Analysis of Machine Learning Algorithms for Network Intrusion Detection Systems

Bedirhan UĞUZ, Özhan SUAT, İlker SIĞIRCI

Computer Engineering Department, METU

1.Introduction

For the past few years, network intrusion attacks are becoming more sophisticated and network traffic data has become enormous in volume, velocity and variance. In other words, we have started to observe the big data characteristics. Recent technological developments gave us an ability to process large volume, velocity and high variance data and it led researchers to machine learning methods. Thus, most of the intrusion detection systems embarked on using machine learning tools and algorithms. In this project, we reviewed and compared the most used machine learning algorithms performances on different datasets and attack types. We selected three popular intrusion detection datasets that are NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018. Applied machine learning algorithms are evaluated based on their accuracy, precision,sensitivity(recall) and f-score.

2. Related Work

Machine Learning techniques on IDS have been used for both benchmarking and real life testing purposes over a decade. Hence we have gathered some important research papers that are similar to our work. A research about intrusion detection over the last few decades is given in [1] which the authors conclude Hybrid Machine Learning techniques have been used vastly. In [2] misuse detection followed anomaly detection was proposed. In the paper [3], genetic algorithms and decision trees were used for automatic rule generation for improving the capability of IDS. In the papers [4] and [5], integrated utilization of neural networks in IDS was suggested.

In the paper [6], an application of recurrent neural networks approach is proposed. In paper [7], they compared the performance of the neural network models for anomaly detection on datasets that represent four different scenarios.

3. Datasets

We chose to work with three main datasets for our project. These are: NSL-KDD, UNSW-NB15, CSE-CIC-IDS2018.

NSL-KDD is an enhanced version of KDD’99 which has been widely used by many researchers for benchmarking the Intrusion Detection System algorithms over the years. It has 4 main attack types with properly distributed form and it doesn’t have any redundant records. It has a reasonable amount of records and that makes it easy to test the machine learning models on it. Moreover, it splits into train and test sets with the equal number of features for each dataset. Hence, NSL-KDD is chosen as the first dataset.

KDD and NSL-KDD are a little bit older and they are not representing the current network threat environment completely. Therefore, UNSW-NB15 was proposed. It has different attack types and is generated as a hybrid of real modern normal activities and synthetic contemporary attack behaviours which is a different way to produce an intrusion detection dataset. Also, we could find lots of research that we can use as a guideline while producing and analyzing the results. Thus, it is selected as the second dataset.

CSE-CIC-IDS2018 is proposed by Communications Security Establishment and Canadian Institute for Cybersecurity as a collaborative project. It is newer than NSL-KDD and UNSW-NB15. It is generated in a systematic manner in order to produce a diverse and comprehensive benchmark dataset. Moreover, it is generated in a well designed infrastructure which has lots of attacking machines, victim machines and servers. After UNSW-NB15, it is a good candidate for reviewing algorithms performance in an environment that evolves over time. Therefore, CSE-CIC-IDS2018 is chosen as the third dataset.

NSL-KDD

In 1999, International Knowledge Discovery and Data Mining Tools Competition was held with the goal of collecting traffic records. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between “bad’’ connections, called intrusions or attacks, and “good’’ normal connections. As a result of this competition, those records are collected and bundled into a dataset called KDD '99.

However, KDD’99 dataset has some major problems as mentioned in [8]. Hence, we chose NSL-KDD instead.

NSL-KDD data set was brought into existence, as a revised, cleaned-up version of the KDD’99 from the University of New Brunswick.

We chose NSL-KDD instead of KDDCup19 because of its following advantages:[9]

1. It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.
2. There are no duplicate records in the proposed test sets; therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.
3. The number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which makes it more efficient to have an accurate evaluation of different learning techniques.
4. The number of records in the train and test sets are reasonable, which makes it affordable to run the experiments.

The dataset contains 43 features in total. The first 41 of them refer to the traffic input while the last two of them refer to the attack label and its score(severity of the attack).

In the dataset, attack labels fall into four main categories:

DoS = ['back.','land.','neptune.','pod.','smurf.', 'teardrop.']

R2L = ['ftp_write.','guess_passwd.','imap.','multihop.','phf.','spy.','warezclient.','warezmaster.']

U2R = ['buffer-overflow.','loadmodule.','perl.','rootkit.']

Probe = ['ipsweep.','nmap.','portsweep.','satan.']

