TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 629 – 643
© TÜBİTAK
doi:10.3906/elk-2104-50

Performance analysis and feature selection for network-based intrusion detection with deep learning

Serhat CANER[1], **Nesli ERDOĞMUŞ**[1,*], **Y. Murat ERTEN**[2]
[1]Computer Engineering Department, İzmir Institute of Technology, İzmir, Turkey
[2]Computer Engineering Department, İzmir University of Economics, İzmir, Turkey

Received: 07.04.2021 • **Accepted/Published Online:** 03.09.2021 • **Final Version:** 21.03.2022

Abstract: An intrusion detection system is an automated monitoring tool that analyzes network traffic and detects malicious activities by looking out either for known patterns of attacks or for an anomaly. In this study, intrusion detection and classification performances of different deep learning based systems are examined. For this purpose, 24 deep neural networks with four different architectures are trained and evaluated on CICIDS2017 dataset. Furthermore, the best performing model is utilized to inspect raw network traffic features and rank them with respect to their contributions to success rates. By selecting features with respect to their ranks, sets of varying size from 3 to 77 are assessed in terms of classification accuracy and time efficiency. The results show that recurrent neural networks with a certain level of complexity can achieve comparable success rates with state-of-the-art systems using a small feature set of size 9; while the average time required to classify a test sample is halved compared to the complete set.

Key words: Network intrusion detection, deep learning, feature selection, recurrent neural networks

1. Introduction

The unprecedented growth in the size of computer networks and their essential role in many aspects of life, from business to education, has rendered data integrity and security as one of the most crucial issues to be addressed. Networks can be attacked for the purpose of stealing information, disrupting the operations or damaging the system hardware or software. As the methods for such malicious activities get more sophisticated each year, the security measures need to be revisited and improved in order to counter balance the imposed threat.

There are a range of tools designed to timely detect a suspicious network activity and thereby to minimize or deter any potential damage. Intrusion detection systems (IDSs) are one of those network security tools which monitor and analyze network traffic either to identify patterns (fingerprints) of previously encountered intrusive activities or to detect anomalies. Those systems are most commonly grouped under:

- Host-based intrusion detection systems (HIDSs) that monitor and analyze the internals of an operating system such as system files;
- Network-based intrusion detection systems (NIDSs) that monitor and analyze the traffic flow to and from all devices on the network.

The scope of this study falls under the second group and aims to provide an insight on performances of deep learning based intrusion detectors. It does so by evaluating the accuracy and efficiency of different types of deep neural networks (DNNs) in classification of network traffic data and identification of attacks.

*Correspondence: neslierdogmus@iyte.edu.tr

Machine learning and in particular deep learning, has lately gained ground, mainly due to the increasing computational power and available data and it is proven to be a potent field. Consequently, DNNs have become popular and reliable tools in various research problems, including network security. These instruments alleviate the prerequisite to have an exhaustive field expertise for making a contribution to some extent. Hence, high accuracy can be achieved with deep learning without incorporating any expert NIDS knowledge.

Main contributions of this study can be summarized as the following: Firstly, recent related work is analyzed thoroughly. Four fundamental questions in NIDS research are identified and proposed approaches for each direction are determined. During this analysis, it has become clear that the absence of predefined experimental protocols on NIDS datasets stands as a major obstacle in the way of comparable and reproducible research. With this study, training and test partitions of 5 folds for the utilized dataset is made publicly available for research purposes and the code for experiments is made open source[1]. Secondly, four types of DNNs, each with 6 different hyperparameter settings are trained and evaluated for attack detection and classification. The experiments are conducted on the CICIDS2017 dataset [1] that contains 15 kinds of attacks such as DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port scan and Botnet. Finally, using the best performing DNN, the raw network traffic features in the dataset are analyzed with respect to their impact on the classification accuracy. All findings are compared to the results reported in the recent literature.

The rest of the paper is organized as follows: Section 2 presents recent studies on deep learning based NIDS. It discusses various approaches with regard to the four fundamental questions. Section 3 describes the methodology of the proposed approach. It also details the utilized deep learning models and the training process. Experimental results are provided and discussed in Section 4. The paper is concluded in Section 5.

2. Related work

Training both shallow and deep artificial neural networks (ANNs) have been studied for many years. The first training methodology for deep ANNs with many layers was published in 1967 [2], however, the processing power at that time was not sufficient to successfully tackle complicated problems. It was not until 2010s that deep learning excelled at a multitude of tasks and became a pervasive tool for predictive problems. This mainly happened due to solutions for functional issues such as the vanishing gradient problem [3, 4] or overfitting [5] and the availability of sufficiently large training data and computing power.

Not surprisingly, deep learning based solutions have also infiltrated the NIDS research. In recent years, many publications have appeared that propose the use of DNNs for intrusion detection. The reader may refer to numerous survey studies conducted in the field. We would like to mention six recent surveys that mainly review deep learning methods for NIDSs [6–11]. In those papers, [6] and [7] list available datasets and proposed methods for NIDS but no performance comparison is given. In [8], seven deep learning models are implemented and tested on CICIDS2018 and the Bot-IoT [12] datasets and in [9], four deep learning models are trained and evaluated on two legacy (KDD 99, NSL-KDD [13]) and two more recent (CICIDS2017 and 2018) datasets.

During our literature review, we observe that the task of machine learning based intrusion detection is approached differently with respect to how the following four questions are answered:

1. What type of classifier is used?
2. What is the input of the classifier?
3. What is the output of the classifier?
4. How is the class imbalance handled for training?

[1]https://github.com/neslierdogmus/caner_ids_2020

The answers to those questions in the literature and how they are handled in this study are given in the following subsections.

2.1. Classification

One of the main components in machine learning based NIDSs is a classifier that predicts the labels of the raw, extracted or selected features. The classification methods used for this task in the relevant literature can be divided into two parts: deep learning based approaches and classical machine learning approaches.

Among studies that utilize deep architectures for classification, a multilayer perceptron (MLP) with 8 hidden layers with 140, 120, 100, 80, 60, 40, 20, and 120 nodes and another with 6 hidden layers with 1024 nodes are evaluated in [14] and [15], respectively. Additionally, in [16] convolutional neural networks (CNNs), in [17, 18] LSTMs and in [17] deep belief networks (DBNs) are proposed as NIDS classifiers. Among others, numerous machine learning approaches have been proposed such as support vector machines (SVM) [19, 20], decision trees [21], Bayesian networks [22] and random forests [21–25].

