Securing Fog-to-Things Environment Using Intrusion Detection System Based On Ensemble Learning

Poulmanogo Illy*, Georges Kaddoum*, Christian Miranda Moreira*, Kuljeet Kaur*, and Sahil Garg*
* Electrical Engineering Department, École de Technologie Supérieure, Montréal, Canada.
Email: poulmanogo.illy.1@ens.etsmtl.ca

***Abstract*—The growing interest in the Internet of Things (IoT) applications is associated with an augmented volume of security threats. In this vein, the Intrusion detection systems (IDS) have emerged as a viable solution for the detection and prevention of malicious activities. Unlike the signature-based detection approaches, machine learning-based solutions are a promising means for detecting unknown attacks. However, the machine learning models need to be accurate enough to reduce the number of false alarms. More importantly, they need to be trained and evaluated on realistic datasets such that their efficacy can be validated on real-time deployments. Many solutions proposed in the literature are reported to have high accuracy but are ineffective in real applications due to the non-representativity of the dataset used for training and evaluation of the underlying models. On the other hand, some of the existing solutions overcome these challenges but yield low accuracy which hampers their implementation for commercial tools. These solutions are majorly based on single learners and are therefore directly affected by the intrinsic limitations of each learning algorithm. The novelty of this paper is to use the most realistic dataset available for intrusion detection called NSL-KDD, and combine multiple learners to build ensemble learners that increase the accuracy of the detection. Furthermore, a deployment architecture in a fog-to-things environment that employs two levels of classifications is proposed. In such architecture, the first level performs an anomaly detection which reduces the latency of the classification substantially, while the second level, executes attack classifications, enabling precise prevention measures. Finally, the experimental results demonstrate the effectiveness of the proposed IDS in comparison with the other state-of-the-arts on the NSL-KDD dataset.**

***Index Terms*—Intrusion detection system, Machine learning, Ensemble learner, NSL-KDD, Fog-to-Things.**

I. Introduction

The Internet of Things (IoT) paradigm offers prodigious opportunities to the industries [1]. This technology is expected to be further active with the imminent Fifth-Generation (5G) mobile communications system [2]. However, the massive deployment of IoT networks and their usage in critical domains such as smart housing, smart transportation, and e-health, results in the generation of abundant sensitive data on real-time basis. Due to this reason, these networks are deemed to be one of the most vulnerable sites for different security attacks and risks. To tackle this issue, many research studies have been focused on the first security layer, *i.e.*, the prevention layer. Thus, stronger authentication, authorization, and cryptography techniques have been proposed in the literature. However, despite the deployment of such strong security measures, a system can still be compromised by an enduring adversary using advanced techniques or high computational resources. Therefore, under any prevention layer, there must be an intrusion detection layer. This is the motivation for the development of intrusion detection systems (IDS). Majority of intrusion detection solutions deployed commercially implement signature-based approaches. Unlike the signature-based IDS, the machine learning-based IDS are capable of detecting even unknown attacks. Nevertheless, the fundamental challenge in this direction involves the designing of an efficient machine learning based IDS that performs well on real-time data.

The majority of machine learning-based IDS proposed in the literature have been built on KDDCUP'99 dataset [3]. The corresponding evaluations results indicate impressive performances in terms of high accuracy (99%) and negligible false positive rate (1%) [4]–[6]. Despite their good performances, the existing solutions are still not employed widely in commercial tools, relatively to the signature-based approaches. To understand this situation, the work in [7] conducted a statistical analysis on KDDCUP'99 dataset and found some important issues, mainly induced by a huge number of redundant records. To address these problems, the authors provided a new dataset named NSL-KDD (comprising of KDDTrain+, KDDTest+, and KDDTest-21) that is more realistic and challenging enough to compare different solutions. Based on these refinements, many machine learning methods have been proposed and compared in the literature. In [7], the authors implemented five different methods, namely Naive Bayes/Decision-Tree, Random Tree, Decision Tree J48, Random Forest, and Multi-Layer Perceptron on the refined datasets that led to overall accuracy of 82.02% on KDDTest+ and 66.16% on KDDTest-21 datasets respectively. To improve the detection, the work in [8] employed different feature selection metrics at the pre-processing phase for dimensionality reduction on NSL-KDD dataset. Overall, accuracy of 82.32% and 66.77% were achieved on KDDTest+ and KDDTest-21 respectively, which is quite a small performance improvement. Ibrahim *et al.* in [9] employed a Self-Organization Map (SOM) Artificial Neural Network (ANN) which is an unsupervised learning