However, there is an imbalance problem which can add bias to our models as can be seen in Figure 1. Mainly, most of the records in the dataset are 'normal' traffic, which means they are not any attack types. Moreover, the number of U2R and R2L are respectively low compared to other attack types. Although this exactly represents a real life case, we needed to handle these imbalances when preprocessing the dataset.

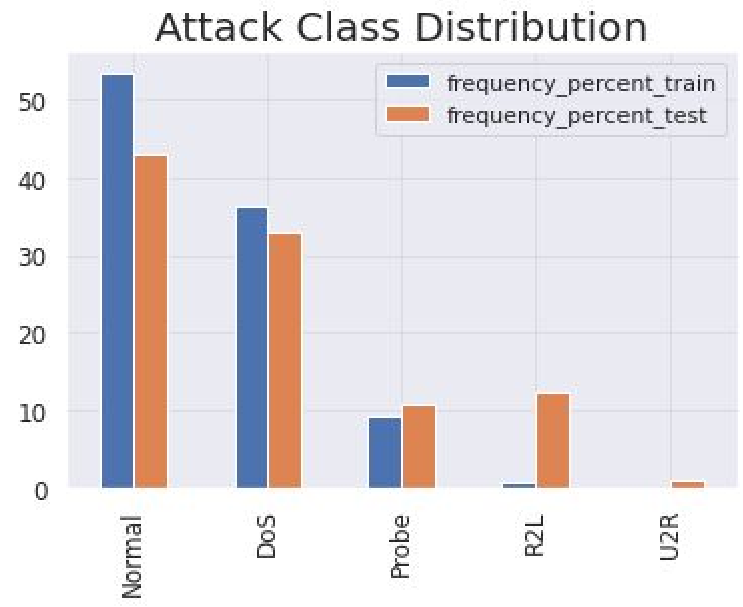

Figure 1: Attack Class Distribution of NSL-KDD

Lastly, the feature types in this dataset can be broken down into 4 types:(Numbers represent corresponding column number): 4 Categorical (2, 3, 4, 42), 6 Binary (7, 12, 14, 20, 21, 22).23 Discrete (8, 9, 15, 23–41, 43), 10 Continuous (1, 5, 6, 10, 11, 13, 16, 17, 18, 19)

Preprocessing

Before applying any machine learning model, we needed to preprocess the dataset in order to normalize its rows. Since the KDD-NSL dataset is more enhanced and strict from the original KDDCup99 dataset, it doesn't contain any NaN or empty values within it. That means we didn't have to do any fit-transform operation for it. Hence, we have focused on eliminating categorical columns.

As mentioned in the structure of the dataset, NSL-KDD dataset contains 4 categorical columns, which means they contain non-numeric values. Since machine learning models can't take these values as input, we needed to convert them into numerical types. Before starting this conversion, we have dropped the last column. This column gives information about attack score which doesn't have any contribution when finding the label of the attack.

After that, we have left with 3 categorical columns which are 'protocol_type', 'service' and 'flag'. Since their distribution is fairly even, we have created dummy columns for all of those. To be specific, we have created 84 new dummy columns. After that we have converted those columns with Label Encoder in order to give them as input to One Hot Encoder. One Hot Encoder creates a binary column for each category and returns a corresponding matrix for that category. Hence, we have successfully encoded all the categorical columns into numeric types.

Feature Scaling:

The features are scaled with Standard Scaler in order to avoid features with large values which may impact too much in the results.

Feature Selection

Since the dataset contains 122 features, we needed to select the most important ones that represent the dataset fully. The reason for feature selection was when there are lots of features which impact is low, that makes the prediction harder. Hence, we have chosen to select the most 13 features with Recursive Feature Elimination (RFE). With this way, we got a subset of features that represent the data most.

