table 8 explains the hyper parameters to test CTU-13 dataset.

We configured the deep learning structure in a way to result both higher and lower accuracies, in which the performance of DNN, CNN, GRU, and LSTM was 99%, 94%, 76%, and 70%, respectively. UMT was also configured as 0.8 to select better models in the voting procedure. Figure 10 illustrates the accuracy of each model created by various extracted subsets and deep learning architectures.

Figure 11 compares the accuracy of VNN with the utilized deep learning structures (DNN, CNN, LSTM,

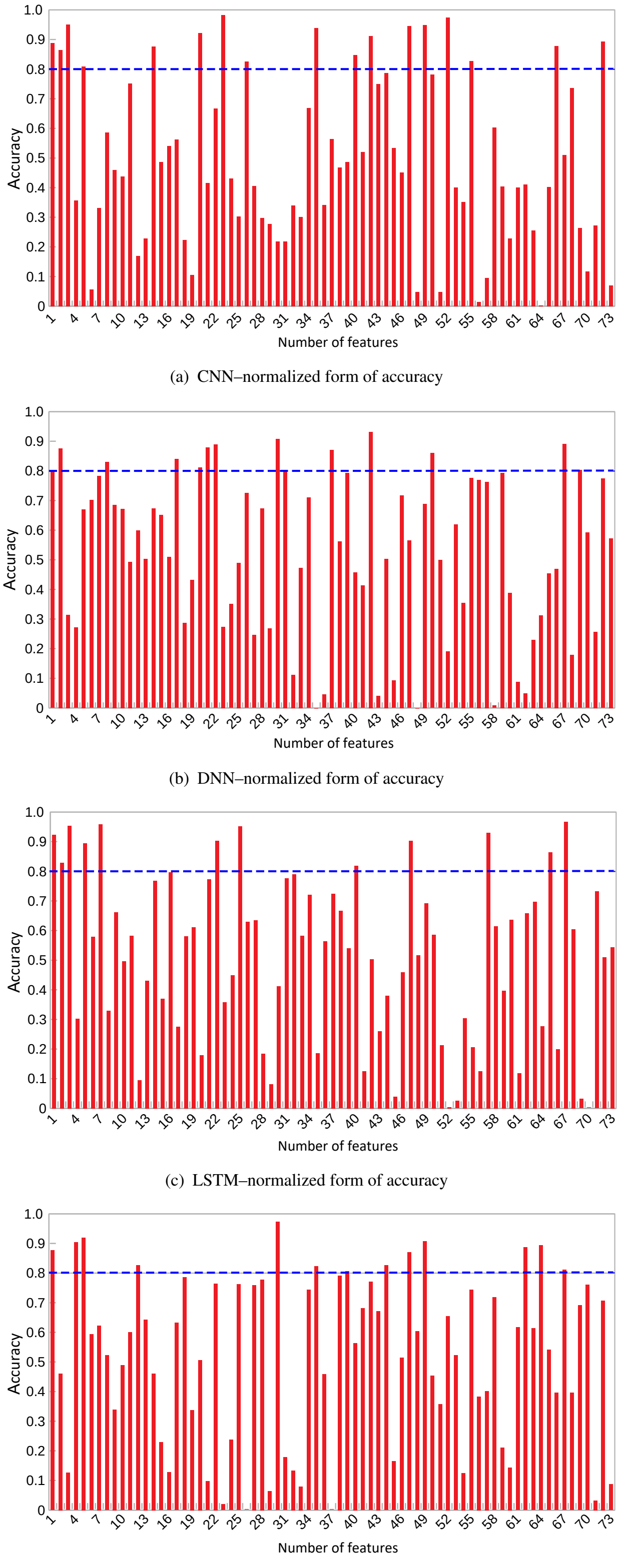
Algorithm 3 SAWANT best model selection procedure

```
input_1: $\Gamma = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$ //predicted malicous rates set
    where $\gamma_i = \{pmr_{i_1}, pmr_{i_2}, \dots, pmr_{i_m}\}$
    //predicted malicous rates of $m$ test cases
input_2: $Z = \{\zeta_1, \zeta_2, \dots, \zeta_n\}$ //accuracy of the models
input_3: $\lambda$ //unsatisfied model threshold
input_4: pivot
output: A set of $k$ best $\gamma$ of the testcase pivot
//initializing the voting parameters
$E \leftarrow 0$ //total sum of uncertainty factors
$\Delta \leftarrow \{\}$ //inittializing the output
$M \leftarrow \{\}$ //inittializing the set of satisfied models
for each $\zeta_i \in Z$
    $n_i \leftarrow$ normalize($\zeta_i$)
    if $n_i > \lambda$
        //adding corresponding uncertainty factor to $M$
        $M \leftarrow$ add($\gamma_i$)
    end
end
for each $\gamma_i, \gamma_j \in M$
    $\delta_{\gamma_i} \leftarrow \delta_{\gamma_i} + r(\gamma_i, \gamma_j)$ //$r$ is correlation coefficient
end
$\Delta_{\text{sorted}} \leftarrow$ sort($\Delta$)
$\Gamma' \leftarrow$ remain the top 50% $\Gamma$ based on $\Delta_{\text{sorted}}$
for each $\gamma_i, \gamma_j \in \Gamma'$
    $r \leftarrow r(\gamma_i, \gamma_j)$
    $r' \leftarrow r(\gamma_i - \{\text{pmr}_{i_{\text{pivot}}}\}, \gamma_j - \{\text{pmr}_{j_{\text{pivot}}}\})$
    if $r$ is greater than $r'$
        $\theta_{i,j} \leftarrow 1$
    else
        $\theta_{i,j} \leftarrow 0$
    end
end
partition $\Gamma'$ based on $\Theta$
return the largest partition as the voting candidate
```