This research was supported by the NSERC B-CITI CRDPJ 501617-16 grant.

neural network used for classifying the system's input data into normal and abnormal/intrusive instances. In this work, the authors performed only a binary classification, and the model resulted to merely 75.49% detection rate on KDDTest+, after training it rigorously for an hour. Ingre *et al.* proposed in [10] a deep learning method using a tansig transfer function, Levenberg-Marquardt (LM) and BFGS quasi-Newton Back-propagation algorithm for updating weights and biases. Using the same dataset, the multi-class classification gave 81.2% of accuracy. Additionally, the authors also performed a binary classification that led to 79.9% accuracy. In another work [11], Yin *et al.* used ANN and applied a Recurrent Neural Network (RNN) after a data pre-processing (normalization) phase. The authors built a multi-class classification and a binary classification models. These models were trained on KDDTrain+ with different numbers of hidden nodes, and different values of learning rates. However, the maximum accuracy achieved was 83.28%, for the binary classification on KDDTest+.

Motivation: To summarize, most of these works were founded on two conventional approaches. Either they tried different learning algorithms and chose the one that performed the best on the test dataset, or they focused on tuning the hyperparameters of one model until they got satisfactory accuracy. However, there are machine learning problems in which even the best learner is not accurate enough. Therefore, many solutions have investigated the combination of learners in different application domains such as image processing, financial forecasting, and weather forecasting [12]–[14]. These ensemble learners have resulted to important improvements over the single learners. The work in [15] analyzed the data pre-processing influence on intrusion detection using NSL-KDD dataset. Among the machine learning methods employed to build and evaluate models on different pre-processed data, ensemble methods were used. The best performance recorded, 84.84% accuracy for binary classification, was produced by an ensemble model. The base learners in this ensemble complemented each other and reached a higher accuracy. Thus, motivated by this result, the proposed work considers that investigating more diverse base learners, and suitably combining them, will result to ensemble models that perform better on NSL-KDD dataset.

Contributions: The contributions of this work are three fold:

- Firstly, the proposed IDS employs diverse base learners using different algorithms and implements different ensemble methods. On NSL-KDD dataset, the ensemble learners attain higher accuracy than the single learners.
- Secondly, a deployment architecture in a fog-to-things environment is proposed. It reduces considerably the latency of the IDS by performing anomaly detection at a first stage. In a second stage, it provides attack classification that helps to determine precise prevention measures.
- Thirdly, analysis and comparison of the proposed IDS with different solutions in the literature using NSL-KDD dataset has been carried out.

Organization: The rest of the paper is organized as follows. Section II describes the combination of multiple learners to build ensemble learners. Section III presents the proposed IDS solution in fog-to-things environment. A detailed description of the NSL-KDD dataset is done in Section IV. Following this, Section V presents the experimental evaluations of the proposed models and some comparisons with the current state-of-the-art. Finally, Section VI concludes the work.

II. Combination of Multiple Learners

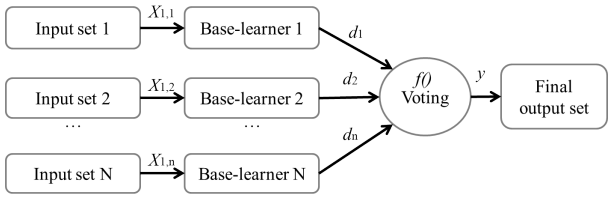
In this section, the concept of combining multiple learners for achieving higher accuracy is presented.