In this study, four DNN architectures, MLP, LSTM, gated recurrent unit (GRU), CNN, each with six different complexity levels are trained and tested using CICIDS2017 dataset.

2.2. Classifier input

Raw network traffic data may be noisy and and include redundant components. Because of this, attack detection and classification performances and efficiencies may degrade. In order to both reduce the input dimensionality and to increase its discriminative power, numerous feature extraction and selection techniques have been proposed.

For feature extraction, one of the most popular approaches recently is to use autoencoders. An autoencoder (AE) is an artificial neural network that can learn how to efficiently represent data in an unsupervised manner. It compresses the input to obtain a lower-dimensional representation and then computes the output from that representation with the goal of reconstructing the original input. In [23], the best performance is reportedly obtained when original feature size of 77 in the CICIDS2017 dataset is reduced down to 40. Similarly in [20] and [22], latent representations are computed using an autoencoder and the dimensionality of data is reduced down to 20 and 59, respectively. In [26], long short-term memory (LSTM) network is utilized for feature extraction.

In order to select features, Vijayanand et al. [19] utilize whale optimization algorithm and genetic operators to determine the 35 most informative features in the CICIDS2017 dataset. After manually removing 18 irrelevant and noisy features first, Khammasi and Krichen use nondominated sorting genetic algorithm II and select 33 features from the remaining set. In [25], correlation based feature selection with bat algorithm is applied and the best 13 features are selected. Elmasry et al. propose a particle swarm optimization based approach to select an optimized feature subset of size 23. Finally in [27], the most useful features are first selected by random forest algorithm and then they are grouped according to the degree of similarity using affinity propagation algorithm.

In our study, feature selection is realized by analyzing the importance of each individual raw feature by performing an ablation test and ranking the features with respect to the performance loss in their absences.

2.3. Classifier output

The intrusion detection task essentially poses a binary classification problem in which the network activities are classified as either benign or malicious. In reality, malicious activities are greatly outnumbered by the benign ones. Hence, this task can also be approached as an anomaly detection issue. However, in most NIDS datasets, attacks are also divided into several subclasses and this transforms the case at hand into a multiclass classification task.

In this regard, the literature fails to agree on a common problem definition. Number of classes in which the network flow data are classified into varies greatly. In the CICIDS2017 dataset, there are 15 classes including the benign class and among the recent machine-learning based solutions evaluated on this dataset, the number of classes has at least eight different values. In [16, 21, 22, 28], all 15 classes are used as the classifier outputs. In [14, 23, 24], three web attack types (brute force, SQL injection and cross site scripting) are grouped as one, resulting in a total of 13 classes. Furthermore in [15, 17], four denial of service (DoS) attack classes (GoldenEye, Hulk, Slow-httptest and slowloris) are merged into a single DoS category and two attack classes with very few samples (heartbleed and infiltration) are ignored which brings the number of classes down to 8. As expected, in some studies binary classification is performed by grouping all different types of attacks into a single attack category [20, 27].

In our experiments, all classifiers are trained and tested with 15 classes. For comparison purposes, binary and 13-class classification results are also calculated from the same confusion matrices by collecting some attack types in a single class as mentioned above. Certainly, training a separate network for each different output size would be optimal and this comparison is not fair. However, our focus in this study is the evaluation of numerous DNN architectures and retraining them for different output sizes is excluded.

2.4. Dealing with class imbalance

It is reasonable to think of an attempt for network intrusion as a rare event. [29] reveals that there is a cyber-attack every 39 seconds, but considering that thousands of gigabytes of Internet traffic occurs every second, it does not change the fact that attacks seldom happen compared to benign activities. That imbalance is also reflected in many intrusion detection datasets available for research, including CICIDS2017.

It is normally a desired property for a sample to mimic the population distribution. However, such severe class imbalance is problematic in machine learning since it leads to poor predictive performance for the outnumbered classes. Moreover, it is more crucial to correctly classify those classes because they belong to an abnormality to be detected. This problem is handled with various approaches in the NIDS literature. In some studies, the minority classes are oversampled, in some others the majority classes are undersampled and in some both over and undersampling are applied. For instance in [18], random downsampling is performed for benign and port-scan classes in CICIDS2017. For oversampling the minority classes, the most common method is the synthetic minority oversampling technique (SMOTE) [30] as utilized in [14, 16]. The key idea in SMOTE is to create synthetic examples of the minority class rather than to oversample them with replacement. This is achieved by computing synthetic examples along the line segments that join minority class nearest neighbors. With the rise of generative adversarial networks (GANs), creating synthetic samples of minority classes using different GAN models have also been proposed [23, 24, 31].

In this paper, in order to observe the impact of class imbalance, the DNNs under analysis are trained twice: Once with a training partition that has the actual dataset distribution (referred as nonuniform) and once with a training partition that has uniform distribution among benign and attack classes (referred as uniform). The second training partition is obtained by undersampling only the benign class; the discrepancy among individual attack class counts is disregarded.

3. Methodology

ANNs are computational models that consist of layers of interconnected artificial neurons with activation functions that decide what a neuron's output will be based on its inputs. A DNN is an ANN with multiple layers between the input and output layers. Many DNN topologies exist with varying working principles but essentially, we can distinguish two types of DNNs based on whether they process feedback or not: Feed-forward networks, in which data flows only in the forward direction and there are no feedback; and recurrent (feedback) networks that allow data signals travel in both directions using loops. In order to compare the performances of these two different types of DNNs in multiclass classification for NIDS, two types of feed-forward (MLP and CNN) and two types of recurrent (LSTM and GRU) DNNs are trained. MLP, CNN and LSTM are selected due to their frequent usage in NIDS research. Additionally, for recurrent DNNs, GRU is included in the experiments to see if they would have any advantage over LSTMs. GRUs have fewer parameters and thus may train a bit faster or need less data to generalize.

MLP networks consist of input, output and a number of hidden layers [32]. Input and output layers' dimensions are determined by the feature size and number of classes, respectively. On the other hand, the number of hidden layers and the size of each hidden layer are hyperparameters related to the network structure. CNNs are also composed of multiple layers of artificial neurons but it also has convolutional layers that can perform feature extraction.