Table 8 Hyper parameters used to test CTU-13.

Hyper parameter	Value
Train size	10%
Test size	90%
Dropout	0.2
Batch input	On
Activation function	Relu
Number of CNN layers	4
Number of LSTM layers	2
Number of DNN layers	2
Number of GRU layers	2
Number of input attributes	72
Number of input subsets	73
Output	Malicious rate
UMT	0.8
TU	0.5

and GRU). VNN decreased false alarms significantly, especially for LSTM and DNN methods, where 272 507 out of 668 597 errors (around 40%) and 12 418 out of 17 112 errors (about 72%) were corrected, respectively. Table 9 expresses the detail of error correction over CTU-13 dataset.



(a) CNN–normalized form of accuracy

(b) DNN–normalized form of accuracy

(c) LSTM–normalized form of accuracy

(d) GRU–normalized form of accuracy

Fig. 10 Model accuracy reported by the system during the training phase.

Tables 10 and 11 also summarized VNN performance over DNN as the best suited model in our case study.

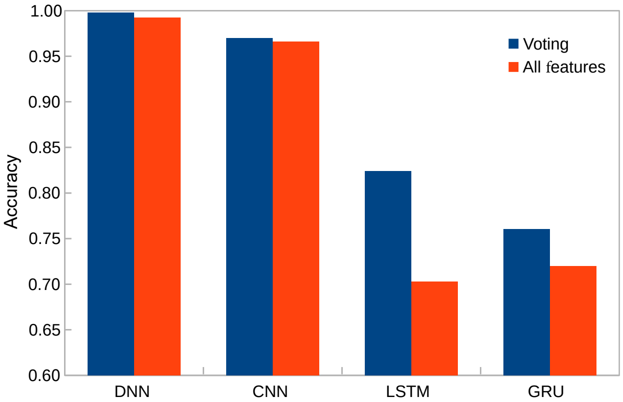


Fig. 11 System accuracy: voting-based vs. normal-based using CTU-13 dataset.

Table 9 CTU-13 error correction.

Method	Number of errors	Number of corrections	Correction rate (%)
DNN	17 112	12 418	72.57
CNN	76 523	8251	10.78
LSTM	668 597	272 507	40.74
GRU	630 541	90 902	14.42

Table 10 CTU-13 confusion matrix.

		Predicted		
		Normal	Malicious	Total
Actual	Normal	2 103 058	254	2 103 312
	Malicious	767	145 921	146 688
	Total	2 103 825	146 175	2 250 000

Table 11 Measurement result of CTU-13 study.

FPR	FNR	Accuracy	Precision	Recall	F_Score
0.0017	0.0004	0.9995	0.9999	0.9996	0.9998

5 Conclusion

This paper presents a novel voting-based deep learning framework, called VNN, to correct false alarms reported by other deep learning structures and increase the system performance. The key novelty of VNN was the ability to create several models using various kinds of deep learning structures and different aspects of data, then choosing the best models to achieve higher accuracy.

Experimental results revealed that VNN was highly effective for any kinds of deep learning structures with various hyper parameters where it corrected false labels interestingly up to 75%.

Although VNN provides high accurate prediction, creating several models is a really time-consuming procedure. In fact, 190 different models were created for each binary and 5-class classification problems over KDDCUP'99 dataset. 292 models were also generated on CTU-13. In the future, we plan to overcome this issue by developing a heuristic function, in order to ignore generating less effective models in advance. In addition, giving feedback from the candidates and utilizing the results to create more robust deep learning architecture are another direction to work in the future. Deeper analysis on different attack types (e.g., those provided in KDDCUP'99—DoS, R2L, U2R, and probing) will give us a suitable feedback to create more robust models. The proposed method missed U2R and probing attacks, however the number of samples were too small. But we plan to address this issue in the future.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61872212) and the National Key Research and Development Program of China (No. 2016YFB1000102).

