Evolving Systems
https://doi.org/10.1007/s12530-020-09364-z

ORIGINAL PAPER

Developing new deep-learning model to enhance network intrusion classification

Hanane Azzaoui[1] · Akram Zine Eddine Boukhamla[1] · David Arroyo[2] · Abdallah Bensayah[1,3]

Received: 19 November 2019 / Accepted: 27 December 2020

Abstract
Network traffic has recently known tremendous growth, and it is set to explode over the next few years. Alongside the increase in traffic, network attacks have become more complex, advanced, and efficient. Therefore, intrusion detection systems (IDS), among other countermeasures, must be adapted accordingly to the development of new threats, which implies the design of new detection methods with better accuracy and adaptability characteristics. Furthermore, methods training and validation can be conducted only on the grounds of adequate datasets. Therefore, using updated datasets and efficient classifiers are key factors. In this paper, we introduce a new Deep Neural Network (DNN) based IDS model for network traffic classification. Experimental analysis is carried out using both the CICIDS2017 dataset, which contains many new and up-to-date attacks alongside the well-known NSL-KDD dataset. The results are analyzed based on different performance metrics. The proposed model proves an accuracy of 99.43% and 99.63% using CICIDS2017 and NSL-KDD datasets, respectively. Furthermore, the performance of the proposed DNN model has been compared with the most recent schemes and higher accuracy is achieved.

Keywords Intrusion detection system · CICIDS2017 dataset · Deep neural network · Deep learning · NSL-KDD dataset

1 Introduction

During the last few decades, network attacks became complicated phenomena, which most systems suffer from frequently. These attacks keep evolving on an almost daily basis calling for continuous researches over different mechanisms against them. One widely known approach is to deploy a Network Intrusion Detection System (NIDS) to analyze, classify, and detect malicious traffic from normal traffic.

Network-based intrusion detection (NIDS) monitor and analyze network traffic by sniffing all inbound and outbound packets and searches for any suspicious traffic such as denial-of-service attacks, port scans, etc. This task is accomplished using either signature-based detection (Kumar 2012) or anomaly-based detection methods (Jyothsna et al. 2011).

The first category of NIDS is a signature-based intrusion detection system. This system can detect attacks by looking for specific patterns (signatures) (Kumar 2012). It can effectively detect previously known intrusions,SNORT is one famous light-weighted signature-based IDS (Zhou et al. 2010). This approach does not report a high false alarm rate because it uses pre-defined rules to identify recognized attacks (Gogoi et al. 2014), but still, it fails to detect zero-day attacks with unknown signatures. On the other hand, anomaly-based IDS, which is our interest in this paper, uses machine learning to train a model with normal and any other abnormal traffic, then, predict new behavior against this model (Jyothsna et al. 2011). This approach allows the detection of both known and new attacks.

Developing an accurate IDS model and validating it is a challenge and requires another important task, namely the selection of a training dataset. Only a few of them available for public use and they tend to be unbalanced. Another constraint regarding these datasets is that many of them are not very diversified, not sufficient, or may not reflect the actual status of cyber-attacks at any given time. Most recent researches are validated using the KDD-99 dataset (Tavallaee et al. 2009), NSL-KDD (Dhanabal and Shantharajah 2015), and since network threats evolve quickly, these

✉ Hanane Azzaoui
azzaoui.hanane@univ-ouargla.dz

[1] LINATI Laboratory, Department of Computer Science and Information Technologies, University of Kasdi Merbah, Ouargla, Algeria

[2] Spanish National Research Council, CSIC, Madrid, Spain

[3] Department of Mathematics, University of Kasdi Merbah, Ouargla, Algeria

Published online: 19 January 2021

datasets may not cover modern attacks and they may be considered as outdated datasets. These challenges produce the need for an up-to-date dataset constantly.

Previous works on IDS (Subba et al. 2016; Kim and Gofman 2018; Almi'ani et al. 2018; Chiba et al. 2018) used different machine learning methods, such as K-NN, SVM, C4.5, and shallow neural networks (ANN) on KDD'99 and NSL-KDD datasets. These studies encounter limits when it comes to proving the performance of the model against modern attacks. One of the most up-to-date datasets is CICIDS2017 (Sharafaldin et al. 2018) that contains records of benign traffic along with seven common network attack instances, which satisfies real-world criteria. Besides, authors from Kim and Gofman (2018) concluded that a deep neural network performs poorly as IDS classifiers. However, we prove that this is not the case for all datasets (Wolpert and Macready 1997).

In this paper, we propose an IDS model based on Deep Neural Network (DNN) with four layers then we compare its results against shallow neural networks and other classic machine learning approaches. Our model architecture was the result of building and comparing 36 model combinations with different hyper-parameters. We evaluated the performance of our model using the CICIDS2017 dataset, which gives our study more reliability. Moreover, to enrich our study more and make the comparison more objective with previous researches, we conducted the same experiments using the well-known NSL-KDD dataset.

