Towards a Hierarchical Deep Learning Approach for Intrusion Detection

François Alin, Amine Chemchem(✉), Florent Nolot, Olivier Flauzac, and Michaël Krajecki

CRESTIC - Centre de Recherche en Sciences et Technologies de l'Information et de la Communication - EA 3804, Reims, France
{francois.alin,mohamed-lamine.chemchem,florent.nolot,olivier.flauzac, michael.krajecki}@univ-reims.fr

Abstract. Nowadays, it is almost impossible to imagine our daily life without Internet. This strong dependence requires an effective and rigorous consideration of all the risks related to computer attacks. However traditional methods of protection are not always effective, and usually very expensive in treatment resources. That is why this paper presents a new hierarchical method based on deep learning algorithms to deal with intrusion detection. This method has proven to be very effective across traditional implementation on four public datasets, and meets all the other requirements of an efficient intrusion detection system.

Keywords: Machine learning · Deep learning · Intrusion detection · Artificial intelligence · Cyber security

1 Introduction

Over the last two decades, many solutions have emerged to protect and secure computer systems. They are complementary but not always sufficient. Thus, antivirus software acts on a host computer and protects against viruses and malicious programs. If this solution is effective for isolated machines and viruses already known, it is not recommended to trust them, especially when connected to the Internet.

To solve the problem of network intrusion, firewalls come to the rescue. A firewall is used to control communications between a local network or host machine and the Internet. It filters traffic in both directions and blocks suspicious traffic according to a network security policy. It is therefore the tool that defines the software and the users that have the permission to connect to the Internet or to access the network [1]. With anti-virus, the firewall increases the security of data in a network. However, this combination remains powerless in front of malicious users with knowledge of all the requirements of security policy. Indeed, once a software or user has the permission to connect to the Internet or a network, there is no guarantee that they will not perform illegal operations. Moreover, many studies have shown that 60% to 70% of attacks come from within systems [2].

Published by Springer Nature Switzerland AG 2020
S. Boumerdassi et al. (Eds.): MLN 2019, LNCS 12081, pp. 15–27, 2020.
https://doi.org/10.1007/978-3-030-45778-5_2

To handle this problem, in addition to antivirus and firewalls, intrusion detection systems (IDS) are used to monitor computer systems for a possible intrusion considered as unauthorized use or misuse of a computer system [3].

IDSs are designed to track intrusions and protect system vulnerabilities. They are very effective at recognizing intrusions for which they are programmed. However, they are less effective if the intruder changes the way he attacks. Whatever the performance of the IDS, they are often limited by the amount of data that the IDS can handle at a time. This limitation does not allow for permanent monitoring and leaves violations for intruders.

This research explores a first implementation of classical machine learning models for intrusion detection. These models take into account the historical IDSs data with their corresponding classes. Then, through a comparative study, we select the best method to use it for the inference in order to detect intrusions in real-time. In addition, we present in this study a hierarchical learning method that is proving to be very effective in comparison with the traditional classification method.

The rest of this paper is organised as follows: next section presents some related works, followed by the proposed strategy in Sect. 3. In Sect. 4 we show the experimental results and give some discussions. Finally, in Sect. 5 we conclude with some perspectives.

2 Related Works

Many recent researches try to handle the intrusion detection problem with the artificial intelligence and machine learning approaches.

The authors of [4] explore the issue of the game theory for modelling the problem of intrusion detection as a game between the intruder and the IDS according to a probabilistic model, the objective of their study is to find a frequency for an IDS verification activities that ensures the best net gain in the worst case. We think that is a good idea, but if the attacker changes his behaviour, the proposed approach will no more be able to intercept him effectively.

In [5] a new agent architecture is proposed. It combines case-based reasoning, reactive behavior and learning. Through this combination, the proposed agent can adapt itself to its environment and identify new intrusions not previously specified in system design. Even if the authors showed that the hybrid agent achieves good results compared to a reactive agent, their experimental study did not include other learning approaches such as support vector machine, K nearest neighbors... In addition, the learning set used in this study is very small, only 1000 records.

