MLP4NIDS: An Efficient MLP-Based Network Intrusion Detection for CICIDS2017 Dataset

Arnaud Rosay, Florent Carlier, Pascal Leroux

MLP4NIDS: an efficient MLP-based Network Intrusion Detection for CICIDS2017 dataset

Arnaud Rosay[1], Florent Carlier[2], and Pascal Leroux[2]

[1] STMicroelectronics, 11 rue Pierre-Félix Delarue 72100 Le Mans, France
arnaud.rosay@st.com
http://www.st.com
[2] CREN, Le Mans Université, Avenue Olivier Messiaen, 72085 Le Mans, France
{florent.carlier, pascal.leroux}@univ-lemans.fr
http://cren.univ-nantes.fr/

Abstract. More and more embedded devices are connected to the internet and therefore are potential victims of intrusion. While machine learning algorithms have proven to be robust techniques, it is mainly achieved with traditional processing, neural network giving worse results. In this paper, we propose usage of a multi-layer perceptron neural network for intrusion detection and provide a detailed description of our methodology. We detail all steps to achieve better performances than traditional machine learning techniques with a detection of intrusion accuracy above 99% and a low false positive rate kept below 0.7%. Results of previous works are analyzed and compared with the performances of the proposed solution.

Keywords: Machine Learning, Multi-Layer Perceptron, Network Intrusion Detection, CICIDS2017 Dataset.

1 Introduction

In recent years, IoT is growing in all areas. This is typically the case for smart agents (IoT-a) that proliferates to provide complex functionalities. Such smart agents rely on communication to cooperate [1]. This is also true for cars that shall embed a cellular connection for emergency call. This is mandatory in Europe from April 1st, 2018 for all passengers cars and light commercial vehicles [12] and also in Russia from January 2017. The presence of a modem enables the emergence of new services requiring data transfer through this cellular connection. In such a context, attacks against IoT-a may lead to loss of functionality or even worse open access to other elements of the network potentially compromising privacy. Cars become potential victims of hackers able to run remote attacks on entire fleet of vehicles. In response to these risks, a first approach is to use network intrusion detection system (IDS) tools such as Snort (open source IDS tool) to analyze network traffic and extract signatures that can be compared to known attack signatures. This type of approach has two limitations: on the one hand, the signature of an attack must already be known and on the other hand, it generates a high false positive rate [4]. An alternative approach is to

use Machine Learning (ML) techniques. These techniques include supervised learning to classify network flows into different categories and unsupervised learning to detect anomalies.

Supervised learning is only possible by having a dataset available, in this case the recording of network frames containing normal traffic and attacks. Cellular or WLAN conecctivity are quite common in IoT and connected cars and we can consider that the intrusion is similar to the attack that would be carried out on a conventional computer network. Common network intrusion detection datasets like KDD-Cup99 [9], NSL-KDD [17] or CICIDS2017 [14] containing different types of attacks against computers or servers can be used for IoT and connected cars.

Traditional machine learning algorithms and neural network-based techniques (also called deep learning) can be used as supervised learning methods. The former group contains algorithms like Decision Tree, Random Forest, or SVM. Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) are examples of the latter group. In IDS researches, many papers cover deeply the traditional approaches while a smaller amount of papers concentrate on neural network techniques.

Our main contributions consist in two parts. Firstly, we propose an approach based on multi-layer perceptron exercised on recent dataset. CICIDS2017 contains intrusion attacks and traffics that are representative of current network usage. All steps are detailed from dataset analysis to tuning of neural network. Secondly, we analyse previous works on the same dataset and compare their performances with our experimental results.

This paper presents in section 2 the related work. Our proposed methodology is described in section 3. Section 4 provides the results of our approach using several metrics and compares it to previous works. A link to the source code is provided so that it can be used to reproduce results and for further improvements. Finally, section 5 concludes this paper and identifies ideas for future work.

2 Related work

Several studies have been conducted on multiple network intrusion detection datasets using various methods. Dhanabal et Shantharajah [3] analyzed the NSL-KDD dataset content and studied several classifiers from the traditional machine learning techniques. They obtained pretty good intrusion detection with accuracy around 99% with SVM and J48 (C4.5) decision tree. Tang et al. [16] used a deep learning approach on the same dataset. They proposed a small neural network with a very limited number of features in input and reached an accuracy of almost 76%.