In RNNs, an abstract concept of sequential memory is implemented by providing the output of a hidden unit back to itself. However, this memory is short-term, so the network struggles to remember the information for long data sequences. Moreover, the problem of vanishing gradients arises. LSTMs [33] and GRUs [34] are introduced to combat these shortcomings. They both have internal mechanisms called gates that regulate the information flow.

In this research, six MLPs, six LSTMs, six GRUs and six CNNs, each with a different number and size of hidden layers (Table 1) are trained and tested. All DNNs has a single dense layer at the end with softmax as its activation function for classification. The hyperparameter sets given in Table 1 are constructed so that each DNN architecture can be tested with varying depth and breadth. For CNNs, the hyperparameter selection is based on the architecture proposed in [16] which is replicated as CNN-4.

Table 1. Hyperparameter values and number of parameters for DNN models.

Model	Number of neurons per hidden layer	Number of parameters	Model	Number of neurons per hidden layer	Number of parameters
MLP-1	200-200-10	57975	LSTM-1	30	13545
MLP-2	400-400-20	199935	LSTM-2	30-30	20985
MLP-3	1200-600-20	826535	LSTM-3	30-30-30	28425
MLP-4	200-200-200-20	100335	LSTM-4	60	34275
MLP-5	200-400-400-20	264735	LSTM-5	60-60	63555
MLP-6	400-800-200-20	516535	LSTM-6	60-60-60	92835
GRU-1	30	10275	CNN-1	32-32	11631
GRU-2	30-30	15855	CNN-2	64-64	30415
GRU-3	30-30-30	21435	CNN-3	32-32-32	16367
GRU-4	60	25935	CNN-4	32-64-32	27759
GRU-5	60-60	47895	CNN-5	32-64-64	29295
GRU-6	60-60-60	69855	CNN-6	64-64-64	49103

The best performing DNN model is determined according to the classification performances and used to analyze the contribution of individual network flow data features to the accuracy. This analysis is conducted by training and evaluating the network as many times as the total feature length by excluding one of the features at each iteration.

Next, the amount of the deterioration with respect to the original classification accuracy is used to assess the effectiveness of the features and they are sorted accordingly. The network is again trained multiple times, this time by adding one feature to the feature set at each iteration, starting from a set of top three features. With this study, the smallest feature set size that achieves an accuracy comparable to the complete set of features is determined.

Finally, the result of the feature analysis is tested by randomly sampling 100 feature sets of the same size and again training the network and evaluating its performance. The probability distribution of the accuracy variable is predicted using kernel density estimation (KDE) and the probability that the obtained findings are meaningful and repeatable is computed by hypothesis testing.

4. Experimental results and discussion

In this section, firstly, a detailed explanation is given for the utilized dataset and the preprocessing steps. Next, the evaluation metrics used in this study and in other NIDS related studies are introduced. Finally, the results for the conducted experiments are given in two parts: performance comparison for the aforementioned DNNs and feature analysis with the best performing DNN.

4.1. Dataset and preprocessing

CICIDS2017 dataset [1] is provided by Canadian Institute of Cybersecurity and it contains traffic data for benign and 14 types of attacks, collected over a span of 5 days. In total, there are 3119345 network flows labelled as one of these classes, each with 83 features obtained by a network traffic analysis tool (CICFlowMeter). Among those features, flow id, source IP, source port, destination IP, destination port and timestamp are used for manual labelling of the flow and they do not contain any information about the content. Hence, these 6 data columns are excluded and the experiments are carried out considering the remaining 77 features.

Before the experiments, the features in the training set are transformed to the range of (0,1) by min-max normalization. The obtained transformation parameters (maximum and minimum values of each feature) are used to apply the same scaling on the features of the test samples.

As given in [1], the number of class instances is quite imbalanced. Additionally, 288804 of the flow samples are either do not have any data or are a duplicate of another sample. After those are removed, the defects in the remaining 2830541 flow data are fixed by replacing ”Not a number” (Nan) entries with zeros and ”infinity” (inf) entries with the dataset average of the corresponding features.

A subset of size 1715249 is determined as the excess benign samples by including all Monday data and selecting the rest randomly. When this set is excluded, the distribution of the benign and attack samples in the dataset becomes uniform. In the experiments, this set of excess benign samples is used for two purposes:

1. To compare the network performances when trained with uniform and nonuniform training sets. To this end, the DNNs are trained with and without extra benign samples in the training partitions for all folds. The test partitions are uniform.
2. To compare the network performances when tested with uniform and nonuniform test sets. A nonuniform test set is more realistic since in real life, network intrusion attacks rarely happen with respect to benign

traffic size. However, such test set composition may cloud the performance assessment since attack samples are outnumbered and hence performance metrics are biased towards the majority class. In order to evaluate the performances in both settings, extra benign samples are added to the test partitions for all folds. The training partitions are uniform.

For the experiments, 5 folds are created with nonoverlapping training and testing partitions in which each sample is tested exactly once. The number of samples for each class, fold and partition after preprocessing and excluding the excess benign samples are given in Table 2.

Table 2. Sample sizes per class, per fold and per partition after preprocessing and excluding the excess benign samples (WA: web attack).

	Fold1 train	Fold1 test	Fold2 train	Fold2 test	Fold3 train	Fold3 test	Fold4 train	Fold4 test	Fold5 train	Fold5 test
BENIGN	446296	111350	445980	111666	446079	111567	445931	111715	446298	111348
ATTACK	445937	111709	446254	111392	446155	111491	446303	111343	445935	111711
Bot	1586	380	1553	413	1568	398	1580	386	1577	389
DdoS	102304	25723	102529	25498	102515	25512	102298	25729	102462	25565
DoS GoldenEye	8253	2040	8245	2048	8215	2078	8227	2066	8232	2061
DoS Hulk	184738	46335	185011	46062	184873	46200	185050	46023	184620	46453
DoS Slowhttptest	4392	1107	4375	1124	4409	1090	4454	1045	4366	1133
DoS slowloris	4645	1151	4635	1161	4612	1184	4682	1114	4610	1186
FTP-Patator	6414	1524	6321	1617	6330	1608	6373	1565	6314	1624
Heartbleed	7	4	10	1	8	3	9	2	10	1
Infiltration	31	5	32	4	33	3	29	7	19	17
PortScan	127151	31779	127073	31857	127145	31785	127147	31783	127204	31726
SSH-Patator	4659	1238	4738	1159	4732	1165	4705	1192	4754	1143
WA-Brute Force	1202	305	1223	284	1170	337	1217	290	1216	291
WA-Sql Injection	18	3	14	7	16	5	19	2	17	4
WA-XSS	537	115	495	157	529	123	513	139	534	118

4.2. Evaluation metrics

In the experiments, 15-class classification accuracy (ACC-15) is considered as well as the binary classification accuracy (ACC-2), precision (PRE) and recall (REC). For comparison reasons, 13-class classification accuracy (ACC-13) is also computed by merging all web attacks in a single class. No additional training is done for 2-class and 13-class experiments. Instead, the predictions of 15-class classifiers are manipulated by label grouping.