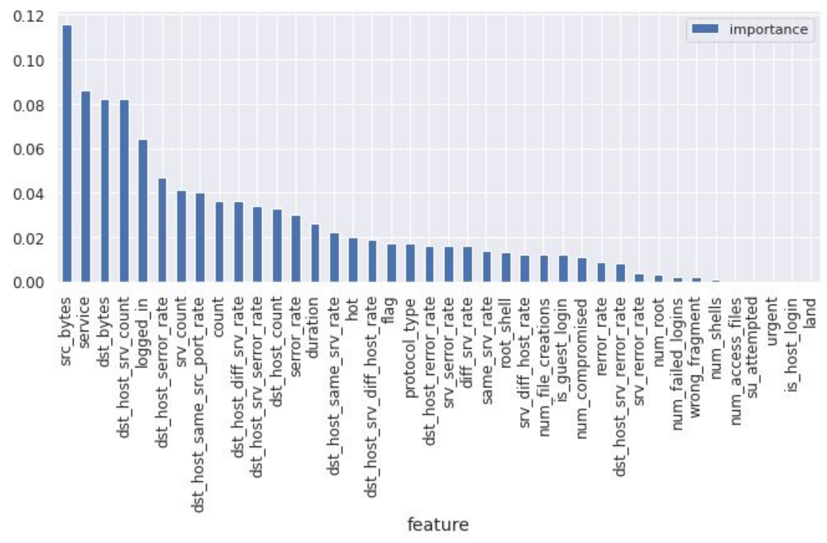

Figure 2: Feature Importance Histogram

UNSW-NB15

KDD98, KDDCUP99 and NSL-KDD datasets were generated two decades ago. They are not representing the current network threat environment completely. According to a research [10], there are three major issues why KDD99 and NSL-KDD do not give good results. First, their lack of modern low footprint attack styles. Second, their lack of modern normal traffic scenarios, and last a different distribution of training and testing sets. Therefore, we run analysis on the UNSW-NB15 dataset.

UNSW-NB 15 dataset includes raw network packets and was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). This dataset includes a hybrid of real modern normal activities and synthetic contemporary attack behaviours. This allows us to work with up to date data. [11]

This dataset has nine types of attacks, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. There are a total 49 features in dataset including attack types. 28 integers, 5 strings, 2 binaries and 10 float feature columns in the dataset. The Argus and Bro-IDS tools are used to generate these 49 features.

The total number of records is 2,540,044 which are stored in the four CSV files.

Preprocessing

First, we have extracted category and label columns from other features. In this dataset, a features list is provided in a csv file. There are 4 types of columns. Nominal, integer, binary and float. These are described in the feature list csv file. We have grouped different types of columns. Then, we read actual data from 4 different files and concatenate all data into a single dataframe. Since we use pandas to read data, pandas use its default data type. We realized that some columns data types are erroneous after reading. So we convert integer, binary and float columns to numeric columns.

There are some missing rows that have NaN instead of attack category. We replaced NaN values with "normal" which indicate that the row represents normal network representation.

There are two binary columns. One of them shows whether source and destination addresses are equal. For this column, we have replaced Nan values with 0 which represents general behaviour of the dataset. The other column represents if it is ftp session and accessed with login. If it is not ftp, it is Nan. So we have replaced it with 0 value. For nominal columns, we lower all values to ensure unity of the columns.

As the last step, since we cannot know actual values of other NaN value rows, we have dropped all NaN value included rows from the resulting dataset.

Feature Selection

We tried a few methods to make the feature selection part accurate. There is a feature_importance property in sklearn tree based classifiers. First, we trained ExtraTreesClassifier which is a widely used technique and some research that uses the same dataset selects features with this method. It creates an array called feature importance that includes column indexes according to their importances. We select the best 20 features according to this feature selection method.

Then, we tried the SelectKBest method to ensure that feature_importance gives accurate results. It has different results from the SelectKBest method. We stored both results and tried models with both feature sets. The SelectKBest method gives slightly more accurate results compared to the feature_importance method.

CSE-CIC-IDS2018

It is a collaborative project between the Communications Security Establishment(CSE) and the Canadian Institute for Cybersecurity(CIC). As network behaviours and intrusion techniques evolve, researchers need to generate a new intrusion detection dataset which is dynamically generated, reproducible, extensible, and modifiable. A systematic approach is employed to generate datasets in order to evaluate, test and analyze the intrusion detection systems. There are seven different attack scenarios such as DoS, Infiltration of the network from inside, DDoS, Botnet, Brute-force, Heartbleed. Also, infrastructure consists of 50 machines and the victim organization has 5 departments and a total of 420 machines and 30 servers. [12]

Researchers extracted all features by using CICFlowMeter. It is a network traffic flow generator and analyser. It can generate bidirectional flows and more than 80 statistical network traffic features. It is developed and used by Canadian Institute for Cybersecurity. The overall dataset is collected in several days and stored in the Amazon Web Services ecosystem by using Amazon S3 services. The total amount of the data is 220.8 GB. Therefore, we have considered only Wednesday and Thursday data individually.