With the release of CICIDS2017, Sharafaldin et al. [14] provided performances of intrusion detection on their dataset. A significant gap is observed on precision, recall and f_1-measure between all traditional ML algorithms and multi-layer perceptron. K-Nearest Neighbors and Quadratic Discriminant Analysis outperformed MLP by almost 20%. Feature selection, data pre-processing and details about

the classifiers are not described and therefore results cannot be reproduced. In their paper, Jiang et al. [6] focused only on denial of service, 4 classes among the 14 of CICIDS2017 dataset. They proposed new features and compared the results of a neural network with features provided in original CICIDS2017 CSV files. A two-level model was proposed by Ullah et Mahmoud [18]. The first level classes traffic either as normal or attack with a decision tree and the second one identifies the attack type with a random forest after data augmentation based on synthetic minority oversampling technique [2] and edited nearest neighbors. The method has been tested on CICIDS2017 and UNSW-NB15 datasets. Ustebay et al. [19] proposed a two-step approach for CICIDS2017 dataset. Recursive Feature Elimination (RFE) based on Random Forest is used to identify the most useful features that are then injected in a neural network.

Those works either obtain lower performances with neural network or when achieving good results do not consider all types of attacks. We propose a detailed methodology to achieve high performance with neural network without removing any type of attacks.

3 Our methodology and MLP solution

After selecting a dataset, we propose an approach consisting in training a multi-layer perceptron neural network in order to quantify the benefits and potential limitations of using deep learning in IDS. Fig. 1 give the overall framework.

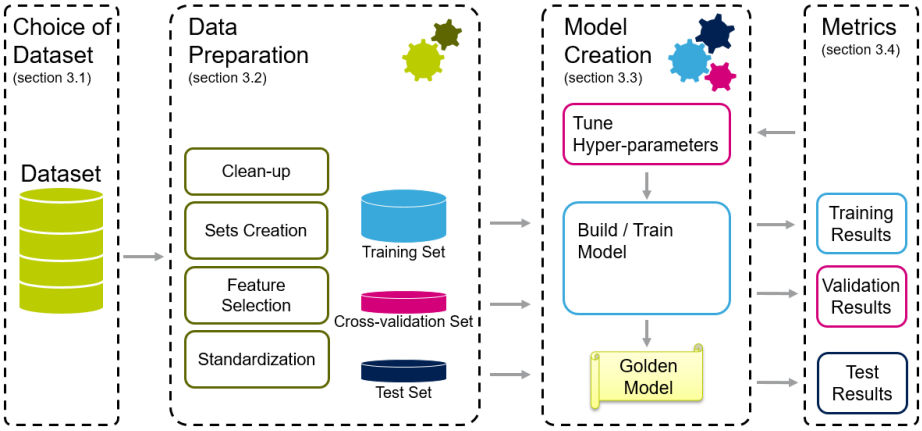

Fig. 1. Overview of the proposed approach.

3.1 Choice of dataset

KDD-Cup99 is a dataset for IDS publicly released in 1999. It is derived from DARPA-98 dataset that contains raw data corresponding to captured network

frames with TCPDUMP. The training set is composed of 24 types of attacks. Raw data have been processed to produce 41 features in KDD-cup99. A first critique of this dataset has been carried out by McHugh [10]. Tavallaee et al. [17] provided a detailed analysis and proposed a derived dataset referred to as NSL-KDD with the goal to solve some of the shortcomings of KDD-Cup99 described McHugh. NSL-KDD dataset has been widely used for IDS since 2009. Despite that attacks has evolved over the time, KDD-Cup99 and NSL-KDD remains a subject of study [13] [15] [20] . We can consider that most of current attacks are not present in a dataset relying on traffic recorded 20 years ago.