For all different class configurations, accuracy values are calculated by the ratio of number of correctly classified test samples to the whole test set size. For binary classification, precision (1-false alarm rate) is calculated as the ratio of number of correctly classified attack test samples to the number of all samples predicted as attacks. Recall (also referred as true positive rate, sensitivity or detection rate in the literature) is calculated as the ratio of correctly classified attack test samples to all attack test samples. Again for comparability, F1-scores (harmonic mean of the precision and recall) is also reported.

With imbalanced datasets like CICIDS2017, overall accuracy values may obscure class specific performances. In order to give more detail in that respect, per class performances are also calculated by the class specific detection rates (number of correctly classified test samples in a class divided by the total number of samples of that class) and a final metric is obtained by taking their average [23, 24]. Similar evaluation approaches are also mentioned and used in [9, 22].

For feature analysis and selection experiments, inference efficiency is also evaluated, in addition to the classification performances. This is measured as the average time taken to predict a class for a single sample. All experiments are conducted on a computer with the following features: 12-core 3.8 GHz CPU, 32 GB RAM, 11GB V-RAM and a Linux operating system.

4.3. Performance comparison for different DNN models

24 DNN models are trained with an input layer of size 77 (the entire feature length) and an output layer of size 15. The number of parameters to be trained are given in Table 1. Since GRU has less gates in its cells, number of parameters are less than LSTM even though the numbers of cells are the same. For the MLP models, the leaky rectified linear unit (LReLU) activation [35] is used with negative slope coefficient of 0.01. LReLUs avoid the problem of dying ReLUs which become inactive and always output 0.

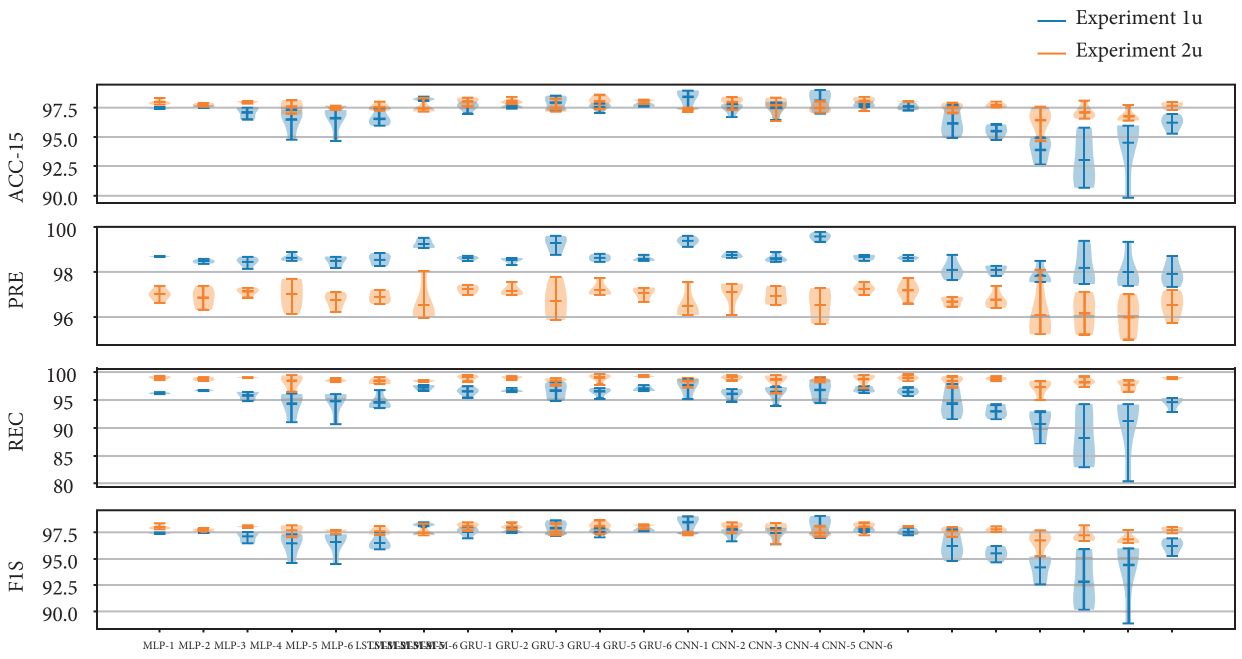
All DNNs are trained using Adam optimization [36] and the cross-entropy loss between the labels and predictions. Number of epochs is limited to 100 but early stopping is employed to halt the training when the validation loss stops decreasing. Finally, the models are tested by predicting test sample labels and computing the evaluation metrics.

In the first set of experiments, the models are trained with a nonuniform training set which includes the excess benign samples. Since the test and training partitions should not overlap, the testing is only done with the uniform test set with similar number of benign and attack samples (1u). In the second set of experiments, the excess benign data are not used in training, so the training set is uniformly distributed with respect to benign and attack samples. Thereby, the trained models can be tested on both the uniform test set used in the first set of experiments (2u) and the nonuniform test set which additionally includes the excess benign samples (2nu). Nonuniform test sample distribution can be argued to be more similar to the real-world data.