The machine learning methods use a set of assumptions over the data distribution in order to build the models. In the classification task, sometimes the problem to solve has patterns that differ from the assumptions made in every single machine learning method. Thus, each method is capable to learn only some parts of these patterns and are unable to achieve satisfactory accuracy alone. In this case, by suitably combining different learners, they can complement each other and attain higher accuracy. This idea is reinforced by the "*No Free Lunch Theorem*" which advocates the following: "There is no single learning algorithm that in any domain always induces the most accurate learner". However, in order to build an ensemble model that improves the classification accuracy, two essential conditions must be met [16].

The first condition is to have base learners that are reasonably accurate in their domain of expertise and differ in their decisions. This condition can be achieved by using different algorithms, hyperparameters, input representations, or training sets.

The second condition is how to combine the outputs of multiple base learners to generate the final result. There are two main combination methods used, namely multiexpert and multistage combinations. In the multiexpert combination, the base learners can work in parallel, and all outputs (in the global approach) or selected outputs (in local approach) are then used to generate the final result. Examples of implementation of this combination method are voting and bagging methods. Fig. 1 shows the scheme of this combination method. Usually, the operation performed on the outputs of the base learners is a simple function such as sum, weighted sum, median, minimum, maximum or product. Table I defines these functions. The multistage combination uses a serial approach where each supplementary learner works on the limitations of the previous learners. They are trained or tested only on the instances where the previous base learners were not capable to reach a satisfactory accuracy. An example of implementation of this combination is the boosting method. Fig. 2 illustrates the multistage combination.

This work used these combination methods to build more accurate classifiers on the NSL-KDD dataset.



The inputs observed by learner i is represented by $X_{1,i}$. The output d_i is the decision produced by base learner i. The final decision $y = f(d_1, d_2, ..., d_n|\phi)$, where $f(.)$ is the combining function with ϕ donating its parameters. This illustration is for the case of a single output. When there are K outputs, for each learner there are $d_{ij}(X_{1,i})$, $i = 1, ..., N$, $j = 1, ..., K$, and combining them, we also generate K values, $y_j, j = 1, ..., K$. The maximum y_j is chosen as the predicted class (Choose C_j if $y_j = \max_{k=1}^{K} y_k$).

Fig. 1: Multiexpert combination scheme.

TABLE I: Classifier combination rules.

Rule	Fusion function $f(.)$
Sum	$y_i = \frac{1}{N}\sum_{j=1}^{N} d_{ij}$
Weighted sum	$y_i = \sum_j w_j d_{ij}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i =$median$_j d_{ij}$
Minimum	$y_i =$min$_j d_{ij}$
Maximum	$y_i =$max$_j d_{ij}$
Product	$y_i = \prod_j d_{ij}$

In these definitions, d_{ij} represents the output produced by base learner j for the instance i. N is the number of base learners used in the Ensemble. For the Weighted sum, w_i is the value to weight d_{ij}. y_i represents the output of the Ensemble leaner for the instance i.

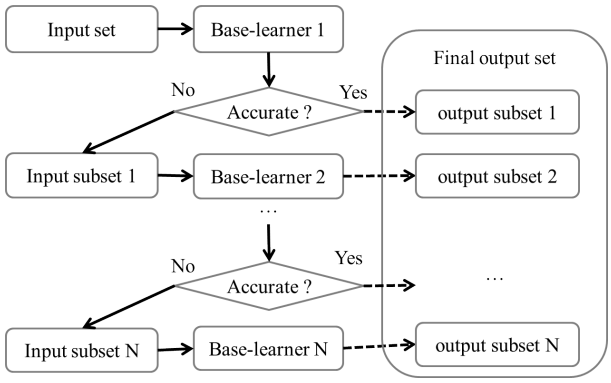


Fig. 2: Multistage combination scheme.

III. Proposed intrusion detection system

A. The Proposed Method

Different steps have been involved in the conception of our ensemble classifiers. At first, we investigated data pre-processing methods including dimensionality reduction and data normalization. Then, diverse base learners using different algorithms have been implemented. We used a Decision Tree, a parametric, a non-parametric, and a deep neural network methods. For each base learner, we tuned the hyperparameters such as the maximum depth for the decision tree algorithm or the number of hidden nodes for the deep learning algorithm. We did not apply advanced tuning methods as we wanted them to be just reasonably accurate, because they are not combined according to their individual accuracy, but for their diversity. After that, the base learners were evaluated in order to analyze their difference in the prediction of each class. Fig. 3 describes the procedure for generating the base classifiers.