First, we took Thursday data from the dataset. It contains DoS attacks data. It has 1048575 data rows with 80 features. 996077 rows are labeled as *Benign*, 41508 rows are labeled as *GoldenEye* and the rest 10990 rows are labeled as *Slowloris*. Both are DoS attack test tools. *GoldenEye* uses HTTP Keep Alive + NoCache as attack vector. *Slowloris* tries to exhaust the server's thread pool so that they can not answer any other people.

Secondly, we took Wednesday data from the dataset. It contains only Infiltration attacks data. It has 609030 data rows with 80 features. 540568 rows are labeled as *Benign* and 68462 rows are labeled as *Infiltration*. In this scenario, researchers send a malicious file by using email and exploit other applications' vulnerabilities. After success, they used the victim machine in order to scan the internal network for other vulnerable targets. According to a research which is conducted on this dataset, most of the machine learning algorithms can not detect *Infiltration* sufficiently.[13] In addition, another research which uses convolutional neural networks stated that *Infiltration* could not be detected efficiently.[14]

Since both of the days' features are the same and were extracted by using the same technique, preprocessing phases are exactly the same.

Preprocessing

The raw datasets need to be processed before applying feature selection techniques in order to eliminate unnecessary features from the feature set and finding correlations between features. While examining the data, we encounter some of the rows have *nan* values on some columns. Most of them belong to the Benign class and eliminating these rows would not affect our machine learning algorithm applications. Therefore, we directly eliminated these rows. In addition, some of the rows have *inf* values in some of their columns. These values will affect our feature scaling process, so those rows also are removed from the dataset. For the Thursday data, we ended up with a total of 1040548 rows. 988050 (95%) of them are labeled as *Benign* and the rest of rows are merged under the *Malicious* label. For the Wednesday data, we ended up with a total of 606902 rows. 538666 (89%) of them are labeled as *Benign* and the rest of the rows are labeled as *Malicious*. Imbalance problem is obvious and while training some models like multi-layer perceptrons, we need to take into account some techniques that address imbalance problems. After those refinements, we have one categorical variable which is the *Protocol* column. By using one hot encoding, we have encoded this variable.

Feature Selection

For both of the dataset, we have 80 features in total and we want to eliminate unnecessary data columns by selecting a subset of features. These operations will help us to reduce computations. We applied two feature selection techniques for this data set and then we merged two results into a one feature set. First, we applied Univariate feature selection with ANOVA F-test. Then we have used the SelectKBest method which is implemented in the Scikit-learn in order to select top features. Secondly, we ran a vanilla random forest algorithm on the datasets which will give us each feature importance value. Then by using SelectFromModel from Scikit-learn, we've selected the best features that are higher than the default threshold. Finally, we merged these two feature lists into a feature set.

4. EXPERIMENTS

NSL-KDD

We have implemented and run Logistic Regression, Naive Bayes, K Nearest Neighbor, Decision Tree Classifier, Adaboost, Random Forest, SVM algorithms on the dataset. As it can be seen in the table below , tree based algorithms such as Random Forest, Decision Tree, and Adaboost gave the best results. The result of Logistic Regression is worse than other algorithms because it couldn't converge. Beside Logistic Regression, the result of SVM was less than others. Moreover, the running time of the SVM algorithm was not applicable since it took 4x more time than all the other algorithms. Even doing parameter tuning with using different kernels(rbf, linear) didn't change the outcome.

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.90	0.97	0.80	0.88
Naive Bayes	0.87	0.97	0.72	0.83
K Nearest Neighbor	0.98	0.98	0.99	1.00
Decision Tree	0.99	0.99	0.99	0.99
Adaboost	0.99	1.00	1.00	0.98
Random Forest	1.00	0.98	1.00	1.00
SVM	0.93	0.95	0.85	0.91

Table 1: The Performance Examination on All 122 Features

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.92	0.97	0.82	0.90
Naive Bayes	0.89	0.98	0.75	0.90
K Nearest Neighbor	0.99	1.00	0.99	0.99
Decision Tree	1.00	1.00	0.99	1.00
Adaboost	1.00	1.00	1.00	1.00
Random Forest	1.00	1.00	1.00	1.00
SVM	0.93	0.96	0.86	0.91

Table 2: The Performance Examination on Selected 13 Features

When experimenting, we have followed two different approaches. First one is making the experiments with all of 122 features and the second one is making them with 13 features that are selected with RFE. We saw that the experiments with 13 selected features gave slightly more accuracy as can be seen in the tables. Hence, we have evaluated our models according to results of selected feature experiments. In addition, as said before, the dataset contains 4 main attack types which are DoS, Probe, R2L, U2R. Among them, we have seen that we can predict DoS attacks more than other attacks types.