In 2015, Moustafa et Slay proposed a new dataset called UNSW-NB15 [11] that has been generated by simulation and representing 31 hours of traffic. As a major difference with NSL-KDD, it contains low footprint and modern attacks retrieved from CVE site[3] that are grouped in nine different families of attacks. UNSW-NB15 is composed of 49 features. A part has been extracted from packet headers while some others have been generated specifically.

Gharib et al. [5] reviewed different datasets and evaluated them with respects to 11 criteria: complete network configuration, complete traffic, labeled dataset, complete interaction, complete capture, available protocols, attack diversity, anonymity, heterogeneity, feature set, metadata. On top of elements already identified in [17], KDD-Cup99 and therefore its NSL-KDD derivative suffer from the lack of some important protocols like HTTPS.

The University of New Brunswick's Canadian Institute for Cybersecurity released the CICIDS2017 dataset based on the framework defined in [14] intending to solve shortcomings of previous datasets. Raw data in the form of PCAP files are provided together with a set of 84 features in CSV files. Network traffic has been recorded over 5 days. The first one only contains normal traffic while 14 types of attack appear for the other days.

Due to the shortcomings of KDD-cup99 and NSL-KDD, we restricted the choice to either UNSW-NB15 or CICIDS2017. Table 1 presents 7 parameters. The first one is year of creation of each dataset. Then the number of features gives us the quantity of input data and labels characterizing each record. The number of records and the distribution between normal traffic and data are important information for dataset selection.

We observe that CICIDS2017 is more recent, include more features and contains more instances than UNSW-NB15. Date of creation and quantity of data are key elements. Firstly, old dataset are no longer representative of current network traffic. As example NSL-KDD does not contain HTTPS exchanges while it represents more than 70% of traffic today. Secondly, more data leads to better learning. For these reasons, CICIDS2017 dataset is selected for the evaluation of our intrusion detection solution.

[3] http://cve.mitre.org/

Table 1. Comparison of UNSW-NB15 and CICIDS2017.

Parameters	**UNSW-NB15**	**CICIDS2017**
Year of creation	2015	2017
Features	49 (+ 2 labels)	84 (+ 1 label)
Attack families	9	14
Duration	31 hours	5 days
# of instances	2,540,044	2,830,743
# of normal instances	2,218,764	2,273,097
# of attack instances	321,283 (12.65%)	557,646 (19.70%)

3.2 Data Preparation

Original dataset Clean-up A series of operations has been carried out to detect the presence of empty lines, redundant features and non numeric values in numeric fields. This systematic verification allowed to drop more than 280,000 instances whose all features were empty. One column corresponding to forward header length appearing twice, one instance has been removed. Most of features are numeric but some values are not a number. Such cases appear in 6 different traffic types: BENIGN, FTP-PATATOR, DoS Hulk, Bot, PortScan and DDoS. As the number of instances containing 'NaN' or 'Infinity' is negligible in each traffic type, these instances have simply been removed. Table 2 provides the number of records for each traffic type and the number of 'NaN'/'Infinity'.

Table 2. CICIDS2017 Traffic Instances.

Traffic	**original instances**	**NaN/Infinity**	**after clean-up**
BENIGN	2,273,097	1,777	2,271,320
Bot	1,966	10	1,956
DDoS	128,027	2	128,025
DoS GoldenEye	10,293	0	10,293
DoS Hulk	231,073	949	230,124
DoS Slowhttptest	5,499	0	5,499
DoS Slowloris	5,796	0	5,796
FTP-PATATOR	7,938	3	7,935
Heartbleed	11	0	11
Infiltration	36	0	36
PortScan	158,930	126	158,804
SSH-PATATOR	2,897	0	2,897
WebAttack BruteForce	1,507	0	1,507
WebAttack SQL Injection	21	0	21
WebAttack XSS	652	0	652

Training set, cross-validation set and test set creation As shown in Table 2, the dataset is highly imbalanced at two different levels. First, the normal traffic (BENIGN) accounts for 80% and second, some attacks (Heartbleed, Infiltration, WebAttack SQL injection) are represented by a very limited number of instances. This is known to be a challenge for supervised learning. The proposal to create dataset for MLP training and testing consists in choosing randomly 50% of each attacks for the training set, 25% for the cross-validation set and 25% for the test set, ensuring that each instance is used only once. Then each set is completed by adding randomly selected instances of normal traffic. This results in a dataset that is balanced in term of normal traffic versus attacks but still imbalanced in term of attack types. The exact composition of the training, cross-validation and test set is provided in Table 3.