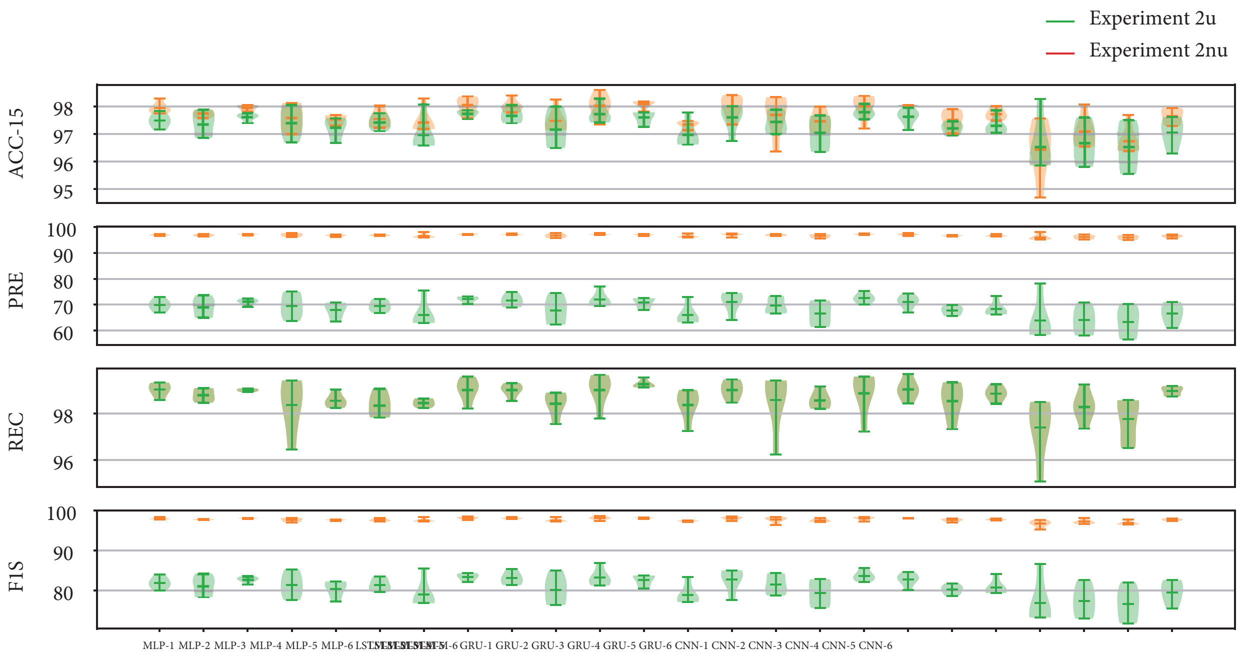
Evaluation metrics for experiments 1u and 2u are given in Figure 1a. When the accuracies and F1 scores are compared, it is observed that in general, the performances have improved even when the training set is made only partially balanced. This indicates that better accuracy values can be obtained with more advanced techniques to balance also the samples of different types of attacks, as claimed in the existing studies [14, 16]. The precision and recall plots show that networks trained with uniform sets are more inclined to predict a sample as an attack than the ones trained with nonuniform training sets. On the other hand, when excess benign samples are included in training, attack predictions become more costly, which leads to higher precision but lower recall rates.

Results for experiments 2u and 2nu are given in Figure 1b. In these experiments, the models trained with uniform training data are tested on both uniform and nonuniform test partitions. More specifically, excess benign samples are included for testing, so that the test data distribution is more realistic. Since the attack data are the same in both test set compositions, recall rates do not change. On the other hand, precision rates drop significantly. This is mainly due to the fact that a large number of benign samples are added for testing and hence, the number of correct attack predictions is overshadowed by the number of incorrect attack predictions. Even though experiment 2nu does not clearly reflect classifier performances, it demonstrates performances on more realistically distributed data.

For all training and testing configurations, mean accuracy values across 5 folds are given in Table 3. Since uniform training generally gives better results and uniform testing avoids the dominance of benign test samples



(a)



(b)

Figure 1. Performances of 24 DNN models (a) when trained with uniform training sets and tested on uniform (2u) and nonuniform (2nu) test sets and (b) when trained with nonuniform training sets that include excess benign samples and tested on uniform test sets (1u) and when trained with uniform training sets and tested on uniform (2u) test sets. Violin plots are used to visualise the entire range and distribution of the evaluation metrics. Horizontal lines in the middle show the means.

on the accuracy and precision metrics, Experiment 2u is chosen to compare different DNN models and decide on the best performing model that is to be used in the subsequent analyses. On that account, the best accuracy is found to be achieved by LSTM-6 model with 98.08%. Furthermore, LSTM-6 attains one of the lowest standard deviations (0.2%) among top performing models which indicates a better generalization capability.

Table 3. Mean accuracy values across 5 folds for three experimental setup and 6 DNN models. Best performing models for each type of experiments are marked as bold.

	Exp.	1	2	3	4	5	6
MLP	1u	97.44	97.58	97.11	96.48	96.61	96.55
	2u	97.94	97.74	97.95	97.58	97.53	97.54
	2nu	97.48	97.34	97.61	97.38	97.22	97.41
LSTM	1u	98.23	97.57	97.60	97.90	97.54	97.78
	2u	97.42	98.04	98.02	97.47	98.02	**98.08**
	2nu	96.96	97.73	97.68	97.16	97.71	97.60
GRU	1u	**98.42**	97.42	97.48	98.09	97.77	97.58
	2u	97.34	97.99	97.69	97.45	98.00	98.02
	2nu	96.96	97.60	97.43	97.03	**97.77**	97.61
CNN	1u	96.19	95.52	93.94	93.01	94.53	96.21
	2u	97.51	97.72	96.43	97.10	96.74	97.64
	2nu	97.20	97.29	96.54	96.66	96.54	97.05

The most successful models among MLP, LSTM, GRU and CNN networks are compared with six recent publications in which evaluation metrics are reported on the whole CICIDS2017 dataset for 15, 13 and 2 class classification. At the top part of Table 4, the overall success rates are presented as the average of metrics obtained from 5 folds. The results suggest that comparable results are obtained by all four DNN types, with LSTM-6 slightly exceeding the others. At the bottom part, the asterisks indicate that these metrics are obtained separately for each class and their averages are calculated. The same evaluation approach as in [23, 24] is adopted for 13-class classification. For such an imbalanced dataset, per class evaluation gives a clearer insight on how classifiers perform individually for each attack type. To the contrary of overall evaluation, with this method, all classes in the test set contribute equally to the average success rate irrespective of their sample size.