UNSW-NB15

We applied Adaboost, Decision Tree, Extra Trees Classifier, K Nearest Neighbor, Decision Tree Classifier, Random Forest Classifier, Logistic Regression on UNSW-NB15 dataset. With this dataset, accuracy and precision results are not good enough because some non-attacks are classified as attacks on some algorithms. With parameter tuning, we increased precision but this caused a decrease in recall. Decrease in recall is worse than decrease in precision. Because all attacks shall be captured. Recall represents TP (True Positive) / FN (False Negative). If it is close to 1.0, it shows we captured all attacks successfully.

Algorithm	Accuracy	Precision	Recall	F1-Score
Adaboost	0.91	0.61	0.82	0.70
Extra Trees Classifier	0.99	0.91	1.00	0.95
K Nearest Neighbor	0.94	0.44	0.79	0.56
Decision Tree Classifier	0.98	0.89	0.97	0.93
Random Forest Classifier	0.98	0.88	0.97	0.92
Logistic Regression	0.89	0.53	0.84	0.65

Table 3: The Performance Examination on UNSW-NB15

Features are selected with SelectKBest method which gives slightly higher results than feature importance method which can be generated from tree based classifiers. Contrary to expectations, Adaboost doesn't give high results with this data set. This data set includes different types of attacks. Even if we classify data as attack or not attack, the model should be suitable with multiple classes to successfully predict results. Also K Nearest Neighbor doesn't give an accurate result. It classifies most of the non-attacks as attacks as can be seen from precision value. Most accurate algorithm is ExtraTreesClassifier. This algorithm predicts only a small chunk of non-attack as attack and %99 of its predictions are correct.

We checked other work results to compare our results and according to a work [15], our extra trees classifier result which is our best result nearly matching. Both have captured nearly all packets and their precision is slightly higher than our work. Their precision is %92 and our precision is %91.

CSE-CIC-IDS2018

We have implemented and run Logistic Regression, Random Forest, Naive Bayes, K Nearest Neighbor, Decision Tree Classifier, Adaboost, Multi Layer Perceptrons, and Quadratic Discriminant Analysis algorithms on the dataset. According to our literature review, these methods are the most popular ones that are used with this dataset. While examining results, we must consider precision, recall and f1-score because there is an imbalance problem and accuracy can be very deceptive.

First, we examine Thursday's dataset which contains *DoS* attack type.

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.97	0.91	0.53	0.67
Random Forest	1.00	1.00	1.00	1.00
Naive Bayes	0.86	0.14	0.33	0.19
K Nearest Neighbor	0.99	0.99	0.99	0.99
Decision Tree	1.00	1.00	1.00	1.00
Adaboost	1.00	1.00	1.00	1.00
Multi Layer Perceptron	0.95	1.00	0.00	0.01
Quadratic Discriminant Analysis	0.99	0.99	0.98	0.98

Table 4: The Performance Examination Results on CSE-CIC-IDS2018 Thursday’s Dataset

As it is seen in the table, Random Forest, Decision Tree, and Adaboost gave the best results. Logistic Regression model did not converge after 10000 iterations, so it could not fit a line that separates the data. Therefore, recall is lower than other methods.

Since there is an imbalance problem in the dataset, we have experienced an overfitting in Multi Layer Perceptron. Thus, the model predicts all the data points as *Benign*. In order to train the model, extra work is needed in the preprocessing state. Undersampling on *Benign* samples can be a solution.

Random Forest, Decision Tree, and Adaboost algorithms worked very well. However, execution times are very different. Random Forest is three times faster than Decision Tree and ten times faster than Adaboost, so we can say that the Random Forest algorithm is the best algorithm on Thursday's dataset with the shortest execution time and highest accuracy.

Second, we examine Wednesday's dataset which contains *Infiltration* attack type. As mentioned earlier, this attack type could not be detected well with these machine learning algorithms, but again we have applied the same algorithms with Thursday's dataset.