Based on their diversities, we defined combinations of different learners to build multiexpert ensembles that employ voting functions. We also implemented learners using Random Forest classifier and bagging classifier. Furthermore, Multistage ensembles that use a boosting algorithm were also employed. For each ensemble method, we tuned some parameters (e.g., number of estimators) in order to have models with better classification accuracy. These ensemble methods have been evaluated and have resulted to significant accuracy improvement over the base learners. The obtained accuracy is also better than most of the solutions proposed in the literature. Fig. 4 describes the creation of ensemble classifiers.

Combining machine learning methods leads to computationally intensive solutions. Therefore, as a second contribution, we proposed a new deployment architecture, detailed in the next subsection, in order to take advantage of the ensemble methods without largely increasing the system latency.

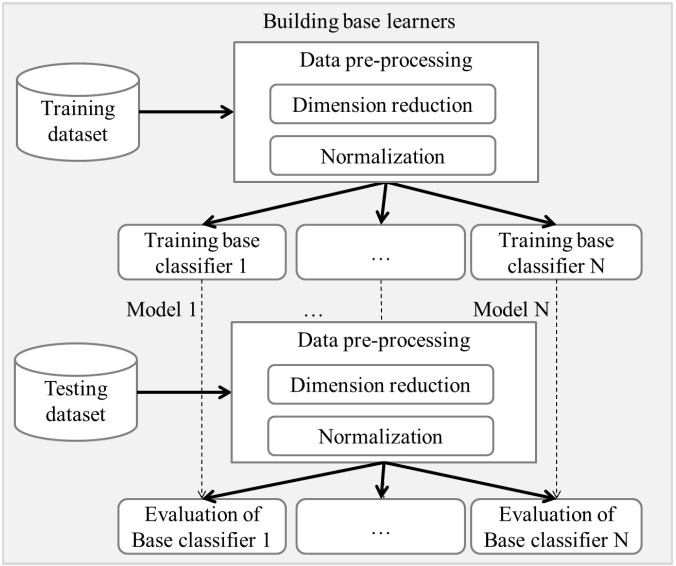


Fig. 3: Training and evaluation of base classifiers.

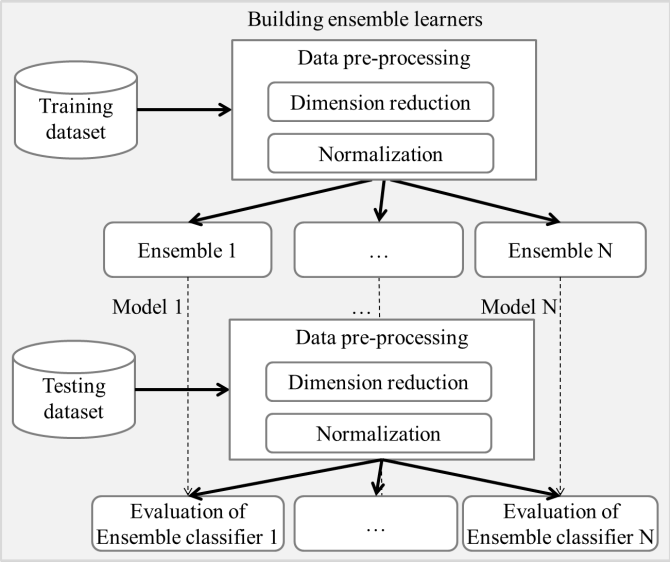


Fig. 4: Building ensemble classifiers.

B. The Deployment Architecture

The more complex the combination of base learners is, the more computationally intensive the ensemble learner is. Thus, in a fog-to-things environment, we proposed to split the problem into two tasks and distribute it between the fog nodes and the cloud. Fog computing extends the cloud computing and has a suitable architecture to run applications that require both low latency and considerable computation resources [17]. In the context of IoT, with the high number of latency-sensitive applications, fog computing could be an inevitable solution [18]. The deployment of the IoT applications in a fog computing architecture is also known as fog-to-thing solutions.