In [22], the results are obtained by reducing the dimensionality of the features from 81 to 59 using AE and classifying using random forest. The authors use extra 4 features from the dataset: source IP, source port, destination IP, destination port which are largely omitted in the literature and in this study. The performances reported by [16] are achieved by the implemented CNN-4 architecture but in addition, the class imbalance is handled using SMOTE for oversampling and Gaussian mixture model based undersampling. This comparison reveals that even though balancing benign and attack samples improve overall performances, incorporating more sophisticated class imbalance solutions into the system is crucial.

The results with class averages point towards the same direction. They show that all three DNN models perform poorly with small number of samples such as heartbleed and infiltration. Oversampling for rare classes is observed to be essential in order for them to be properly classified. In [23, 24], GANs are utilized to generate new synthetic data similar to the existing rare samples. [14] also handles the data imbalance with SMOTE for oversampling and edited nearest neighbor method for undersampling.

Table 4. Comparison of the most successful models among MLP, LSTM, GRU and CNN networks with recent related studies. In the subtable given below, each metric is obtained separately for each class and the average over all classes is reported.

	ACC-15	ACC-13	PRE	REC	F1
[16]	99.85	-	99.85	99.88	99.86
[21]	95.16	-	99.65	85.32	91.93
[22]	99.50	-	99.80	98.50	99.60
MLP-3	97.95	97.96	97.11	99.00	98.05
LSTM-6	98.08	98.09	97.06	99.25	98.15
GRU-6	98.02	98.03	97.17	99.03	98.09
CNN-2	97.72	97.72	96.74	98.85	97.78
[14]	-	99.95	94.31	95.62	94.10
[23]	-	99.86	98.46	93.29	95.38
[24]	-	99.83	98.68	92.76	95.04
MLP-3*	-	97.96	94.23	71.90	72.55
LSTM-6*	-	98.09	92.81	78.56	74.70
GRU-6*	-	98.03	89.72	81.22	75.03
CNN-2*	-	97.72	94.40	72.52	70.52

4.4. Feature analysis and selection

In the second group of experiments, the available features in the dataset are evaluated with respect to their contributions to the performance of LSTM-6 network. To this end, firstly an ablation test is conducted for which the features are excluded one by one and the performance of the network trained with the remaining feature set is assessed. It is observed that while exclusion of some features decrease the performance, for some others the accuracy improves. The results are given in the first bar plot in Figure 2.

In the next step, the features are sorted in increasing order with respect to the network performances trained in their absence. Starting from the top 3 features (fwd-iat-std, init-win-bytes-forward and fin-flag-count), networks are trained by cumulatively including the next feature in the queue, finally reaching the point where the whole feature set is used. In total 75 networks are trained with increasing feature size from 3 to 77 for which the mean ACC-15 metrics are presented in the second bar plot in Figure 2. The baseline rates with 3 features are 88.52%, 88.87%, 93.27%, 90.99% for ACC-15, precision, recall and F1 score, respectively. After including the top 9 features the accuracy values stabilize and do not improve significantly. The evaluation metrics achieved with those 9 features are 97.58% for accuracy, 96.58% for precision, 98.86% for recall and 97.71% for F1 score. When compared to the LSTM-6 performance using the whole feature set, the loss in the performances are found to be very small. On the other hand, according to the average single sample inference times given in last plot in Figure 2, the amount of time required for predictions decreases substantially. It takes the network 100.17 µs with 77 features to make a prediction whereas this drops down to 40.56 µs with 9 features. The time cost to predict a class for a single sample performance for LSTM-6 is compared with other similar studies in Table 5. It shows that a better time efficiency is achieved with feature selection without a considerable loss in classification success rates. Compared to [16], classification can be done twice as fast with only 2.27% decrease in 15-class classification accuracy.

Lastly, in order to assess our hypothesis that some features hold more useful information for intrusion detection, the null hypothesis is tested. Since no significant increase is observed for more than 9 features, 9