The main contributions of this paper are:

- We introduce a new DNN-based IDS model architecture that is very effective against almost all network attacks.
- We performed extensive experimental analysis using the CICIDS2017 and NSL-KDD datasets to train and validate the proposed model against state-of-the-art IDS models.

The rest of the paper is organized as follows. Section 2 discusses recent related work. Section 3 presents CICIDS2017 and its properties. Section 4 describes the proposed deep neural network-based IDS model. Section 5 provides the experimental results. Finally, conclusions and future works are presented in Sect. 6.

2 Related work

Subba et al. (2016) proposed an intrusion detection system based on a three-layered ANN model. They evaluated their model using the NSL-KDD dataset. They used feature selection methods to reduce the feature vector to 36 features. After, they normalized the numeric attributes using the mean normalization method before feeding it to the model. Authors aimed to classify normal traffic versus four different attacks (Probe, DoS, U2R, and R2L). They reported a resulted accuracy of 95.05% for multi-classification using their model, and 98.86% correct classification in case of two-class (normal and attack) NSL-KDD.

As a comparison between shallow and deep neural networks for intrusion detection, Kim et al. (2018) tested a variety of shallow and deep neural network models on a preprocessed NSL-KDD dataset containing 148,000 instances of 41 features and 22 attacks. They reported that shallow neural network models perform better than deep models in case of intrusion detection. The authors reported 98.50% accuracy using a shallow model with one hidden layer of 17 nodes and 48.30% accuracy at best using deep models.

The intrusion detection system using a clustered version of Self-Organized Map (SOM) network is proposed in Almi'ani et al. (2018). Their system consists of two stages, SOM network and hierarchical agglomerative clustering using k-means applied on SOM neurons. The proposed work was demonstrated using the NSL-KDD benchmark dataset after min–max normalization, where they reported accuracy of 96.66% of binary classification, attack/normal connection instances.

Chiba et al. (2018) have developed an Anomaly Network Intrusion Detection System (ANIDS) based on Back Propagation Neural Network (BPNN). They performed a series of experiments in five working phases. The first phase is the determination of the parameters for ANIDS. Next, selection of a set of relevant values for intrusion detection for each parameter. In the third phase, they generate all possible combinations of values of these parameters. Finally, building the IDSs corresponding to all these combinations and realizing a comparison of the performance of twelve IDSs. The authors used KDD CUP 99 dataset to evaluate their approach, they reported accuracy of 99.10%. They come to conclude the best number of nodes for the hidden layer ($H = 0.75 \times \text{Input} + \text{Output}$) for their adopted architecture.

Based on a study by Sharafaldin et al. (2018) over eleven datasets since 1998 (DARPA98, KDD99, ISC2012, ADFA13, etc.). It has been found that most of these datasets are outdated and suffer from a lack of traffic diversity. This drove them toward generating a new dataset, CICIDS2017. In their paper, they evaluated the performance of CICIDS2017 using seven common machine-learning algorithms, which are KNN, Random Forest, ID3, Adaboost, MLP, Naive-Bayes, and QDA. The best results (accuracy) they reported were given by Random Forest 98%, also they used a Multilayer perceptron (MLP) but it didn't give a promising result (77.00%) comparing to other classifiers.

ANN Based Hybrid NIDS also showed impressive results, Chandrashekhar and Raghuveer (2014) applied different linear and non-linear data normalization methods on

KDDCup'99, then it has been given as input to network hybrid-IDS which consists of three layers, Fuzzy C-Means Clustering, ANN and Support Vector Machine.

(SVM). They compared the output results to determine the more relevant normalization technique for the intrusion detection dataset. From the analysis of the result, it is proved that Z-score (98.46%), Logarithmic (97.84%), and Decimal Scaling (97.08%) normalization techniques gave a better detection rate.

Bayesian networks event binary classification scheme has been proposed in Kruegel et al. (2003) to overcome a large number of false alarms and the lack of integration of additional information into the decision process by improving and the aggregation of different model outputs and allow one to seamlessly incorporate additional data. The authors achieved a significant reduction of false alarms as shown in their experimental results.

In Abdulhammed et al. (2019), the authors used two feature dimensionality reduction approaches, Auto-Encoder (AE) and Principle Component Analysis (PCA) to reduce CICIDS2017 dataset's features dimension from 81 down to 10. The resulting low-dimensional features are used to build various classifiers such as Random Forest (RF), Bayesian Network, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA) for designing an IDS. the accuracy is wobbling between 66% with 10 features and 96.7% with 60 features. For LDA with 10 and 40 features, the accuracy is fluctuating between 85% and 96.6%, respectively.