Anomaly detection essentially requires very low latency in order to allow fast response for reducing the potential damage an anomalous traffic can cause. Since it is a binary classification, it will also require a less complex model than a multi-class classification. Thus, we built an anomaly detection model using the ensemble method discussed in the previous section. This model is deployed in fog nodes as a first level classifier. When an anomaly is detected, the security administrator receives an alert in order to take urgent and temporary measures. Then, the anomalous traffic is sent to the cloud for the second task.

The attack classification requires a more complex model, demanding more resources. Moreover, this task is less latency-sensitive than the first one. In this vein, we built an attack classification model, using also the ensemble method, that we deployed in the cloud as a second level classifier. When the attack category is predicted, the information is sent to the security administrator in order to apply complementary and more precise prevention mechanisms.

Our deployment architecture is illustrated in Fig. 5. This architecture provides a faster intrusion detection system with attack classification for efficient prevention mechanisms. Our simulation results show that this new architecture is a promising solution for intrusion detection in a fog-to-things environment. The next two sections present the implementation of this approach using NSL-KDD dataset.

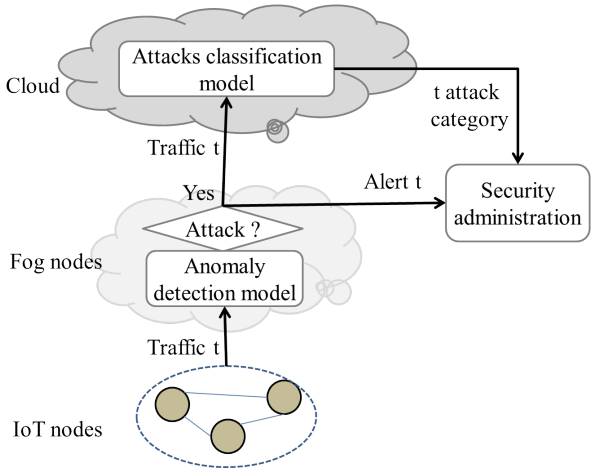


Fig. 5: Deployment architecture of the proposed IDS in fog-to-things environment.

IV. Description of NSL-KDD dataset

The NSL-KDD dataset is built from the KDDCUP'99 dataset that is also based on the data captured in DARPA'98 IDS evaluation program [3]. KDDCUP'99 training dataset contains approximately 4,900,000 single connection vectors each of which contains 41 features and is labeled as either normal or an attack, with exactly one specific attack type. The specific attack types are regrouped into different attack categories, namely probing (Probe), Remote to Local (R2L), Denial of Service (DoS), and User to Root (U2R). In the past, KDDCUP'99 dataset was widely used for evaluating machine learning models. However, one of its distinctive disadvantages was that even a simple machine learning model could easily have 99% of accuracy without using any advanced tuning operation either at the pre-processing or the training phase. Under such conditions, it was quite difficult to compare different machine learning models on this dataset.

In 2009, the work in [7] conducted a statistical analysis on this dataset and found important issues. The first issue is the huge number of redundant records. About 78% and 75% of the records are duplicated in the train and test datasets, respectively. In train dataset, this caused learning algorithms to be biased towards the more frequent records, and thus prevent them from learning infrequent records. For the test dataset, the huge redundancies caused the evaluation results to be higher for the methods with better detection rates on the frequent records. The second problem is the low level of challenge. After using 21 classifiers, most of the models gain a very high accuracy and very low false alarm rate. This was a problem in the way that it reduced the challenge and the advantage to compare different models. NSL-KDD is constituted by selection records of the complete KDD dataset in a way that

avoids the aforementioned shortcomings. The first issue was resolved by eliminating all redundant records from the training and the testing datasets. To solve the second issue, NSL-KDD created two challenging test datasets, namely KDDTest+ and KDDTest-21. One increased just the challenge by reducing "*easy to classify records*" (*i.e.*, records successfully classified by most of the 21 models employed by the authors) and the second that eliminated all the records that were "*the easiest to classify*" (*i.e.*, records successfully classified by all the 21 models), making definitely difference between different models. Table II presents the NSL-KDD train dataset (KDDTrain+) and the two test datasets (KDDTest+ and KDDTest-21).