The authors in [6] proposed an intrusion detection framework based on an augmented features technique and an SVM algorithm. They validated their method on the NSL-KDD dataset, and stated that their method was superior to other approaches. However, they did not mention which data are used for the test. In addition, the application of features augmentations technique increases

the risk of falling into an over fitting case, especially when processing large data, so we believe this is not an ideal choice for analyzing large network traffic for intrusion detection.

In [7], the authors applied a hybrid model of genetic algorithms with SVM and KPCA to intrusion detection. They used the KDD CUP99 dataset to validate their system. However, this dataset contains several redundancies, so the classifier will probably be skewed in favor of more frequent records. With the same way, the authors of [8] combined decision tree with genetic algorithms and features selection for intrusion detection. They used also the KDD dataset which is not really reliable for validating methods, it would have been more interested to take other datasets to confirm the proposed approach. An interesting multi-level hybrid intrusion detection model is presented in [9], it uses support vector machine and extreme learning machine to improve the efficiency of detecting known and unknown attacks. The authors apply this model on the KDD99 dataset, which has the previously mentioned drawbacks. On the same dataset, a new LSTM: long short term memory model is presented in [10] to deal with four classes of attacks, and the results are satisfactory. In [11] the authors present a text mining approach to detect intrusions, in which they present a new distance measure. However, most of logs features are in numerical format, and taking them as text data will considerably increase the complexity of the calculation in terms of memory capacity and also in terms of learning time. This is the biggest flaw in the last two papers mentioned.

A very interesting survey is presented in [12]. The paper describes the literature review of most of machine learning and data mining methods used for cyber security. However, the methods that are the most effective for cyber applications have not been established by this study, the authors affirm that given the richness and complexity of the methods, it is impossible to make one recommendation for each method, based on the type of attack the system is supposed to detect. In our study we draw inspiration from the methods cited by this paper such as decision trees, support vector machine, k nearest neighbors,... for a comparative study established on the most popular datasets. In addition, we enrich our comparative study with several neural networks models, and with a new proposed hierarchical classification method.

3 The Proposed Strategy

Contrary to the idea in [13], in which the authors present an hierarchical classification of the features for intrusion detection. In our study we propose a hierarchical method that starts by detecting malicious connexion with a binary classification, and then, in a second time, the algorithm tries to find the corresponding attack class by a multi-label classification as shown in Fig. 1. The idea is to detect an abnormal connection very quickly and launch a warning to the admin, then try to classify this connection in the corresponding attack class.

For this, we will design a hierarchical approach to the machine learning methods, and compare their performances with the classical algorithms of classification.

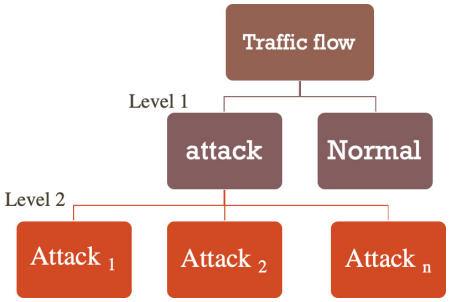

Fig. 1. The proposed hierarchical classification method

In the context of data mining and machine learning: classification is done using a model that is built on historical data. The goal of predictive classification is to accurately predict the target class for each record in new data. A classification task begins with build training data for which the target values (or class assignments) are known. Many classification algorithms use different techniques for finding relations between the predictor attribute's values and the target attribute's values in the build data [14,15]. In the following subsections, a summarised overview of the implemented machine learning algorithms is reported.

3.1 Naïve Bayes Classifier

The naïve Bayes algorithm is based on Bayesian probability theory following assumptions of naive independence [16]. It is one of the most basic classification techniques with various applications, such as email spam detection, personal email sorting, and document categorization.

Even though it is often outperformed by other techniques. The main advantage of the naïve Bayes remains that it is less computationally intensive (in both CPU and memory), and it requires a small amount of training data. Moreover, the training time with Naive Bayes is significantly smaller as opposed to alternative methods [17].