The authors in Yulianto (2019) used Synthetic Minority Oversampling Technique (SMOTE), PCA, and Ensemble Feature Selection (EFS) to improve the performance of AdaBoost-based IDS on CICIDS-2017 Dataset. The evaluation results show that the proposed AdaBoost classifier using EFS and SMOTE produces accuracy, precision, recall, and F1-Score of 81.83%, 81.83%, 100%, and 90.01% respectively.

Ahmim et al. (2019) proposed a novel hierarchical IDS combining different classifier approaches based on decision trees and rules-based concepts. They evaluated the proposed system using the CICIDS2017 dataset to classify the network traffic as Attack/Benign. The authors reported an accuracy of 99,665% with a detection rate of 94.457%.

3 CICIDS2017 dataset

As network modern attacks evolve on a daily basis, the use of old datasets may not lead to desired and objective results. Thus, we opted for using one of the most recently published IDS datasets, which is the CICIDS2017 dataset.

To overcome older datasets and meets real-world criteria, Canadian Institute for Cyber-security has created, among other newer datasets, the CICIDS2017 dataset among other newer ones (Sharafaldin et al. 2018). Some old datasets suffer from lack of traffic diversity, features set, metadata, and volumes. Some of them do not take into account the diversity of attacks, others anonymize traffic and ignore meaningful data payload. It is the most complete and up to date dataset according to Sharafaldin et al. (2018). CICIDS2017 meets all eleven criteria that must be considered to build an accurate benchmark dataset (Gharib et al. 2016) with updated attacks such as DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Portscan and Botnet.

CICIDS2017 was collected during 5 days under complete network configuration which includes a variety of network devices such as Modem, Firewall, and Routers…etc., and the presence of a variety of operating systems. Collecting realistic traffic was performed by profiling the behavior of human interactions and generating naturalistic benign background traffic. Besides, it covers six attack profiles: Brute Force Attack, Heartbleed Attack, Botnet, DoS Attack, DDoS Attack, Web Attack, and Infiltration Attack, these attacks are represented by 85 numeric and nominal features (Ring et al. 2019).

A recent study by Boukhamla and Coronel (2018) improved the performance of the classification process over CICIDS2017 by selecting the features that were more representative accurately, using Principal Component Analysis (PCA) procedure to end up with only 36 relevant features (75% of total features). The new PCA-dataset has been evaluated using three well-known classifiers.

3.1 Dataset preprocessing

Dataset preprocessing is an indispensable step, therefore, we needed to filter and clear out the CICIDS2017 dataset and select the most relevant features. It contains redundant records, irrelevant features, null values in all instances, and unknown/infinity values to be processed. From a total number of 85 features, we removed network-related features such as IP address, Flow ID and Timestamp, etc. Moreover, by removing zero-valued features we ended up with only 68 features.

Due to an enormous number of connection instances (2,299,308 records) and the lack of compatibility between instances numbers of some attacks compared to the rest, we decided to work on a custom subset of 356,510 records (about 15% of the original dataset) instead of the whole dataset. Also, we have replaced all unknown values with zeroes and infinities with a max positive number, as it is far greater than the maximum value of all features.

As shown in Table 1, some connection types (Benign, DoS, and PortScan) have an enormous number of records compared to others. Hence, we tried to equilibrate instances

Table 1 Data distribution of original and custom cicids2017 dataset

Traffic type	CICIDS2017		CUSTOM CIC-IDS2017	
	Instances	Percentage (%)	Instances	Percentage
BENIGN	1,743,153	75.81	100,000	28.04%
DDoS	128,027	5.56	80,000	22.43%
DoS	252,661	10.98	80,000	22.43%
PortScan	158,957	6.91	80,000	22.43%
Botnet	1966	0.085	1966	0.55%
BruteForce	13,835	0.60	13,835	3.88%
SQLInjection	21	0.0009	21	0.0058%
XSS	652	028	652	0.1828%
Infiltration	36	0.0015	36	0.01ù
Total	2,299,308	100	56,510	100%

numbers by choosing a custom subset of data by downsizing Benign, DDoS, DoS, and PortScan instances randomly.

3.2 Data normalization

Some inputs to ANN might not have a naturally defined range of values. In such a case, feeding these raw values into our network will not work very well. As the network has learned to work on values from the lower part of the range, while the actual inputs will be from the higher part of this range and possibly above range. However, a variety of practical studies proves that normalizing the inputs enhance the reliability of the trained network, make training faster and reduce the chances of being stuck in local optima (Chandrashekhar and Raghuveer 2014; Jayalakshmi and Santhakumaran 2011). We have been interested in two methods, Z-score normalization and Min–Max normalization following the formulas in Eqs. (1) and (2), respectively.

$$x_i = \frac{x_i - \mu}{\sigma} \tag{1}$$

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{2}$$

where μ is the mean, and σ is the standard deviation.