TABLE II: Organization of the NSL-KDD dataset.

	Number of records used in training and testing datasets		
Class	***KDDTrain+***	***KDDTest+***	***KDDTest-21***
Normal	67343	9711	2152
DOS	45927	7458	4342
Probe	11656	2421	2402
R2L	995	2754	533
U2R	52	200	2421
Total	125973	22544	11850

V. Experimentation results and discussions

This section presents the experimentation carried out and the results obtained. It also discusses these results and provides a comparison with the previous works. The experimentation is done at two stages, *i.e.*, the anomaly detection stage then attack classification stage. All the tasks are performed using the Python programming language and Scikit-learn (the free machine learning library). The computer used is equipped with Intel(R) Xeon(R) CPU E3-1225 v6 and 16.0GB RAM.

A. The Anomaly detection stage

We began the implementation with the anomaly detection stage. Table III presents the organization of the NSL-KDD dataset for the binary classification.

TABLE III: Organization of the NSL-KDD dataset for the binary classification.

	Number of records used in training and testing datasets		
Class	***KDDTrain+***	***KDDTest+***	***KDDTest-21***
Normal	67343	9711	2152
Attack	58630	12833	9698
Total	125973	22544	11850

Three base classifiers have been implemented including Decision Tree classifier (DT), K-Nearest Neighbour classifier (KNN), and Multi-Layer Perceptron classifier (MLP). Four ensemble methods have been implemented including Random Forest classifier (RF), Bagging Classifier, AdaBoost Classifier, and Voting Classifiers. In the Bagging and Boosting methods, Decision Tree classifiers are used as base learners. In the voting methods, we made different combinations of the three base learners and the three other ensembles according to the difference they presented in the prediction of each of the two classes. All the models were trained using KDDTrain+ with 38 selected features, based on the work defined in [8]. As the data normalization did not provide a significant improvement on the accuracy of most of the models, we did not use it at the anomaly detection stage. For each model, we slightly tuned the hyper-parameters and kept the best ensembles. Advanced tuning methods were not employed as we want base learners to be just reasonably accurate, because they are not combined according to their individual accuracy, but for their diversity. Fig. 6 gives the details of all models used and their accuracy with KDDTest+ dataset, then Fig. 7 gives the accuracy of these models with KDDTest-21 dataset. For each model the accuracy showed is the one got with the tuned hyper-parameters.

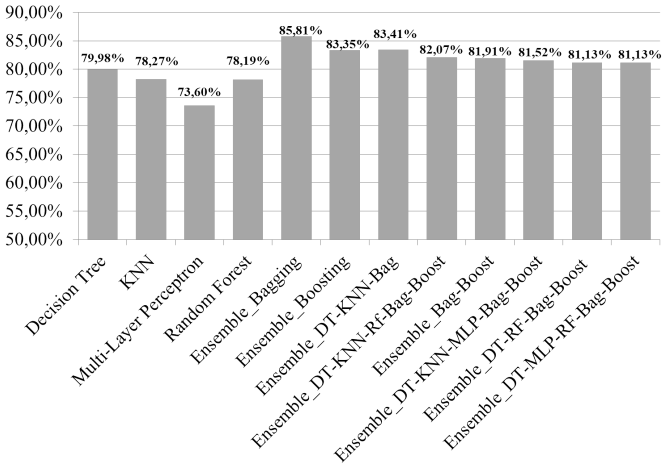


Fig. 6: Accuracy of anomaly detection models using KDDTest+.