3.2 K-Neighbors Approach

Nearest Neighbors is one of the simplest, and rather trivial classifiers is the rote classifier, which memorizes all training data and performs classification only if the attributes of the test object match one of the training examples exactly [18]. A more known variation, the k-nearest neighbor (kNN) classification [19], finds a group of k objects in the training set that are closest to the test object and bases the assignment of a label on the predominance of a particular class in this neighborhood. There are three key elements of this approach: a set of labeled objects, a distance or similarity metric to compute distance between objects, and the value of parameter k, which represents the number of nearest neighbors.

To classify an unlabeled object, the distance of this object to the labeled objects is computed, its k-nearest neighbours are identified, and the class labels of these nearest neighbours are then used to determine the class label of the object.

Given a training set DR and a test object $z = (x', y')$ the algorithm computes the distance (or similarity) between z and all the training objects $(x, y) \in DR$ to determine its nearest-neighbor list: D_z. (x_i is the training data of $object_i$, while y_i is its class. Likewise, x' the data of the test object and y' is its class.) Once the nearest-neighbors list is obtained, the test object is classified based on the majority class of its nearest neighbors:

$$Majority\_Voting\_y' = argmax_v \sum_{x_i, y_i \in D_z} I(v = y_i).$$

where v is a class label, y_i is the class label for the i^{th} nearest neighbors, and $I()$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

3.3 Support Vector Machine

Support vector machines (SVM) have exhibited superb performance in binary classification tasks. Intuitively, SVM aims at searching for a hyperplane that separates the two classes of data with the largest margin (the margin is the distance between the hyperplane and the point closest to it) [20,21].

Most discriminative classifiers, including SVMs, are essentially two-class classifiers. A standard method of dealing with multi-class problems is to create an ensemble of yes/no binary classifiers, one for each label. This method is called "one-vs-others" [22].

3.4 Random Forests

Random Forests are a part of ensemble learning. Ensemble learning [23] deals with methods which employ multiple learners to solve a problem. The capacity of working with several learners in the same time achieve better results than a single learner. Random forest works by creating various decision trees in the training phase and output class labels those have the majority vote [24]. They achieve high classification accuracy and can handle outliers and noise in the data. Random Forest is implemented in this work because it is less susceptible to over-fitting and it has previously shown good classification results.

3.5 Multilayer Perceptron Neural Networks

The basic unit in a neural network is called "neuron" or "unit". Each neuron receives a set of inputs, which are denoted by the vector $\overline{X}_i$ [25]. In addition, each neuron is associated with a set of weights A, which are used for computing a function f of its inputs. A typical function that is basically used in the neural

network is the linear function, it is defined as follows: $p_i = A.\overline{X}_i$. We assume that the class label is denoted by y_i. The goal of this approach is to learn the set of weights A with the use of the training set. The idea is to start off with random weights, and gradually update them when a mistake is done by applying the current function on the training example. The magnitude of the update is regulated by a learning rate μ. This forms the core idea of the perceptron algorithm.

Algorithm 1. Perceptron Algorithm [26]

inputs: Learning Rate: μ
Training rules $(\overline{X}_i, y_i) \forall i \in \{1...n\}$.
Initialize weight vectors in A to 0 or small random numbers.
Repeat
– Apply each training rule to the neural network
– **if** $((A.\overline{X}_i)$ does not matches $y_i)$ **then**
update weigts A based on learning rate μ.
until weights in A converge.

3.6 Convolutional Neural Network Classifier

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers, and then followed by one or more fully connected layers as in a standard multilayer neural network. The neurons of a convolutional layer are grouped in feature maps sharing the same weights, so the entire procedure becomes equivalent to convolution [27]. Convolutional layers are usually followed by a non-linear activation-layer, in order to capture more complex properties of the input data. The pooling layers are usually used for subsampling the preceding layer, by aggregating small rectangular values subsets. Maximum or average pooling is often applied by replacing the input values with the maximum or the average value, respectively. Finally, one or more dense layers are put in place, each followed by an activation-layer, which produce the classification result.