Table 3. Dataset Split.

Traffic	**Training set**	**Cross-validation set**	**Test set**
BENIGN	278,274	139,135	139,135
Bot	978	489	489
DDoS	64,012	32,006	32,006
DoS GoldenEye	5,146	2,573	2,573
DoS Hulk	115,062	57,531	57,531
DoS Slowhttptest	2,749	1,374	1,374
DoS Slowloris	2,898	1,449	1,449
FTP-PATATOR	3,967	1,983	1,983
Heartbleed	5	2	2
Infiltration	18	9	9
PortScan	79,402	39,701	39,701
SSH-PATATOR	2,948	1,474	1,474
WebAttack BruteForce	753	376	376
WebAttack SQL Injection	10	5	5
WebAttack XSS	326	163	163
Total	**556,548**	**278,270**	**278,270**

Feature selection Feature selection is one of the fundamental concepts of machine learning that greatly influences the model performance. Irrelevant or partially relevant features can have a negative impact and lead to a decrease in accuracy. An analysis of the dataset revealed 8 features that are not informative as their value is constant whatever the traffic types. Consequently, features 'Bwd PSH Flags', 'Bwd URG Flags', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate' have been dropped. As we don't want the model to learn when attack occurs, the 'Timestamp' feature cannot be considered as informative.

Each instance is characterized by its source and destination port and IP address, its protocol and a flow identifier containing the same information. As

'FlowID' feature is redundant with other features, it has been removed from the dataset.

Our model will be trained with 2 feature sets in order to ease comparison with previous works on the same dataset. Variant-1 contains 73 characteristics: all features except those listed above. In variant-2, source/destination IP address and source port have be dropped, resulting in 70 features.

Standardization Data pre-processing is essential to prepare the dataset for an efficient training. In particular, the neural network can better learn when all features are scaled within the same range. This is especially useful when the inputs are on very different scales. Several normalization techniques exist. In our implementation, Z-score normalization (also called standardization) has been selected as it presents better results than other techniques on the selected dataset. For each feature $\mathcal{F}_j$, the transformation of each value x_i is given by equation 1 where $\mu^{(\mathcal{F}_j)}$ and $\sigma^{(\mathcal{F}_j)}$ are respectively the mean and standard deviation values of feature $\mathcal{F}_j$.

$$x_i^{(\mathcal{F}_j)} = \frac{x_i^{(\mathcal{F}_j)} - \mu^{(\mathcal{F}_j)}}{\sigma^{(\mathcal{F}_j)}} \tag{1}$$

3.3 Model creation

Model description Multi-layer perceptron is a fully connected, feed-forward neural network classifier. Figure 2 shows the architectural design of our model. Inputs correspond to the normalized values of the selected features of the original dataset. Both hidden layers contain 256 nodes. The classifier has 15 outputs for the 14 types of attacks and the benign traffic. Dropout is used as regularization technique for hidden layers to prevent over-adjustment on training data by dropping units randomly in the MLP with a probability *keep_prob*.

Neural network output $\boldsymbol{h}^{(3)}$ is calculated by chaining outputs of the different layers according to equations 2, 3 and 4 in which $\boldsymbol{x}$, $\boldsymbol{W}^{(i)}$ and $\boldsymbol{b}^{(i)}$ are respectively the input vector containing selected features, the matrix of weights and the vector of biases for layer i.

$$\boldsymbol{h}^{(1)} = g_1\left(\boldsymbol{W}^{(1)T}.\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) \tag{2}$$

$$\boldsymbol{h}^{(2)} = g_1\left(\boldsymbol{W}^{(2)T}.\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right) \tag{3}$$

$$\boldsymbol{h}^{(3)} = g_2\left(\boldsymbol{W}^{(3)T}.\boldsymbol{h}^{(2)} + \boldsymbol{b}^{(3)}\right)' \tag{4}$$