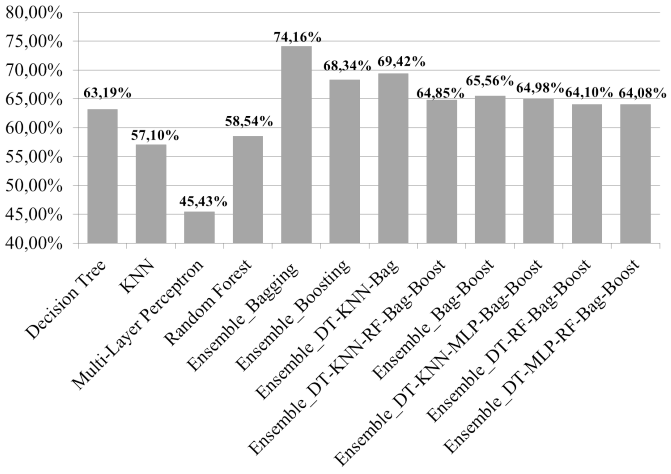


Fig. 7: Accuracy of anomaly detection models using KDDTest-21.

The Bagging classifier provided the highest accuracy for anomaly detection. In our environment, the training time was less than 7 seconds and the prediction time for all the KDDTest+ records was less than 3 seconds. Table IV compares this result with previous works on anomaly detection using the same train and test datasets. As some works did not test their models with KDDTest-21, these cells are filled with the symbol "-".

TABLE IV: Comparison of the accuracy of solutions using NSL-KDD dataset for binary classification.

Methods used	**KDDTest+**	**KDDTest-21**
Decision Tree Bagging Ensemble (Proposed method)	85.81 %	74.16 %
Four base learners and ensembles [15] Ensemble_J48_PART PART J48 C5.0 NB	- 84.84 % 82.61 % 82.13 % 77.94 % 77.26 %	-
Recurrent neural networks [11]	83.28 %	68.55 %
ANN with tansig transfer function, Levenberg-Marquardt (LM) and BFGS quasi-Newton Backpropagation (BFG) algorithm [10]	79.9 %	-
Self-Organization Map (SOM) Artificial Neural Network [9]	75.49 %	-

First, these results confirm that ensemble models can provide better performance than using a single complex model. Secondly, we can see that anomaly detection does not require complex voting methods in order to have a good prediction model. So, a simple ensemble model can be used in the fog nodes for efficient anomaly detection. The attack classification task can be performed as a complementary task in the cloud.

B. The Attack classification stage

The second part of the experimentation concerned the attack classification. Table V presents the organization of the NSL-KDD dataset for this four categories classification.

TABLE V: Organization of the NSL-KDD dataset for four categories classification.

	Number of records used in training and testing datasets		
Class	***KDDTrain+***	***KDDTest+***	***KDDTest-21***
DOS	45927	7458	4342
Probe	11656	2421	2402
R2L	995	2754	533
U2R	52	200	2421
Total	58630	12833	9698

As mentioned in the anomaly detection stage, different methods have been implemented including base methods (Decision Tree, KNN, MLP) and ensemble methods (Random Forest, Bagging, Boosting, and Voting). All the models were trained using KDDTrain+ with 38 features selected. As the data normalization provided a significant improvement on the accuracy of most of the models, we used it for this stage. For each model, we slightly tuned the hyper-parameters and kept the best ensembles. Fig. 8 gives the details of all models used and their accuracy in KDDTest+ dataset then Fig. 9 gives the accuracy of these models on KDDTest-21 dataset. For each model the accuracy showed is the one got with the tuned hyper-parameters.

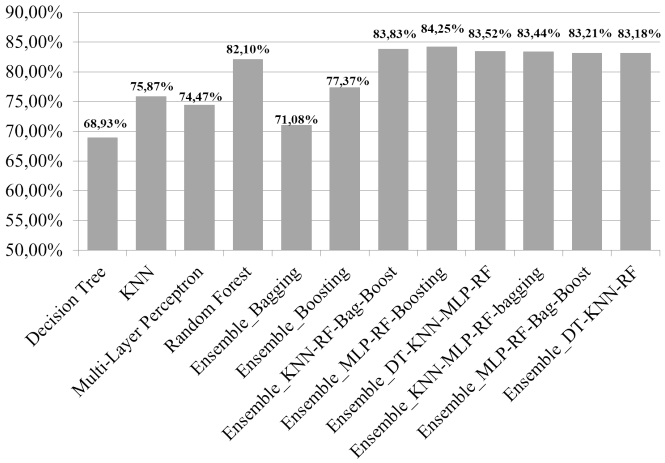


Fig. 8: Accuracy of attack classification models using KDDTest+.