The training of CNNs is performed similarly to that of classical Multilayer Perceptron Networks, by minimizing a loss function using gradient descent-based methods and back-propagation of the error.

In this study, our best CNN model is reached after trying many architectures. It is inspired by the contribution of [28] in sentiments detection. Since its model demonstrated well performant results, we adapted it in our study for intrusion detection. It is mainly composed of four hidden layers in addition to the input and the output layer. In the following, we show a summarization of the architecture of our best CNN model:

– The input layer is a convolution of one dimension with a number of neurones equals to the number of the dataset features.

- The second layer is a max pooling 1D with a pool size equals to 4.
- The third layer is a flatten with 512 neurones.
- The fourth is a dense layer with 512 neurones with 'relu' as activation function.
- The output layer is a dense layer with the 'sigmoid' function, and a number of neurones equals to the number of classes.

4 Experimentation Results and Discussion

In this section, we implement various machine learning models to classify the different sets of network traffic of the four well known benchmarks, which are summarised in Table 1:

Table 1. The datasets characteristics

Dataset	Rows	Features	Classes
KDD 99	4,898,430	42	23
NSL-KDD	125,973	42	23
UNSW-NB15	2,540,044	49	10
CIC-IDS 2017	2,832,678	79	14

4.1 Performance Evaluation

For the performance evaluation, we calculate the accuracy with the F-score of each approach. The F-score also called F-measure is based on the two primary metrics: precision and recall. Given a subject and a gold standard, precision is the proportion of cases that the subject classified as positive that were positive in the gold standard. It is equivalent to positive predictive value. Recall is the proportion of positive cases in the gold standard that were classified as positive by the subject. It is equivalent to sensitivity. The two metrics are often combined as their harmonic mean, the formula can be formulated as follows:

$$F = \frac{(1+\beta^2) \times recall \times precision}{(\beta^2 \times precision) + recall}$$

$$Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}$$

Where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives. The F-measure can be used to balance the contribution of false negatives by weighting recall through a parameter $\beta \geq 0$. In our case β is set to 1, F1-score is than equal to:

$$F1\_score = \frac{2 \times recall \times precision}{precision + recall}$$

4.2 Environment and Materials

We use in this implementation, Python language, Tensorflow tool, and Keras library. All the algorithms are executed and compared using the **NVIDIA DGX-1**[1]. The DGX1 is an Nvidia server which specializes in using GPGPU to accelerate deep learning applications. The server features 8 GPUs based on the Volta daughter cards with HBM 2 memory, connected by an NVLink mesh network.

4.3 Multi-class Classification Results

In this paper, first, we classify the datasets as they are labelled, without any modification. The obtained results are shown in Table 2, columns of Multi-label classification.

We notice from these results that most of the approaches succeed in obtaining good training and test scores. However, the best of these models does not exceed 71% accuracy on the test benchmark.

Considering the delicacy of the domain, and the dangerousness that can generate an IDS which classifies network traffic as normal when it is an attack. We propose a hierarchical classification strategy to achieve greater accuracy.

4.4 Hierarchical Classification Strategy

Given the main objective of an intrusion detection system, which is to detect potential attacks, we have decided in this strategy to adopt an hierarchical classification.

First, we start with a binary classification, merging all attack classes into one large class and labelling it 'attack', and on the other hand keeping the 'normal' class without any modification.

Then, after separating normal network traffic to that which represents a potential attack, a multi-class classification can be applied within the "attack" class to know what type of attack it is.

Binary Classification (Level 1: Normal/Attack). We start the hierarchical classification strategy by the detection of an abnormal connexion. This is reached by the binary classification task. For this, we have merged all the connections which have a different label from 'normal' into a single class that we have labelled 'attack'. The obtained results are show in Table 2, columns of hierarchical (level 1).

[1] https://www.nvidia.fr/data-center/dgx-1/.