Activation functions g_1 and g_2 bring non linearity to neural network. As MLP does not provide an intrinsic normalization of its outputs, scaled exponential linear unit given in equation 5 is used as activation function for hidden layers with values $\lambda = 1.0507$ and $\alpha = 1.6733$ as defined in [8] to take advantage of self normalization. Output layer is a softmax activation function g_2 as defined in

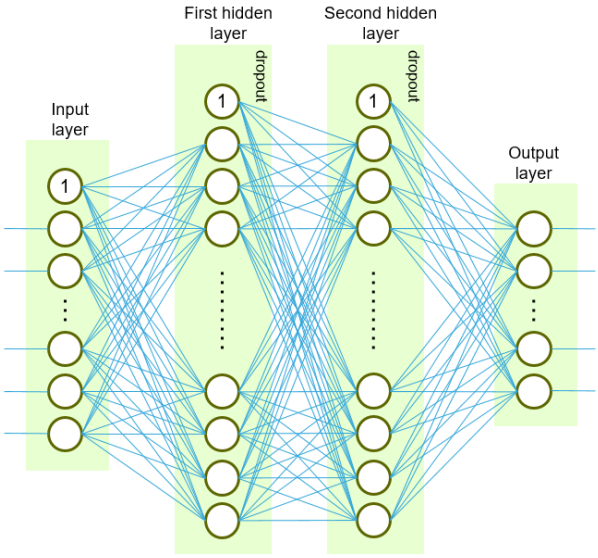

Fig. 2. MLP architectural design.

equation 6 so that each output can be interpreted as the probability of predicting a given class. The predicted label $\hat{y}$ is given by $\hat{y} = argmax\ \boldsymbol{h}^{(3)}$.

$$g_1(z) = \lambda. \begin{cases} \alpha.(e^z - 1) & \text{for } z < 0 \\ z & \text{for } z \geq 0 \end{cases} \tag{5}$$

$$g_2\left(z\right)_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}} \text{ for } j \in [1; 15] \tag{6}$$

Training For model implementation and training, we have used python and Tensorflow as deep learning framework. Training set is divided in mini-batch of 32 instances. MLP learns classification by tuning the weights w between neural network nodes in order to reduce the cross-entropy loss function $\mathcal{L}(w)$ as defined in equation 7 where y the ground truth label and $\hat{y}$ the predicted class. $\mathcal{L}(w)$ is optimized with Adam algorithm. Three parameters α, β_1 and β_2 described in [7] allow to configure this optimizer.

$$\mathcal{L}(w) = -\left[y. \log\left(\hat{y}\right) + \left(1 - y\right). \log\left(1 - \hat{y}\right)\right] \tag{7}$$

Two different models corresponding to the two variants described in section 3.2 have been trained.

3.4 Evaluation Metrics

Several key information can be extracted from a confusion matrix as depicted in Table 4: True Positive (TP) is the number of attacks correctly predicted

as attacks, True Negative (TN) is the number of normal instances classified as normal traffic while False Positive (FP) and False Negative (FN) are respectively the number of normal instances classified as attacks and the number of attacks predicted as normal traffic.

Table 4. Confusion matrix.

		Predicted labels	
		Attacks	Normal
Actual labels	Attacks	TP	FN
	Normal	FP	TN

Different metrics can be derived from the information contained in the confusion matrix. As some existing papers use only a subset of metrics, we propose to cover more metrics in order to enable comparison of our work with future studies on the same dataset.

TN_{rate} given in equation 8 is the percentage of traffic classified as benign over the actual number of benign instances and FP_{rate} in equation 9 is the percentage of benign traffic classified as attacks.

$$TN_{rate} = \frac{TN}{TN + FP} \tag{8}$$

$$FP_{rate} = \frac{FP}{TN + FP} \tag{9}$$

Precision and *Recall* given in equations 10 and 11 correspond respectively to the percentage of attacks correctly detected over the number of instances predicted as attacks and the percentage of attacks correctly detected over the total number of actual attacks.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

Accuracy measures the proportion of total number of correct classifications.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