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.89	0.00	0.00	0.00
Random Forest	0.83	0.05	0.03	0.03
Naive Bayes	0.85	0.18	0.09	0.12
K Nearest Neighbor	0.88	0.18	0.03	0.05
Decision Tree	0.82	0.08	0.06	0.06
Adaboost	0.89	0.00	0.00	0.00
Multi Layer Perceptron	0.77	0.13	0.20	0.16
Quadratic Discriminant Analysis	0.81	0.17	0.18	0.17

Table 5: The Performance Examination Results on CSE-CIC-IDS2018 Wednesday's Dataset

In this table, we can easily observe that accuracy is not a good metric for our case because if we examine logistic regression and Adaboost results, we see that accuracy is better than other algorithms. However, these two models predict every data point as innocent and since the dataset is imbalanced, it gives higher accuracy.

As we mentioned in the dataset review section and literature review, we were expecting these results. Actually, those algorithms could not detect the attack well, but among all these methods we can say that Quadratic Discriminant Analysis and Multi Layer Perceptron are better than other algorithms.

5. CONCLUSION

As a result, we apply most used methods on 3 datasets. According to the results we get more accurate results when we use tree based classifiers.

First dataset we worked with is NSL-KDD. Since NSL-KDD is a matured dataset, it gives the most accurate results on tried methods. Tried methods detect nearly all attacks and non-attacks correctly. However it doesn’t represent contemporary attacks and communication patterns.

UNSW-NB15 dataset gives slightly lower results than other datasets because it contains variate of attacks and doesn’t fit models easily. It gives the most accurate result with an extra trees classifier. This classifier is only tried with UNSW-NB15 because this dataset contains more diverse features than others. Extra trees classifier creates a random tree for each feature and this method is the most suitable method for the dataset.

CSE-CIC-IDS2018 is also a contemporary network dataset. Its results are more accurate than UNSW-NB15. However, both dataset gives more accurate results with tree based classifiers. Random forest and decision tree classifiers are best methods for this dataset. Random forest is very similar to extra trees classifier. Thus, we can say that a random forest classifier or extra trees classifier is the most suitable method for our 3 dataset and can be used in intrusion detection systems.

REFERENCES

[1] Y. Hamid, M. Sugumaran, and V. Balasaraswathi, “IDS Using Machine Learning - Current State of Art and Future Directions,” British Journal of Applied Science & Technology, vol. 15, no. 3, pp. 1–22, Jan. 2016.

[2] J. Zhang and M. Zulkernine, “A hybrid network intrusion detection technique using random forests,” in First International Conference on Availability, Reliability and Security (ARES’06), 2006, p. 8–pp.

[3] C. Sinclair, L. Pierce and S. Matzner. ”An application of machine learning to network intrusion detection”. In Proceedings of the 15th annual computer security applications conference (ACSAC), pp. 371377. IEEE Compute

[4] H. Debar, M. Becker and D. Siboni. ”A neural network component for an intrusion detection system”. In Proceedings of the IEEE computer society symposium on research in security and privacy, pp. 240250. IEEE Computer Society, 1992. DOI http://dx.doi.org/ 10.1109/risp.1992.213257.

[5] J. Cannady. ”Artificial neural networks for misuse detection”. In Proceedings of the 1998 national information systems security conference (NISSC), pp. 443456. Citeseer, 1998.

[6] H. Debar and B. Dorizzi. ”An application of a recurrent network to an intrusion detection system”. In International joint conference on neural networks, 1992. IJCNN., vol. 2, pp. 478 483 vol.2. jun 1992. DOI http://dx.doi.org/10.1109/ijcnn. 1992.226942.

[7] Z. Zhang, J. Lee, C. Manikopoulos, J. Jorgenson and J. Ucles. ”Neural networks in statistical anomaly intrusion detection”. Neural network world, vol. 11, no. 3, pp. 305316, 2001.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A Detailed Analysis of the KDD CUP 99 Data Set,” Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

[9] “NSL-KDD dataset” [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[10] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.

[11] Moustafa, N., 2020. The UNSW-NB15 Data Set Description. [online] Unsw.adfa.edu.au. Available at:

<https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/> [Accessed 15 June 2020].

[12] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

[13] Q. Zhou, D. Pezaros, Evaluation of Machine Learning Classifiers forZero-Day Intrusion Detection– An Analysis on CIC-AWS-2018 dataset, arXiv preprint arXiv:1905.03685v1

[14] Kim Jiyeon, Shin Yulim, Choi Eunjung. An Intrusion Detection Model based on a Convolutional Neural Network. J Multimed Inf Syst 2019;6(4):165-172. https://doi.org/10.33851/JMIS.2019.6.4.165

[15] N. Moustafa, B. Turnbull and K. R. Choo, "An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4815-4830, June 2019, doi: 10.1109/JIOT.2018.2871719.