References

[1] Sophos 2020 threat report, https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophoslabs-uncut-2020-threat-report.pdf, 2020.

[2] McAfee labs threats report, https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf, 2019.

[3] S. Behal, K. Kumar, and M. Sachdeva, D-FACE: An anomaly-based distributed approach for early detection of DDoS attacks and flash events, *Journal of Network and Computer Applications*, vol. 111, pp.49–63, 2018.

[4] O. Elejla, B. Belaton, M. Anbar, and A. Alnajjar, Intrusion detection systems of ICMPv6-based DDoS attacks, *Neural Computing and Applications,* vol. 30, no. 1, pp. 45–56, 2018.

[5] M. H. Haghighat and J. Li, Edmund: Entropy based attack detection and mitigation engine using netflow Data, in *Proc. of 8th International Conference on Communication and Network Security*, Chengdu, China, 2018, pp. 1–6.

[6] M. Idhammad, K. Afdel, and M. Belouch, Semi-supervised machine learning approach for DDoS detection, *Applied Intelligence,* vol. 48, no. 10, pp. 3193–3208, 2018.

[7] D. S. Terzi, R. Terzi, and S. Sagiroglu, Big data analytics for network anomaly detection from netflow data, in *Proc. of 2017 International Conference on Computer Science and Engineering*, Antalya, Turkey, 2017, pp. 592–597.

[8] J. M. Vidal, A. L. S. Orozco, and L. J. G. Villalba, Adaptive artificial immune networks for mitigating DoS flooding attacks, *Swarm and Evolutionary Computation*, vol. 38, pp. 94–108, 2018.

[9] R. Wang, Z. Jia, and L. Ju, An entropy-based distributed DDoS detection mechanism in software-defined networking, in *Proc. of 2015 IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, 2015, pp. 310–317.

[10] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, Multi-classification approaches for classifying mobile app traffic,

Journal of Network and Computer Applications, vol. 103, pp. 131–145, 2018.

[11] M. Lotfollahi, M. J. Siavoshani, R. S. Hosseinzade, and M. S. Saberian, Deep packet: A novel approach for encrypted traffic classification using deep learning, *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[12] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, MIMETIC: Mobile encrypted traffic classification using multimodal deep learning, *Computer Networks,* vol. 165, pp. 1186–1191, 2019.

[13] N. Mansouri and M. Fathi, Simple counting rule for optimal data fusion, in *Proc. of 2003 IEEE Conference on Control Applications*, Istanbul, Turkey, 2003, pp. 1186–1191.

[14] D. Ciuonzo, A. De Maio, and P. S. Rossi, A systematic framework for composite hypothesis testing of independent Bernoulli trials, *IEEE Signal Processing Letters*, vol. 22, no. 9, pp. 1249–1253, 2015.

[15] A. Khan and F. Zhang, Using recurrent neural networks (RNNs) as planners for bio-inspired robotic motion, in *Proc. of 2017 IEEE Conference on Control Technology and Applications*, Mauna Lani, HI, USA, 2017, pp. 1025–1030.

[16] J. Kim and H. Kim, Applying recurrent neural network to intrusion detection with hessian free optimization, in *Proc. of 2015 International Workshop on Information Security Applications*, Jeju Island, Korea, 2015, pp. 357–369.

[17] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, Long short term memory recurrent neural network classifier for intrusion detection, in *Proc. of 2016 International Conference on Platform Technology and Service*, Jeju South, Korea, 2016, pp. 1–5.

[18] C. Yin, Y. Zhu, J. Fei, and X. He, A deep learning approach for intrusion detection using recurrent neural networks, *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[19] S. Althubiti, W. Nick, J. Mason, X. Yuan, and A. Esterline, Applying long short-term memory recurrent neural network for intrusion detection, in *Proc. of IEEE Southeast Conference 2018*, St. Petersburg, FL, USA, 2018, pp. 1–5.

[20] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, Deep recurrent neural network for intrusion detection in SDN-based networks, in *Proc. of 2018 4th IEEE Conference on Network Softwarization and Workshops*, Montreal, Canada, 2018, pp. 202–206.

[21] Y. Yao, Y. Wei, F. Gao, and G. Yu, Anomaly intrusion detection approach using hybrid MLP/CNN neural network, in *Proc. of Sixth International Conference on Intelligent Systems Design and Applications*, Jinan, China, 2006, pp. 1095–1102.

[22] K. Wu, Z. Chen, and W. Li, A novel intrusion detection model for a massive network using convolutional neural networks, *IEEE Access*, vol. 6, pp. 50 850–50 859, 2018.

[23] M. E. Aminanto and K. Kim, Deep learning-based feature selection for intrusion detection system in transport layer, in *Proc. of Summer Conference of Korea Information Security Society*, Busan, Korea, 2016, pp. 535–538.