3.3 Performance assessment with k-fold cross-validation

To assess how the result of our model will generalize and estimate its skill we used cross-validation with fourfolds. Cross-validation is a resampling procedure, which has one parameter k that refers to the number of groups that a given dataset to be split into. And such, as we used fourfolds cross-validation, we divided each attack of the dataset into 4 groups, after, we merged three randomly chosen groups of each attack into one training sub-dataset to make sure that this training dataset has a fair share of all attack types.

Table 2 Proposed deep neural network model

Layer	Neurons number	Activation function
Input	68	None
Hidden #1	Input * 2	ReLu
Hidden #2	Input * 2	ReLu
Output	9	Softmax

4 Proposed model

Artificial Neural Network (ANN) is an information processing approach that tries to imitate the human brain. It consists of interconnected neurons collaborating to perform tasks. Generally, such a system learns to solve tasks by considering examples (training data) without using any task-specific rules. Deep Neural Network (DNN) is an artificial neural network with multiple layers between the input and output layer, which can model complex non-linear relationships.

In intrusion detection, DNN learns to recognize attacks (as DDoS attack), by analyzing a large enough set of labeled examples, then the model is used to identify new DDoS attack instances.

We used four-layers DNN with two hidden layers of 136 neurons with ReLu as an activation function. Table 2 shows the architecture of this network. The total number of trainable parameters is 29240.

Getting to this DNN model was not coincidental; our work is based on previous work (Sen et al. 2015; Lokeswari and Rao 2016; Gaidhane et al. 2014; Shah and Trivedi 2012; Kumar and Yadav 2014; Ghosh et al. 2015; Mukhopadhyay et al. 2011; Karsoliya 2012) where the efficiency of ANN models and their hyper-parameters have been discussed. We performed a series of experiments using several hyper-parameters combinations and ANN models, both deep and shallow. First, we selected the most relevant parameters used to construct ANN models, which are detailed in Table 3. After that, we ended up with 36 combinations of DNN models, constructing and testing these combinations gave us different results. After comparing model combinations we found that the best model to work with is the one described in Table 3, and as normalization method, we used Min–Max, this method proved its efficiency in most previous works (Chiba et al. 2018). Besides, we found that the

Table 3 Hyper-parameters combinations

Parameter	Combinations
Normalization method	Z-Score
	Min–Max
Activation function	Sigmoid
	ReLu
	Tanh
Number of neurons in hidden layers	2 * Input (Novel rule)
	0.75 * Input + Output (Chiba et al. 2018)
	(Input + Output)/2 (Gaidhane et al. 2014) (Arithmetic mean)
Optimization algorithm	Stochastic gradient descent
	Adam optimizer
Loss function	*Categorical cross-entropy*

Table 4 Train/test split of the custom CICIDS2017 dataset

Class	Train set	Test set
BENIGN	75,000	25,000
DDoS	60,000	20,000
DoS	60,000	20,000
PortScan	60,000	20,000
Botnet	1475	491
BruteForce	10,377	3458
SQL injection	16	5
XSS	489	163
Infiltration	27	9

Adam optimizer algorithm performed much better and there is no need to fix a learning rate in this case as it is adaptable according to features.

Choosing an activation function will affect the performance and speed of any neural network, which is why it is inevitable to use an activation function that will perform efficiently in an acceptable time. However, in the intrusion detection case, we ignored the speed criteria of this hyper-parameter, because overall, the calculation is not that complicated as it is in deep neural networks and we are dealing with shallow networks. As a result, we found that ReLu (Rectified Linear Units) performed better and faster than Sigmoid and Tanh as an activation function.

We used Accuracy, Detection Rate, and False alarm rate to measure the performance of DNN models. Accuracy is defined as the percentage of correct prediction, it is the fraction of elements correctly classified out of all the elements classified as positive. Whereas Detection Rate is the fraction of attacks detected by the system out of all positive detected elements. False alarm rate represents the ratio of instances misclassified as attacks by the benign instances in the dataset.

5 Experimental results

We implemented all machine learning techniques, candidate neural network-based IDS, and the proposed one using Python with the TensorFlow library (Abadi et al. 2016) on a machine with Intel I5-3210 (2.5 GHz) CPU and 8 G Memory.