Table 2. Classifications reports

Data set	ML approach	Multi-label classification		Hierarchical (level 1)		Hierarchical (level 2)	
		Training	Test score	Training	Test score	Training	Test score
KDD 99	Naive Bayes classifier	0.91	0.30	0.98	0.79	0.84	0.46
	Decision tree classifier	0.99	0.53	0.99	0.69	0.99	0.58
	K neighbors classifier	0.99	0.50	0.99	0.81	0.99	0.58
	Logistic regression classifier	0.99	0.31	0.99	0.83	0.99	0.55
	Support vector classifier	0.99	0.43	0.99	**0.84**	0.99	0.58
	Ada_Boost classifier	0.66	0.34	0.92	0.83	0.92	0.40
	Random forest classifier	0.99	0.53	0.99	0.83	0.99	0.56
	Multilayer perceptron	0.99	0.33	0.99	0.83	0.99	**0.61**
	Best NN model	0.99	0.35	0.99	**0.84**	0.99	**0.61**
NSL-KDD	Naive Bayes classifier	0.87	0.56	0.90	0.77	0.83	0.44
	Decision tree classifier	0.99	0.59	0.99	0.80	0.99	0.42
	K neighbors classifier	0.99	0.69	0.99	0.78	0.99	**0.75**
	Logistic regression classifier	0.97	0.71	0.95	0.78	0.99	**0.75**
	Support vector classifier	0.99	0.70	0.99	**0.82**	0.99	0.73
	Ada_Boost classifier	0.84	0.62	0.98	0.75	0.76	0.23
	Random forest classifier	0.99	0.67	0.99	0.79	0.99	0.55
	Multilayer perceptron	0.99	0.70	0.99	0.81	0.99	0.73
	Best NN model	0.98	0.73	0.99	0.81	0.99	0.73
UNSW-NB	Naive Bayes classifier	0.64	0.56	0.80	0.72	0.60	0.59
	Decision tree classifier	0.80	0.32	0.94	0.70	0.78	0.22
	K neighbors classifier	0.76	0.70	0.93	0.83	0.74	0.74
	Logistic regression classifier	0.75	0.62	0.93	0.78	0.73	0.53
	Support vector classifier	0.78	0.53	0.93	0.80	0.77	0.73
	Ada_Boost classifier	0.59	0.50	0.94	0.77	0.59	0.12
	Random forest classifier	0.81	0.43	0.95	0.76	0.79	0.22
	Multilayer perceptron	0.79	0.71	0.94	0.81	0.78	0.67
	Best NN model	0.74	0.71	0.94	**0.86**	0.78	**0.79**
CIC-IDS	Naive Bayes classifier	0.69	0.63	0.65	0.64	0.81	0.43
	Decision tree classifier	0.99	0.15	0.99	0.30	0.99	0.10
	K neighbors classifier	0.99	0.73	0.99	0.73	0.99	0.91
	Logistic regression classifier	0.94	0.88	0.95	0.90	0.98	0.94
	Support vector classifier	0.96	0.72	0.97	0.67	0.99	0.86
	Ada_Boost classifier	0.65	0.35	0.99	0.53	0.50	0.50
	Random forest classifier	0.99	0.30	0.99	0.32	0.99	0.45
	Multilayer perceptron	0.98	0.87	0.99	0.91	0.99	0.91
	Best NN model	0.96	0.88	0.97	**0.92**	0.99	**0.95**

Classification of Attack Types (Level 2: Multi-class Classification). After applying a binary classification to detect abnormal connections. We implement a multi-class classification approach on the 'attack' class, to obtain more details on the type of attack. All the well known machine learning approaches are implemented and compared in this way. The results are noted in Table 2,

columns of hierarchical classification (level2). We can imagine a third level, if we have subclasses in an attack class.

4.5 Discussion

From these results, we note that the proposed hierarchical approach has considerably improved the effectiveness of the classical classification approach on all the benchmarks studied.