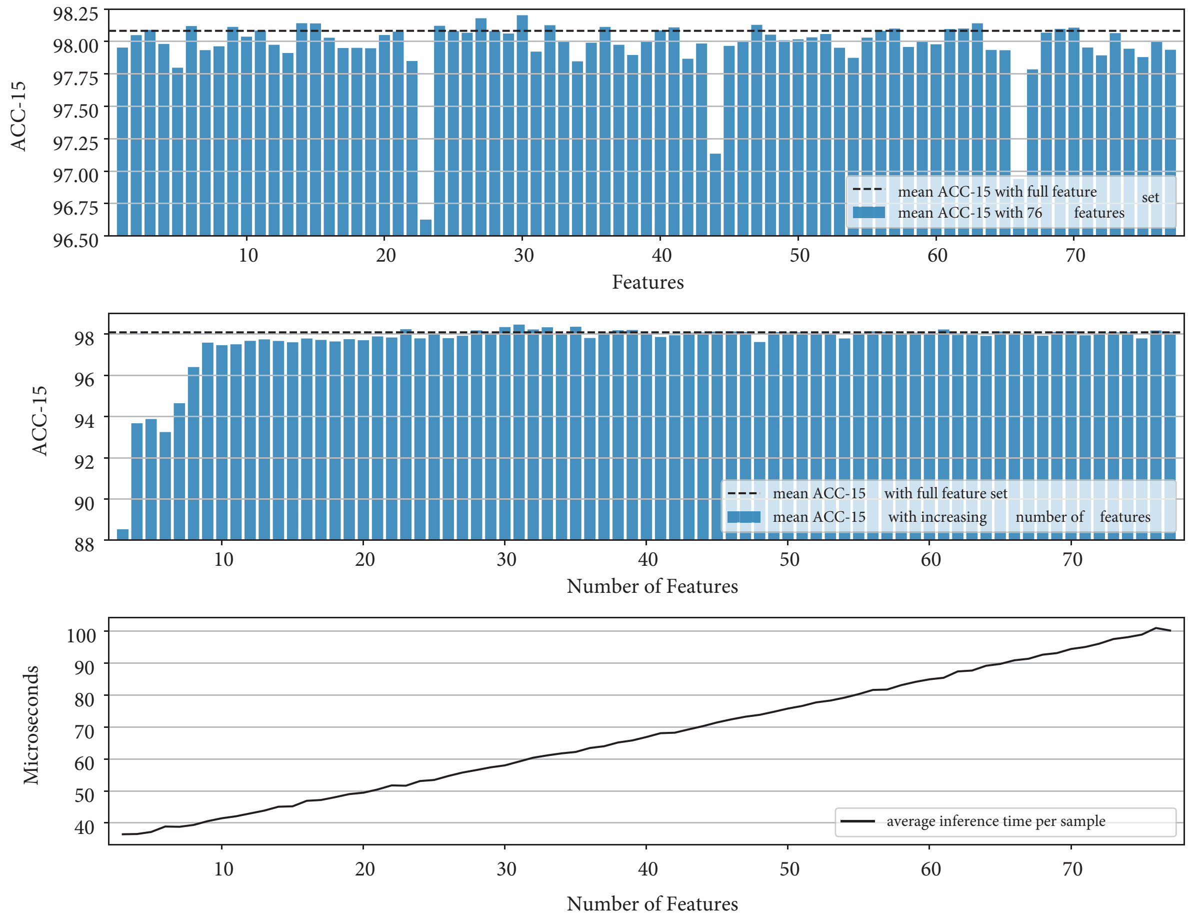


Figure 2. Classification and time performances in feature analysis experiments. Top: Accuracy values obtained in the ablation tests that are conducted by excluding one feature at a time. Middle: Accuracy values obtained by cumulatively adding one feature at a time in the order of increasing ablation performances starting from 3 best performing features. Bottom: For the same experiment, average single sample inference times obtained for increasing feature set size.

is selected as the number of features to be used in the experiments. In total 100 tests are executed each with randomly selected 9 features. Obtained accuracies are given in Figure 3 as a histogram. With 9 features selected as the result of the ablation study, 97.58% accuracy is obtained. None of the randomly selected feature sets could exceed this success rate. Additionally, probability density is estimated using kernel density estimation. The estimated probability density above 1.00 is truncated and estimations are rescaled to normalize the area under the curve to 1. According to the estimated density, probability of obtaining an accuracy higher than 97.58% with randomly selected 9 features, in other words p-value, is found to be 0.0289. This value is less than a significance level of 0.05, so we can safely reject our null hypothesis and accept that the our ablation analysis provides us with a more discriminative feature set.

5. Conclusion and future work

Intrusion detection systems are evolving and will likely continue to evolve as network attack methods change and new computational capabilities are achieved. Particularly, deep learning is shown to have high potential for malicious activity detection and identification in recent studies.

Table 5. Per sample test time comparison for LSTM-6 with other similar systems proposed recently.

	CPU (GHz,MB)	RAM (GB)	V-RAM (GB)	Time (µs)
[16]	3.6, 16	64	11	79.89
[17]	3.8, 16	16	5	6373.33
[18]	2.1, 20	32	11	220.71
LSTM-6	3.8, 64	32	11	**40.56**

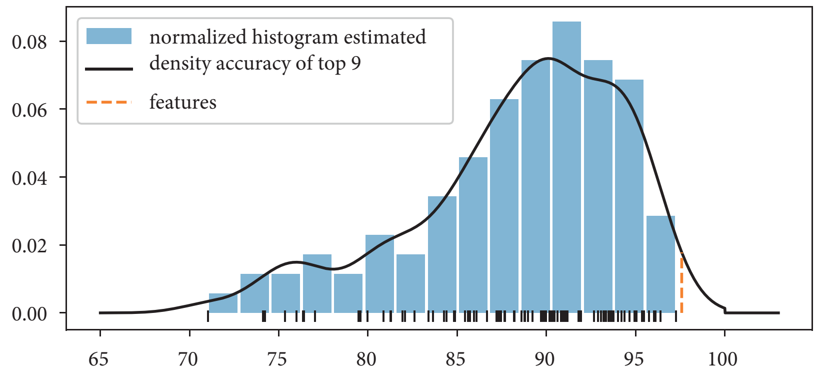


Figure 3. Histogram of 100 accuracy values obtained by using 9 randomly selected features and the probability density function estimated by kernel density estimation. 97.58% accuracy achieved by top 9 features in the ablation tests is marked with dashed line. The probability of randomly obtaining this accuracy and higher is 0.0289.

In this study, firstly, the most recent related work is surveyed in order to have a comprehensive idea of the current state of deep learning based NIDS research. Our analysis reveal that most of the publications suffer from not being repeatable and comparable. This is mostly due to the fact that datasets used do not impose a certain experimentation protocol and researchers have conducted and reported classification experiments with different training/testing sample sizes, different number of classes and different evaluation metrics. The training and testing partitions used for this paper are made publicly available for future use, together with the trained models and the code for data preprocessing, training and testing.

Moreover, four key research directions are specified: classifier types, feature selection/extraction, output labels and class balancing. The existing studies are analyzed accordingly and the stance of this study for each of those directions is explained. 6 MLPs, 6 LSTMs, 6 GRUs and 6 CNNs with different hyperparameter sets are trained and evaluated as classifiers. The impact of imbalanced class size is observed by experimenting with balanced and unbalanced sets.

Using the most successful model, the effectiveness of the features are assessed by an ablation study and used for feature selection. By using only 9 of the raw features 97.58% accuracy is achieved while reducing the test time per sample down to 40.56 µs. Our hypothesis that those 9 features are more effective for intrusion detection and classification is also tested by evaluating 100 randomly selected feature sets of size 9 and the null hypothesis is rejected since the p-value is found to be 0.0289.

In the future, this work can be extended by including the CICIDS2018 dataset and testing the generalizability of our findings. Additionally, the DNN performances can be determined after the class imbalance is fixed with more sophisticated methods. Finally, separate classifiers trained specifically for different numbers of classes can be evaluated.

References

[1] Sharafaldin I, Lashkarin AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security and Privacy; Funchal, Madeira, Portugal; 2018. pp. 108-116.

[2] Ivakhnenko AG, Lapa VG. Cybernetics and Forecasting Techniques: Modern Analytic and Computational Methods in Science and Mathematics. New York, USA: American Elsevier Publishing Company, 1967.

[3] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In: International Conference on Artificial Intelligence and Statistics; Fort Lauderdale, Florida, USA; 2011. pp. 315-323.

[4] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition; Las Vegas, Nevada, USA; 2016. pp. 770-778.