5.1 Experiment #1 (using CICIDS2017 dataset)

For model analysis, we used the custom subset of the CICIDS2017 dataset (Sharafaldin et al. 2018) consisting of normal and attack instances as shown in Table 4. After the features selection stage, we found that six features had zeros as constant values, so they will not affect the classification

Table 5 The comparison of the proposed approach with other representative approaches

Publications	Model	Dataset	Type	Accuracy (%)	DR (%)	FPR (%)
Subba et al. (2016)	ANN	NSL-KDD	Multi-class	95.05	95.05	–
Kim and Gofman (2018)	ANN	NSL-KDD	Multi-class	98.50	–	1.40
Almi'ani et al. (2018)	Self-Organized Map Network	NSL-KDD	Binary-classification	83.46	96.66	0.279
Chiba et al. (2018)	BPNN	KDD-CUP99	Binary-classification	99.10	99.33	1.60
Sharafaldin et al. (2018)	Multi-Layer perceptron	CICIDS2017	Multi-class	–	77.00	–
Sharafaldin et al. (2018)	Random Forest	CICIDS2017	Multi-class	–	98.00	–-
Abdulhammed et al. (2019)	Auto-Encoder	CICIDS2017	Multi-class	–	98.90	0.001
Yulianto et al. (2019)	AdaBoost	CICIDS2017	Multi-class	81.83	81.83	–
Hosseini (2020)	GSPSO-ANN	NSL-KDD	Multi-class	–	94.40	–
		KDD Cup99	Multi-class	–	96.80	–
This paper	DNN	NSL-KDD	Multi-class	99.63	86.50	0.0011
This paper	DNN	CICIDS 2017	Multi-class	99.43	80.33	0.0007

process and they were excluded. This left 68 features to deal with. Table 5 shows the training/test data distribution of all attack types.

The class labels were numbered from zero to eight, label (0) is benign traffic. Thus, the output layer consists of nine nodes; each node represents a class label. The number of nodes in the input layer is set to 68, which is the number of features.

As each neural network has the best set of hyper-parameters to be chosen, which will lead to maximum accuracy, hyper-parameters tuning must be used to determine this set. It is the process of selecting the best values to initialize that leads to the best accuracy. We performed a series of experiments on 36 combinations to determine the best model architecture along with many other hyper-parameters like the optimizer used, loss function, normalization method, and the activation function. One last hyper-parameter we did not talk about, it is the number of epochs to train, for that, we observed training/testing accuracy development over epochs. We found that our DNN model shows maximum accuracy almost around epoch #30 on the CICIDS2017 dataset. Figure 1 shows that test accuracy is greater than training accuracy and this confirms the good generalization ability of the proposed ANN model.

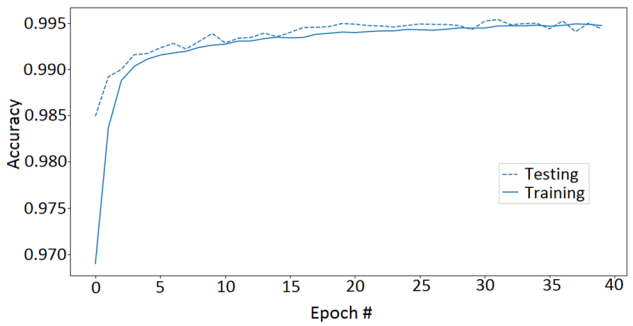

Fig. 1 DNN training/testing accuracy over epochs using the CICIDS2017 dataset

After setting up hyper-parameters, we used cross-validation with four rounds to assess how our model will generalize different train/test dataset splitting. We ran our model four times using different train/test sets each round, then we average results. The average accuracy we obtained is 99.43% with a low standard deviation $\sigma = 0.0033$.

Figure 2 shows detection rate of each attack type, we notice high detection rate for Benign, DDoS, DoS, PortScan, brute force and XSS traffic (98.82%, 99.29%, 99.89%, 99.93%, 99.55%, and 97.54% respectively) with very low false alarm rate as shown in Fig. 4. False alarm rate is low for all attacks as shown in Fig. 3, it is only 0.00279% for benign traffic which is the highest false-positive rate. PortScan and DoS have a very low false alarm rate of 0.00028% and 0.00003% respectively. Bot attack's detection rate is only 72.96%, which is considered low against other types.

SQL injection and infiltration attacks have 25.00% and 29.37% detection rate which are very low compared to other types. This is due to the small number of connection instances of these attacks (Only 21 and 36 records respectively), we expect a higher detection rate in case of enough data.

5.2 Experiment #2 (using NSL-KDD dataset)

The NSL-KDD dataset solved some of the problems of the KDDCUP'99 (Boukhamla and Coronel 2018; UNB 2020). This dataset has been used in much previous research. Therefore, we performed the same experiment we did use CICIDS2017 on the NSL-KDD dataset. This makes it possible to compare our results with previous proposals. NSL-KDD dataset includes 41 features, 125,973 instances, and has four classes as shown in Table 5.