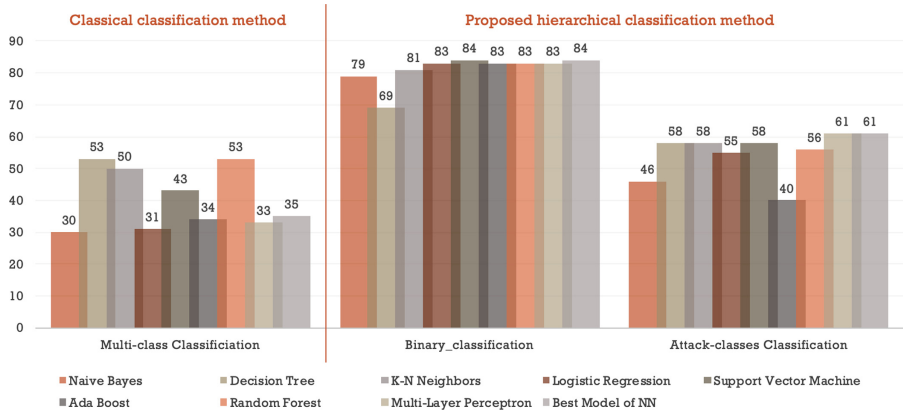

Fig. 2. Classification methods comparison on KDD Dataset

For instance, on the KDD99 dataset, the proposed hierarchical classification approach surpassed the traditional multi-class approach. Like demonstrated on Fig. 2. While the best approach obtained an accuracy of 53% in a multi-class classification, the proposed approach allows to detect an abnormal connection with a rate of 84%, and to predict the attack type with a success rate of 61%.

Also on the NSLKDD dataset, the proposed hierarchical classification approach surpassed the traditional multi-class approach. We can note on Fig. 3 that, while the best approach obtained an accuracy of 73% in a multi-class classification, the proposed approach allows to detect an abnormal connection with a rate of 82%, and to predict the attack type with a success rate of 75%.

In Fig. 4, we note that the proposed approach of hierarchical classification has increased the accuracy rate. While, the best approach obtained an accuracy of 71% in a multi-class classification, the proposed approach allows to detect an abnormal connection with a rate of 86%, and to predict the attack type with a success rate of 79%.

We valid our hypothesis also on the CIC-IDS 2017 dataset. In Fig. 5, we note that the proposed approach of hierarchical classification has increased the accuracy rate. While, the best approach obtained an accuracy of 88% in a multi-class classification, the proposed approach allows to detect an abnormal connection with a rate of 92%, and to predict the attack type with a success rate of 95%.