[5] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 2014; 15 (1): 1929-1958.

[6] Gümüşbaş D, Yıldırım T, Genovese A, Scotti F. A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems. IEEE Systems Journal 2020; 15 (2): 1717-1731. doi: 10.1109/JSYST.2020.2992966

[7] Asharf J, Moustafa N, Khurshid H, Debie E, Haider W, et al. A review of intrusion detection systems using machine and deep learning in Internet of Things: Challenges, solutions and future directions. MDPI Electronics 2020; 9 (7): 1177. doi: 10.3390/electronics9071177

[8] Ferrag MA, Maglaras L, Moschoyiannis S, Janicke H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Elsevier Journal of Information Security and Applications 2020; 50: 102419. doi: j.jisa.2019.102419

[9] Gamage S, Samarabandu J. Deep learning methods in network intrusion detection: A survey and an objective comparison. Elsevier Journal of Network and Computer Applications 2020; 169: 102767. doi: j.jnca.2020.102767

[10] Li J, Qu Y, Chao F, Shum HP, Ho ES, et al. Machine learning algorithms for network intrusion detection. In: Sikos L. (editor). AI in Cybersecurity. Cham, Switzerland: Springer International Publishing A&G, 2019, pp. 151-179.

[11] Uikey R, Gyanchandani M. Survey on classification techniques applied to intrusion detection system and its comparative analysis. In: International Conference on Communication and Electronics Systems; Cairo, Egypt; 2019. pp. 1451-1456.

[12] Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Elsevier Future Generation Computer Systems 2019; 100: 779-796. doi: j.future.2019.05.041

[13] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: IEEE Symposium on Computational Intelligence for Security and Defense Applications; Ottawa, ON, Canada; 2009. pp. 1-6.

[14] Toupas P, Chamou D, Giannoutakis KM, Drosou A, Tzovaras D. An intrusion detection system for multi-class classification based on deep neural networks. In: IEEE International Conference On Machine Learning And Applications; Boca Raton, Florida, USA; 2019. pp. 1253-1258.

[15] Vinayakumar R, Alazab M, Soman K, Poornachandran P, Al-Nemrat A, Venkatraman S. Deep learning approach for intelligent intrusion detection system. IEEE Access 2019; 7: 41525-41550. doi: 10.1109/ACCESS.2019.2895334

[16] Zhang H, Huang L, Wu CQ, Li Z. An effective convolutional neural network based on SMOTE and Gaussian Mixture Model for intrusion detection in imbalanced dataset. Elsevier Computer Networks 2020; 177: 107315. doi: 10.1016/j.comnet.2020.107315

[17] Elmasry W, Akbulut A, Zaim AH. Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. Elsevier Computer Networks 2020; 168: 107042. doi: 10.1016/j.comnet.2019.107042

[18] Zhang Y, Chen X, Jin L, Wang X, Guo D. Network intrusion detection: Based on deep hierarchical network and original flow data. IEEE Access 2019; 7: 37004-37016. doi: 10.1109/ACCESS.2019.2905041

[19] Vijayanand R, Devaraj D. A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network. IEEE Access 2020; 8: 56847-56854. doi: 10.1109/ACCESS.2020.2978035

[20] Pérez D, Alonso S, Morán A, Prada MA, Fuertes JJ, Domínguez M. Comparison of network intrusion detection performance using feature representation. In: International Conference on Engineering Applications of Neural Networks; Xersonisos, Crete, Greece; 2019. pp. 463-475.

[21] Khammassi C, Krichen S. A NSGA2-LR wrapper approach for feature selection in network intrusion detection. Elsevier Computer Networks 2020; 172: 107183. doi: 10.1016/j.comnet.2020.107183

[22] Abdulhammed R, Musafer H, Alessa A, Faezipour M, Abuzneid A. Features dimensionality reduction approaches for machine learning based network intrusion detection. MDPI Electronics 2019; 8 (3): 322. doi: 10.3390/electronics8030322

[23] Lee J, Park K. AE-CGAN model based high performance network intrusion detection system. MDPI Applied Sciences 2019; 9 (20): 4221. doi: 10.3390/app9204221

[24] Lee J, Park K. GAN-based imbalanced data intrusion detection system. Springer Personal and Ubiquitous Computing 2019; 25: 121-128. doi: 10.1007/s00779-019-01332-y

[25] Zhou Y, Cheng G, Jiang S, Dai M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. Elsevier Computer Networks 2020; 174: 107247. doi: 10.1016/j.comnet.2020.107247

[26] Wang A, Gong X, Lu J. Deep feature extraction in intrusion detection system. In: IEEE International Conference on Smart Cloud; Tokyo, Japan; 2019. pp. 104-109.

[27] Li X, Chen W, Zhang Q, Wu L. Building auto-encoder intrusion detection system based on random forest feature selection. Elsevier Computers & Security 2020; 95: 101851. doi: 10.1016/j.cose.2020.101851

[28] Alinn F, Chemchem A, Nolot F, Flauzac O, Krajecki M. Towards a hierarchical deep learning approach for intrusion detection. In: International Conference on Machine Learning for Networking; Paris, France; 2019. pp. 15-27.

[29] Ramsbrock D, Berthier R, Cukier M. Profiling attacker behavior following SSH compromises. In: IEEE/IFIP International Conference on Dependable Systems and Networks; Edinburgh, Scotland, UK; 2007. pp. 119-124.

[30] Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 2002; 16: 321-357. doi: 10.1613/jair.953

[31] Lin Z, Shi Y, Xue Z. Idsgan: Generative adversarial networks for attack generation against intrusion detection. arXiv preprint, arXiv:1809.02077, 2018.

[32] Svozil D, Kvasnicka V, Pospichal J. Introduction to multi-layer feed-forward neural networks. Chemometrics and Intelligent Laboratory Systems 1997; 39 (1): 43-62.

[33] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computation 1997; 9 (8): 1735-1780. doi: 10.1162/neco.1997.9.8.1735

[34] Cho K, Van Merriënboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint, arXiv:1409.1259; 2014.

[35] Maas AL, Hannun AY, Ng AY. Rectifier nonlinearities improve neural network acoustic models. In: ICML Workshop on Deep Learning for Audio, Speech and Language Processing; Atlanta, USA; 2013. pp. 3.

[36] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint, arXiv:1412.6980; 2014.