This dataset has nominal features (Protocol_type, Service, Flag), and they cannot be handled by DNN directly,

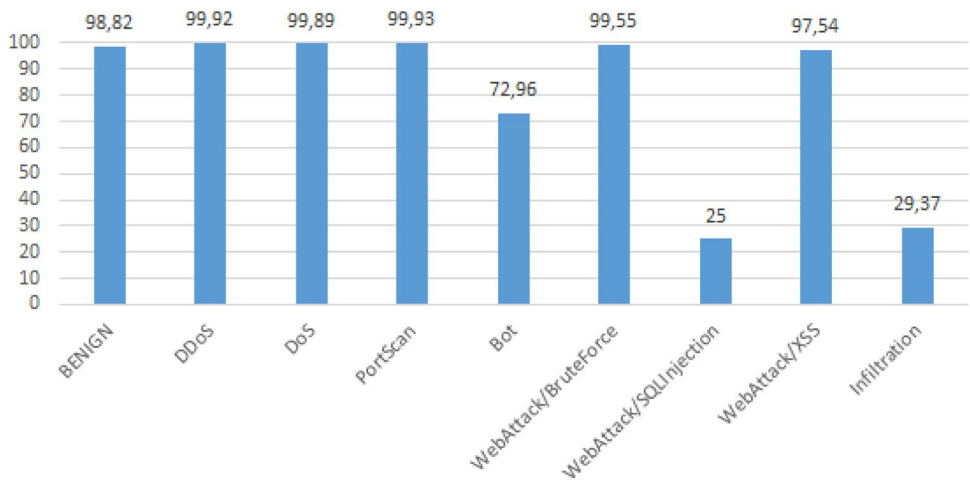

Fig. 2 The detection rate of the CICIDS2017 dataset

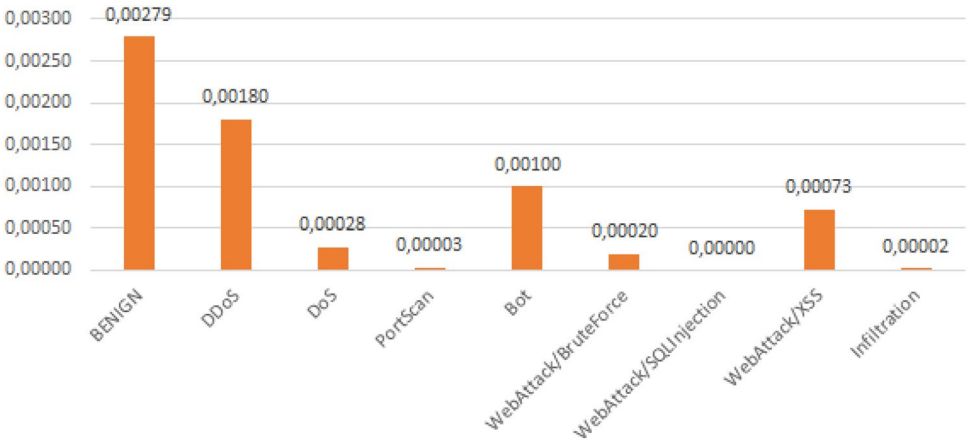

Fig. 3 The detection rate of the CICIDS2017 dataset

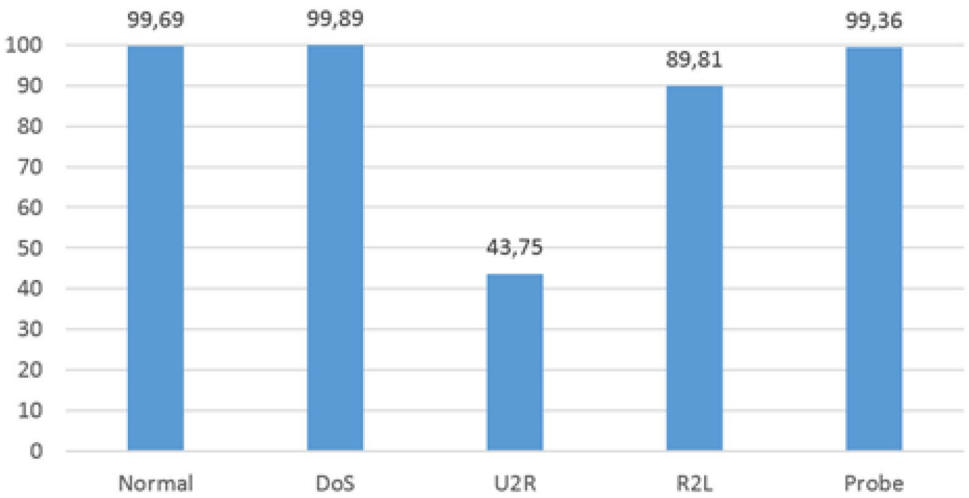

Fig. 4 Attacks detection rate of NSL-KDD dataset

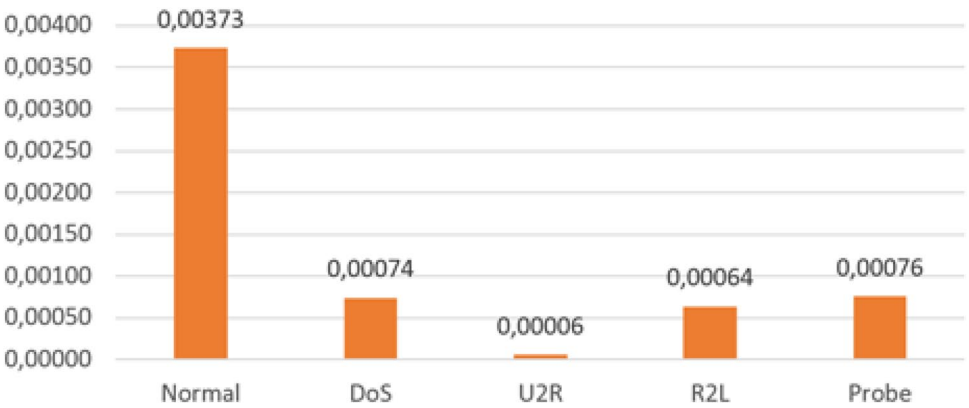

Fig. 5 False alarm rate of NSL-KDD dataset

so we converted all these nominal attributes into binary numeric attributes, this gave us a new dataset with 122 features. Then, we used a fourfold cross-validation. As an evaluation metric, we also used accuracy, detection rate, and false alarm rate. The average accuracy we obtained is 99.63% ($\sigma = 0.002$). Figure 4 shows the detection rate of the attacks.

Figures 4 and 5 shows that normal, DoS, and Probe instances have a very high detection rate and low false alarm rate as expected. The low detection rate of U2R and R2L is due to a lack of data (52 and 995 instances respectively).

Most existing researches on intrusion detection have been performed on older datasets. Table 5 summarizes a comparison between our approach and the most recent similar works

that focus on intrusion detection using both Artificial Neural Networks other machine learning. The nearest result to ours is presented by the work of Chiba et al. (2018) with an accuracy of 99.10%, but still, this study was performed on KDD-CUP'99 with binary classification, normal/attack classification. Sharafaldin et al. (2018) evaluated the performance of CICIDS2017 using a Multi-Layer perceptron (MLP) and reported a detection rate of 77.00%. Unfortunately, they did not provide much detail about the model's architecture nor its properties. On the other hand, the relatively low average detection rate is due to the small number of instances of SQL Injection and Infiltration attacks, only 43 records for both in the CICIDS2017 dataset. For the NSL-KDD dataset, the low number of U2R attack instances of only 11 records reduced the average detection rate.