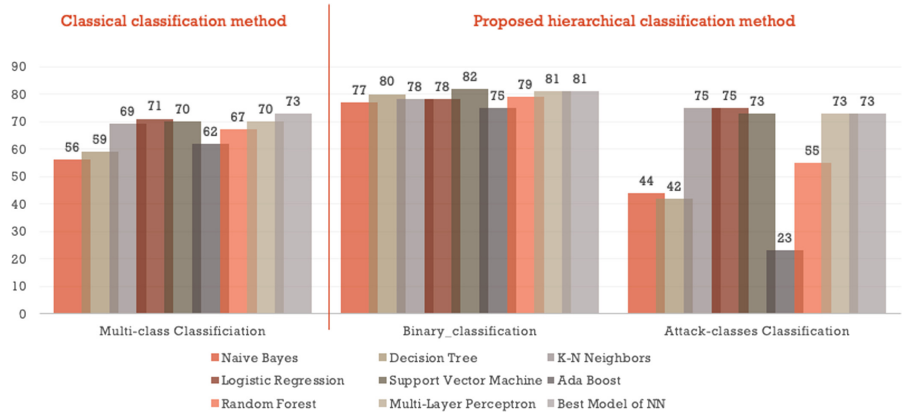

Fig. 3. Classification methods comparison on NSLKDD Dataset

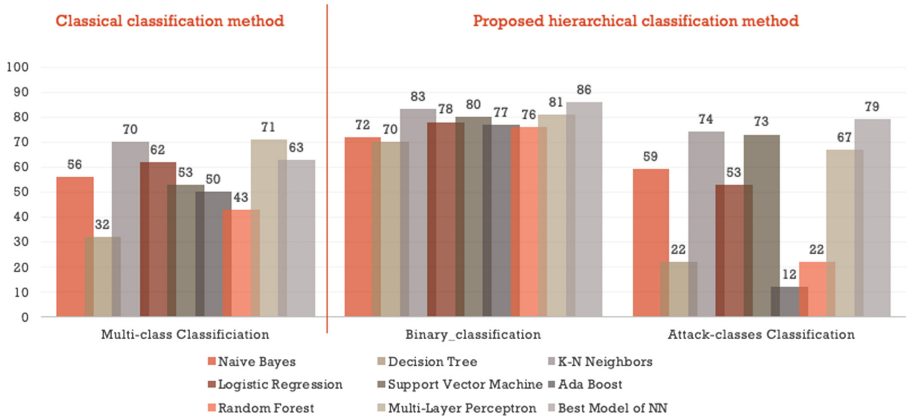

Fig. 4. Classification methods comparison on UNSW15 Dataset

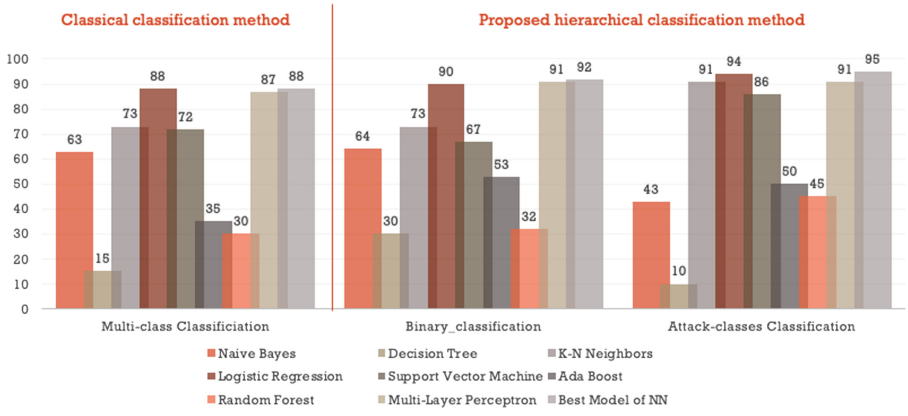

Fig. 5. Classification methods comparison on CIC_IDS 2017 Dataset

According to the results of our comparative study, in general we can validate the hypothesis that to achieve effective intrusion detection, we must start with a binary classification (attack/normal) using our best neural network model, followed by the application of the KNN algorithm or the best neural network model to find out what type of attack is involved.

5 Conclusion

The study presented here leads us to make two conclusions. It appears first that the learning model best suited to the intrusion detection problem is that based on convolutional neural networks. Moreover, by comparing different learning strategies, the approach based on a hierarchical detection of the attacks (starting with a first level of binary classification discriminating only the compliant traffic of the nonconforming traffic) presents the best performances, well in front of the methods of multi-label classification. The system thus obtained has an intrusion detection rate of 95%. These results allow us to consider the implementation of a real-time intrusion detection system based on our CNN model and binary classification. This will require larger datasets and more powerful training infrastructure solutions to further improve the detection rate. Finally, one of the challenges of intrusion detection remains zero-day attack detection. It turns out that the method used to train our neural network gives him the ability to identify as invalid data he has never met during his training. The next task will be to develop this capacity and especially to measure its effectiveness.