[24] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, A deep learning approach for network intrusion detection system, in *Proc. of 9th EAI International Conference on Bio-inspired Information and Communications Technologies*, Brussels, Belgium, 2016, pp. 21–26.

[25] F. Farahnakian and J. Heikkonen, A deep auto-encoder-based approach for intrusion detection system, in *Proc. of 2018 20th International Conference on Advanced Communication Technology*, Chuncheon-si Gangwon-do, South Korea, 2018, pp. 178–183.

[26] R. Salakhutdinov and G. Hinton, Deep boltzmann machines, in *Proc. of Twelfth International Conference on Artificial Intelligence and Statistics*, Clearwater, FL, USA, 2009, pp. 448–455.

[27] N. Gao, L. Gao, Q. Gao, and H. Wang, An intrusion detection model based on deep belief networks, in *Proc. of IEEE 2014 Second International Conference on Advanced Cloud and Big Data*, Huangshan, China, 2014, pp. 247–252.

[28] X. Zhang and J. Chen, Deep learning-based intelligent intrusion detection, in *Proc. of 2017 IEEE 9th International Conference on Communication Software and Networks*, Guangzhou, China, 2017, pp. 1133–1137.

[29] K. Alrawashdeh and C. Purdy, Toward an online anomaly intrusion detection system based on deep learning, in *Proc. of 2016 15th IEEE International Conference on Machine Learning and Applications*, Anaheim, CA, USA, 2016, pp. 195–200.

[30] R. Vinayakumar, K. P. Soman, and P. Poornachandran, A comparative analysis of deep learning approaches for network intrusion detection systems (N-IDSs): Deep learning for N-IDSs, *International Journal of Digital Crime and Forensics*, vol. 11, no. 3, pp. 65–89, 2019.

[31] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, Deep learning approach for intelligent intrusion detection system, *IEEE Access,* vol. 7, pp. 41 525–41 550, 2019.

[32] R. Vinayakumar, K. P. Soman, and P. Poornachandran, Evaluation of recurrent neural network and its variants for intrusion detection system (IDS), *International Journal of Information System Modeling and Design*, vol. 8, no. 3, pp. 43–63, 2017.

[33] R. Vinayakumar, K. P. Soman, and P. Poornachandran, Evaluating effectiveness of shallow and deep networks to intrusion detection system, in *Proc. of 2017 International Conference on Advances in Computing, Communications and Informatics*, Manipal, India, 2017, pp. 1282–1289.

[34] R. Vinayakumar, K. P. Soman and P. Poornachandran, Applying convolutional neural network for network intrusion detection, in *Proc. of 2017 International Conference on Advances in Computing, Communications and Informatics*, Manipal, India, 2017, pp. 1222–1228.

[35] M. H. Haghighat, Z. Abtahi Foroushani, and J. Li, SAWANT: Smart window-based anomaly detection using netflow traffic, in *Proc. of 2019 IEEE 19th International Conference on Communication Technology*, Xi'an, China, 2019, pp. 1396–1402.

[36] KDD CUP 1999 dataset, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[37] T. Janarthanan and S. Zargari, Feature selection in UNSW-NB15 and KDDCUP'99 datasets, in *Proc. of 2017 IEEE 26th International Symposium on Industrial Electronics*, Edinburgh, UK, 2017, pp. 1881–1886.

[38] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R.

Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, et al., Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation, in *Proc. of DARPA Information Survivability Conference and Exposition*, Hilton Head, SC, USA, 2000, pp. 12–26.

[39] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, A detailed analysis of the KDDCUP'99 dataset, in *Proc. of 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, Canada, 2009, pp. 1–6.

[40] A. zgür and H. Erdem, A review of KDD'99 dataset usage in intrusion detection and machine learning between 2010 and 2015, doi: 10.7287/PEERJ.PREPRINTS.1954.

[41] S. J. Finney and C. DiStefano, Non-normal and categorical data in structural equation modeling. *Structural Equation Modeling*: *A Second Course*, no. 10, vol. 6, pp. 269–314, 2006.

[42] CTU-13 botnet traffic dataset, https://mcfp.weebly.com/, 2011.

Mohammad Hashem Haghighat received the BS degree in computer engineering from Shiraz Azad University, Shiraz, Iran in 2008, and the MS degree in computer engineering from Sharif University of Technology, Tehran, Iran in 2010. He is currently a PhD candidate at Tsinghua University, Beijing, China. His research interests include network security, intrusion detection systems, deep learning, and information forensics.

Jun Li received the PhD degree from New Jersey Institute of Technology (NJIT) in 1997, and the MEng and BEng degrees in automation from Tsinghua University in 1998 and 1985, respectively. He is currently a professor at the Department of Automation, Tsinghua University, and his research interests include network security and network automation.