6 Conclusion

With the rapid growth of network traffic and the advancement of cyber-attacks as well, the need to use more effective and accurate IDS has become more urgent. A new model based on Deep Neural Networks has been proposed in this paper. This model is composed of four layers, and by using hyper-parameters tuning on 36 model combinations, we arrived to choose a set of hyper-parameters that delivered the best result.

The experiments were conducted using the CICIDS2017 dataset that overcomes old datasets and contains many updated attack instances to cover most of the modern attack. Besides, to give our work more reliability for the sake of comparison, we perform the same experiments on the well-known NSL-KDD dataset that has been used by most of the previous works. The performance of our DNN model achieved an average accuracy of 99.43% ($\sigma = 0.0033$) and 99.63% ($\sigma = 0.002$) using CICIDS2017 and NSL-KDD dataset respectively.

For our future work, we intend to extend the range of hyper-parameters and examine deeper neural networks to further enhance our model.

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S (2016) Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467

Abdulhammed R, Musafer H, Alessa A, Faezipour M, Abuzneid A (2019) Features dimensionality reduction approaches for machine learning based network intrusion detection. Electronics 8:322

Ahmim A, Maglaras L, Ferrag MA, Derdour M, Janicke H (2019) A novel hierarchical intrusion detection system based on decision tree and rules-based models. In: 2019 15th International conference on distributed computing in sensor systems (DCOSS), IEEE

Almi'ani M, Ghazleh AA, Al-Rahayfeh A, Razaque A (2018) Intelligent intrusion detection system using clustered self-organized map. In: Fifth international conference on software defined systems (SDS), pp 138–144

Boukhamla A, Coronel J (2018) CICIDS2017 dataset: performance improvements and validation as a robust intrusion detection system testbed. Int J Inform Comput Secur 9