$f1_{score}$ is the harmonic mean of *Precision* and *Recall*

$$f1_{score} = \frac{2 \times precision \times recall}{precision + recall} \tag{13}$$

Matthews Correlation Coefficient (MCC) is an interesting measure taking into account all elements of the confusion matrix in a correct way for imbalanced

dataset as opposed to the accuracy which may report high value even when the whole minority class is wrongly classified. As intrusion detection datasets are intrinsically imbalanced, this is an key metric for such application. MCC provides a value between -1 and +1. A perfect prediction corresponds to $MCC = 1$. At the opposite, $MCC = -1$ denotes a full disagreement between predictions and actual classes. A random prediction would result to $MCC = 0$.

$$MCC = \frac{TP \times TN + FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \quad (14)$$

4 Experimental results

4.1 Performance evaluation

After training a golden model, the test set has been used to measure the MLP performance as a 15-class classifier. The resulting confusion matrices for the 2 variants are simplified in Tables 5 and 6 to show the estimated class in 3 columns: benign, correct attack and other attacks. In multi-class classification, an attack can be predicted as another attack. This denotes an issue if the correct class is required. In intrusion detection, such a misclassification is acceptable as it is still considered as an attack. Consequently, predictions of all attack types have been merged in one single class to get a binary classifier to calculate metrics in Table 7.

Table 5 shows that MLP performs well on all classes except 'Infiltration'. This specific issue is most likely due to the extremely low number of instances. The neural network didn't succeed in learning from the only 18 examples available in the training set. A significant difference can be observed in Table 6. On the one hand, 1244 normal traffic instances are classified as error, generating false positive detection. On the other hand, more attack instances are classified as normal traffic for almost all classes, meaning that the classifier failed to detect some attacks. In particular, 'infiltration' and 'WebAttack' are not successfully detected. Again, it corresponds to classes with low number of instances. It confirms the well-know fact that the amount of data is a key point for success in machine learning.

The best results are obtained with variant-1 but we can guess the MLP learns the IP address of the machine conducting the attack. By removing IP addresses and source ports in variant-2, the model requires a longer training to reach good performances without being as good as the previous one. This clearly show the importance of these 3 features. Even if the accuracy is above 99%, the recall falls from 99.99% to 99.34% and FP_{rate} goes up from 0.08% to 0.62%. Our classifier provides an MCC value close to 1 and therefore indicate a pretty good performance.

All results can be fully reproduced by using the source code available on Github (https://github.com/ArnaudRosay/mlp4nids) to generate the training, cross-validation and test sets, build the golden model and obtain the values of all the metrics described in this document.

Table 5. Simplified confusion matrix for variant-1.

		Predictions		
		BENIGN	**Correct attack**	**Other attack**
Actual classes	BENIGN	139,017	-	118
	Bot	5	472	12
	DDoS	1	31,994	11
	DoS GoldenEye	0	2,544	29
	DoS Hulk	0	57,529	2
	DoS Slowhttptest	0	1,345	29
	DoS Slowloris	0	1,434	15
	FTP-PATATOR	0	1,969	14
	Heartbleed	0	2	0
	Infiltration	4	3	2
	PortScan	0	39,675	26
	SSH-PATATOR	0	1,459	15
	WebAttack BruteForce	0	349	27
	WebAttack SQL Injection	0	0	5
	WebAttack XSS	0	0	163

Table 6. Simplified confusion matrix for variant-2.

		Predictions		
		BENIGN	**Correct attack**	**Other attack**
Actual classes	BENIGN	138,277	-	858
	Bot	193	296	0
	DDoS	30	31,970	6
	DoS GoldenEye	33	2,537	3
	DoS Hulk	45	57,486	0
	DoS Slowhttptest	18	1,348	8
	DoS Slowloris	7	1,420	22
	FTP-PATATOR	12	1,967	4
	Heartbleed	0	2	0
	Infiltration	7	0	2
	PortScan	43	39,639	19
	SSH-PATATOR	18	1,455	1
	WebAttack BruteForce	340	0	36
	WebAttack SQL Injection	2	0	3
	WebAttack XSS	162	0	1

Table 7. Performance Results.