References

1. da Costa Júnior, E.P., da Silva, C.E., Pinheiro, M., Sampaio, S.: A new approach to deploy a self-adaptive distributed firewall. J. Internet Serv. Appl. **9**(1), 1–21 (2018). https://doi.org/10.1186/s13174-018-0083-6
2. Stolfo, S.J., Salem, M.B., Keromytis, A.D.: Fog computing: mitigating insider data theft attacks in the cloud. In: 2012 IEEE Symposium on Security Privacy Workshops. IEEE (2012)
3. Carter, E.: CCSP Self-study: Cisco Secure Intrusion Detection System (CSIDS). Cisco Press, Indianapolis (2004)
4. Ouharoun, M., Adi, K., Pelc, A.: Modélisation de détection d'intrusion par des jeux probabilistes. Diss. Université du Québec en Outaouais (2010)
5. Leite, A., Girardi, R.: A hybrid and learning agent architecture for network intrusion detection. J. Syst. Softw. **130**, 59–80 (2017)
6. Wang, H., Jie, G., Wang, S.: An effective intrusion detection framework based on SVM with feature augmentation. Knowl.-Based Syst. **136**, 130–139 (2017)
7. Kuang, F., Weihong, X., Zhang, S.: A novel hybrid KPCA and SVM with GA model for intrusion detection. Appl. Soft Comput. **18**, 178–184 (2014)
8. Stein, G., et al.: Decision tree classifier for network intrusion detection with GA-based feature selection. In: Proceedings of the 43rd Annual Southeast Regional Conference-Volume 2. ACM (2005)

9. Al-Yaseen, W.L., Othman, Z.A., Nazri, M.Z.A.: Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. Expert Syst. Appl. **67**, 296–303 (2017)
10. Kim, J., et al.: Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon). IEEE (2016)
11. RajeshKumar, G., Mangathayaru, N., Narsimha, G.: Intrusion detection a text mining based approach. arXiv preprint arXiv:1603.03837 (2016)
12. Buczak, A.L., Guven, E.: A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutorials **18**(2), 1153–1176 (2016)
13. Wang, W., et al.: HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access **6**, 1792–1806 (2018)
14. Chemchem, A., Alin, F., Krajecki, M.: Improving the cognitive agent intelligence by deep knowledge classification. Int. J. Comput. Intell. Appl. **18**, 1950005 (2019)
15. Chemchem, A., Alin, F., Krajecki, M.: Combining SMOTE sampling and machine learning for forecasting wheat yields in France. In: 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). IEEE (2019)
16. Jain, A., Mandowara, J.: Text classification by combining text classifiers to improve the efficiency of classification. Int. J. Comput. Appl. (2250–1797) **6**(2) (2016)
17. Huang, J., Lu, J., Ling, C.X.: Comparing naive Bayes, decision trees, and SVM with AUC and accuracy. In: Third IEEE International Conference on Data Mining (ICDM), p. 553 (2003)
18. Adeniyi, D., Wei, Z., Yongquan, Y.: Automated web usage data mining and recommendation system using K-nearest neighbor (KNN) classification method. Appl. Comput. Inform. **12**(1), 90–108 (2016)
19. Wu, X.: Top 10 algorithms in data mining. Knowl. Inf. Syst. **14**(1), 1–37 (2008)
20. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Mach. Learn. **46**(1), 389–422 (2002)
21. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, Heidelberg (2013)
22. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multiclass support vector machines. IEEE Trans. Neural Netw. **13**(2), 415–425 (2002)
23. Zhang, C., Ma, Y. (eds.): Ensemble Machine Learning: Methods and Applications. Springer, Heidelberg (2012). https://doi.org/10.1007/978-1-4419-9326-7
24. Dietterich, T.G.: Machine learning: four current directions. AI Mag. **18**(4), 97–136 (1997)
25. Liu, B., Zhang, L.: A survey of opinion mining and sentiment analysis. In: Aggarwal, C., Zhai, C. (eds.) Mining Text Data, pp. 415–463. Springer, Heidelberg (2012). https://doi.org/10.1007/978-1-4614-3223-4_13
26. Aggarwal, C.C.: Data Classification: Algorithms and Applications. CRC Press, Boca Raton (2014)
27. Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., Mougiakakou, S.: Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. IEEE Trans. Med. Imaging **35**(5), 1207–1216 (2016)
28. Kim, Y.: Convolutional neural networks for sentence classification, CoRR abs/1408.5882 (2014)