Chandrashekhar AM, Raghuveer K (2014) Improvising an intrusion detection precision of ANN based hybrid NIDS by incorporating various data normalization techniques—a performance appraisal. Int J Res Eng Adv Technol 2(2):1–7

Chiba Z, Abghour N, Moussaid K, El-omri A, Rida M (2018) A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection. Comput Secur. https://doi.org/10.1016/j.cose.2018.01.023

Dhanabal L, Shantharajah SP (2015) A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. Int J Adv Res Comput Commun Eng 4(6):2319–5940

Gaidhane R, Vaidya C, Raghuwanshi M (2014) Intrusion detection and attack classification using back-propagation neural network. Int J Eng Res Technol 3(3):1112–1115

Gharib A, Sharafaldin I, Lashkari AH, Ghorbani AA (2016) An evaluation framework for intrusion detection dataset. In: International conference on information science and security (ICISS), IEEE.

Ghosh P, Mandal AK, Kumar R (2015) An efficient cloud network intrusion detection system. Information systems design and intelligent applications. Springer, Berlin, pp 91–99

Gogoi P, Bhattacharyya DK, Borah B, Kalita JK (2014) MLH-IDS: a multi-level hybrid intrusion detection method. Comput J 57(4):602–623. https://doi.org/10.1093/comjnl/bxt044

Hosseini S (2020) A new machine learning method consisting of GA-LR and ANN for attack detection. Wirel Netw 26(6):4149–4162

Jayalakshmi T, Santhakumaran A (2011) Statistical normalization and back propagation for classification. Int J Comput Theory Eng 3(1):1793–8201

Jyothsna VV, Prasad VR, Prasad KM (2011) A review of anomaly based intrusion detection systems. Int J Comput Appl 28(7):26–35

Karsoliya S (2012) Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. Int J Eng Trends Technol 3(6):714–717

Kim DE, Gofman M (2018) Comparison of shallow and deep neural networks for network intrusion detection. In: Computing and communication workshop and conference (CCWC) 2018 IEEE 8th Annual, pp 204–208

Kruegel C, Mutz D, Robertson W, Valeur F (2003) Bayesian event classification for intrusion detection. In: 19th Annual computer security applications conference, Proceedings. Las Vegas, NV, USA, 2003, pp 14–23

Kumar V (2012) Signature based intrusion detection system using SNORT. Int J Comput Appl Inf Technol 1(3):35–41

Kumar S, Yadav A (2014) Increasing performance of intrusion detection system using neural network. In: International conference advanced communication control and computing technologies (ICACCCT), IEEE, pp 546–550. https://doi.org/10.1109/icaccct.2014.7019145

Lokeswari N, Rao BC (2016) Artificial neural network classifier for intrusion detection system in computer network. In: Proceedings of the second international conference on computer and communication technologies, Springer India, pp 581–591. https://doi.org/10.1109/NCC.2016.7561088

Mukhopadhyay I, Chakraborty M, Chakrabarti S, Chatterjee T (2011) Back propagation neural network approach to Intrusion Detection System. In: Recent trends in information systems (ReTIS), IEEE, pp 303–308

Ring M, Wunderlich S, Scheuring D, Landes D, Hotho A (2019) A survey of network-based intrusion detection data sets. Comput Secur 86:147–167

Sen R, Chattopadhyay M, Sen N (2015) An efficient approach to develop an intrusion detection system based on multi-layer back-propagation neural network algorithm: IDS using BPNN algorithm. In: Proceedings of the 2015 ACM SIGMIS conference on computers and people research, ACM, pp 105–108

Shah B, Trivedi BH (2012) Artificial neural network based intrusion detection system: a survey. Int J Comput Appl 39(6):13–18. https://doi.org/10.5120/4823-7074

Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: 4th international conference on information systems security and privacy (ICISSP), Portugal

Subba B, Biswas S, Karmakar S (2016) A neural network based system for intrusion detection and attack classification. In: 2016 Twenty second national conference on communication (NCC), Guwahati, pp 1–6

Tavallaee M, Bagheri E, Lu W, Ghorbani A (2009) A detailed analysis of the KDD CUP 99 data set. In: Submitted to second IEEE symposium on computational intelligence for security and defense applications (CISDA), pp 1–6

UNB (2020) Nsl-kdd data set for network-based intrusion detection systems. http://nsl.cs.unb.ca/kdd/nslkdd.html. Accessed 14 Aug 2019

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82

Yulianto A, Sukarno P, Suwastika NA (2019) Improving AdaBoost-based intrusion detection system IDS performance on CIC-IDS-2017 Dataset. J Phys Conf Ser 1192:12–18

Zhou Z, Zhongwen C, Tiecheng Z, Xiaohui G (2010) The study on network intrusion detection system of Snort. In: 2010 International conference on networking and digital society, Wenzhou, pp 194–196

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com