Metrics	Variant-1 (73 feat.)		Variant-2 (70 feat.)	
	Training	**Test**	**Training**	**Test**
TP	278,256	139,125	276,444	138,225
FP	257	118	1,630	858
FN	18	10	1,830	910
TN	278,017	139,017	276,644	138,277
TN_{rate} (%)	99.91	99.92	99.41	99.34
FP_{rate} (%)	0.09	0.08	0.59	0.62
Recall (%)	99.99	99.99	99.34	99.35
Precision (%)	99.91	99.92	99.42	99.38
Accuracy (%)	99.95	99.95	99.38	99.36
F1-score (%)	99.95	99.95	99.38	99.36
MCC	0.9990	0.9991	0.9876	0.9873

4.2 Comparison with prior results

Table 8 compares results with reference papers. As different metrics are used, it is not possible to fill in all cells of the table. In addition to neural network solutions, traditional machine learning algorithms are also considered for this comparison.

Our model outperforms all the results reported by Sharafaldin et al, both for neural network and traditional machine learning techniques. Jiang et al. achieves a slightly worse performances than our solution. It should be noted that the type of attacks has been limited to application layer DoS (slowloris, slowhttptest, hulk, DDos GoldenEye) and with a very limited amount of instances (4171). Therefore it is difficult to make an exact comparison.

A two-stage approach using decision trees proposed by Ullah and Mahmoud [18] reports metrics with an average of 100%, a closer look at the details reveals that 4 attack types are not perfectly detected by the decision trees. These are the exact same classes for which our classifier encounter some difficulties. The lack of significant digits and averaging method in [18] do not allow a fine-grain analysis but the overall comparison shows that our MLP-based solution reaches the same performance level as traditional machine learning algorithms.

The neural network based solution of Jiang et al. [6] achieved results similar to our proposed method but only cover DoS attacks. As shown in Table 5 and 6, thoses attacks are not the most difficult to detect. We can expect a drop of their overall performance when all other attacks are taken into account.

Ustebay et al. [19] also used multi-layer perceptron but did not achieved high performance. Nevertheless, their results cannot be directly compared with [6] and [18] as the latter use IP addresses of the machines conducting attacks. In a real life scenario, addresses of hackers are not known and cannot be used for intrusion detection. It should be noted that once source IP address is removed, RFE reports the source port as the most important feature. More generally, this proves that usage of source/destination ports and addresses features may

improve intrusion detection on CICIDS2017 but may not be realistic for application to a real network.

Table 8. Performance Comparison.

Paper	**Algorithm**	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	FP_{rate} (%)	**MCC**
our work - variant-1	MLP (45 epochs)	99.95	99.92	99.99	0.08	0.9991
our work - variant-2	MLP (715 epochs)	99.36	99.38	99.35	0.62	0.9873
Sharafaldin et al. [14]	MLP	-	77	83	-	-
	Quadratic Discriminant Analysis	-	97	88	-	-
	K-Nearest Neighbors	-	96	96	-	-
Jiang et al. [6]	MLP	99.23	99.87	99.60	0.77	-
Ullah and Mahmoud [18]	Decision Tree + Random Forest	-	100	100	-	-
Ustebay et al. [19]	MLP	91	-	-	-	-

5 Conclusion

This paper proposed an approach based on multi-layer perceptron for network intrusion detection system covering analysis of the dataset, definition of the MLP and its training. As in any machine learning project, cleaning the dataset and selecting features is an important step before optimization of a neural network. The experiment has shown that MLP is a viable solution reaching top performance. Our approach provides better results than previous implementations with neural networks. It should be noted that IP addresses and destination port are important features, helping to detect intrusion detection. Nevertheless this is not suitable for real world implementation. Without these features, our approach reaches high performance at the cost of a longer training phase. Deep learning techniques are computationally expensive. We only focused on performance but in a constrained system it may be a drawback pushing to use traditional machine learning algorithms. This may evolve with the increase of hardware accelerators for deep learning in many electronic devices.

In future work, neural network in supervised learning may be improved by data augmentation techniques. Number of instances in classes that are not well detected is clearly a point to address to obtain better performances. Beyond this, supervised learning does not allow detection of attacks unseen during the training phase. Unsupervised learning methods may be considered to overcome this limitation.

References

1. Carlier, F., Renault, V.: IoT-a, embedded agents for smart internet of things. application on a display wall. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW). pp. 80–83 (Oct 2016). https://doi.org/10.1109/WIW.2016.034
2. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
3. Dhanabal, L., Shantharajah, D.S.P.: A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. In: International Journal of Advanced Research in Computer and Communication Engineering. vol. 4, pp. 446–452 (2015)
4. Garg, A., Maheshwari, P.: Performance analysis of Snort-based Intrusion Detection System. In: 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS). vol. 01, pp. 1–5 (Jan 2016). https://doi.org/10.1109/ICACCS.2016.7586351
5. Gharib, A., Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: An Evaluation Framework for Intrusion Detection Dataset. In: 2016 International Conference on Information Science and Security (ICISS). pp. 1–6 (Dec 2016). https://doi.org/10.1109/ICISSEC.2016.7885840
6. Jiang, J., Yu, Q., Yu, M., Li, G., Chen, J., Liu, K., Liu, C., Huang, W.: ALDD: A Hybrid Traffic-User Behavior Detection Method for Application Layer DDoS. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). pp. 1565–1569 (Aug 2018). https://doi.org/10.1109/TrustCom/BigDataSE.2018.00225
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. 2015, 3rd International Conference for Learning Representations (2014)
8. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in Neural Information Processing Systems, 2017, pp. 971-980 (2017)
9. Lee, W., Stolfo, S.J., Mok, K.W.: Mining in a data-flow environment: Experience in network intrusion detection. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 114–124. KDD '99, ACM, New York, NY, USA (1999). https://doi.org/10.1145/312129.312212
10. McHugh, J.: Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed by Lincoln Laboratory. ACM Trans. Inf. Syst. Secur. **3**(4), 262–294 (Nov 2000). https://doi.org/10.1145/382912.382923
11. Moustafa, N., Slay, J.: UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS). pp. 1–6 (Nov 2015). https://doi.org/10.1109/MilCIS.2015.7348942, iSSN:
12. Parliament, E.: Regulation (EU) 2015/758 of the European Parliament and of the Council of 29 April 2015 concerning type-approval requirements for the deployment of the eCall in-vehicle system based on the 112 service and amending Directive 2007/46/EC. Official Journal of the European Union (May 2015)
13. Riyaz, B., Ganapathy, S.: An Intelligent Fuzzy Rule based Feature Selection for Effective Intrusion Detection. In: 2018 International Conference on Recent Trends in Advance Computing (ICRTAC). pp. 206–211 (Sep 2018)

14. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,. pp. 108–116. SciTePress (Jan 2018). https://doi.org/10.5220/0006639801080116
15. Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q.: A Deep Learning Approach to Network Intrusion Detection. IEEE Transactions on Emerging Topics in Computational Intelligence **2**(1), 41–50 (Feb 2018). https://doi.org/10.1109/TETCI.2017.2772792
16. Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for Network Intrusion Detection in Software Defined Networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM). pp. 258–263 (Oct 2016). https://doi.org/10.1109/WINCOM.2016.7777224
17. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. pp. 1–6 (Jul 2009). https://doi.org/10.1109/CISDA.2009.5356528
18. Ullah, I., Mahmoud, Q.H.: A Two-Level Hybrid Model for Anomalous Activity Detection in IoT Networks. In: 2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC). pp. 1–6 (Jan 2019). https://doi.org/10.1109/CCNC.2019.8651782
19. Ustebay, S., Turgut, Z., Aydin, M.A.: Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier. In: 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). pp. 71–76 (Dec 2018). https://doi.org/10.1109/IBIGDELFT.2018.8625318
20. Zyad, E., Taha, A., Mohammed, B.: Improve R2L Attack Detection Using Trimmed PCA. In: 2019 International Conference on Advanced Communication Technologies and Networking (CommNet). pp. 1–5 (Apr 2019). https://doi.org/10.1109/COMMNET.2019.8742361