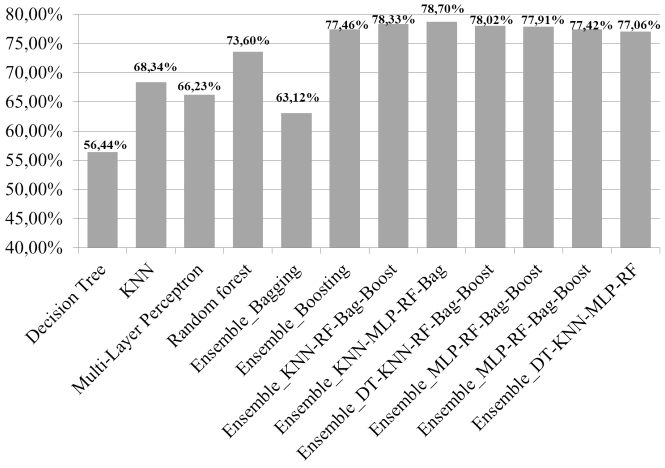


Fig. 9: Accuracy of attack classification models using KDDTest-21.

The voting ensemble using MLP, Random Forest, Bagging and Boosting as base learners gives the best accuracy on KDDTest+ dataset while the voting method using KNN, MLP, Random Forest and Bagging as base learners gives the best accuracy with KDDTest-21 dataset. However, the best model for the two NSL-KDD test datasets is the voting ensemble using KNN, Random Forest, Bagging and Boosting. This voting ensemble provides 83.83% of accuracy and 78.33% of

accuracy in KDDTest+ and KDDTest-21, respectively. So, we proposed this model for the attack classification stage.

Previous works that tried multi-class classification with NSL-KDD performed it using one model for five-categories classification (Probing, DoS, R2L, U2R, and Normal). Table VI gives a comparison of these solutions with our attack classification model (Four-categories classification). As some works did not test their models with KDDTest-21, these cells are filled with the symbol "-".

TABLE VI: Comparison of solutions using NSL-KDD dataset for multi-class classification.

Methods used	**KDDTest+**	**KDDTest-21**
Voting Ensemble using KNN, Random Forest, Bagging and Boosting of decision trees. Four-categories classification. (Proposed method).	83.83 %	78.33 %
Five methods [7] : NBTree Random Tree Decision Trees J48 Random Forest Multi-Layer Perceptron	- 82.02 % 81.59 % 81.05 % 80.67 % 77.41 %	- 66.16 % 63.97 % 63.26 % 58.51 % 57.34 %
Six methods [8]: SimpleCart J48 NBTree Nave Bayes Multi-layer Perception LibSVM	- 82.32 % 81.93 % 80.67 % 75.78 % 73.54 % 71.02 %	- 66.77 % 65.65 % 63.62 % 54.25 % 56.28 % 44.84 %
Recurrent neural networks [11].	81.29 %	64.67 %
ANN with tansig transfer function, Levenberg-Marquardt (LM) and BFGS quasi-Newton Backpropagation [10].	81.20 %	-

For the best of our knowledge, our approach gives the best attack classification model on the KDDTest+ and NSL-KDD-21 datasets. The proposed model uses a complex voting ensemble but as this task will be done in the cloud, the complexity can be handled thanks to the availability of more computation resources. In our experimentation environment, the training took maximum 80 seconds and the prediction for the whole KDDTest+ records took maximum 15 seconds.

Our experimental results show that this architecture is an efficient approach for intrusion detection a in the fog-to-things context in order to have a detection with low latency for fast reaction and accurate attack classification for precise correction using the required prevention mechanisms according to the attack categories.

VI. Conclusion

This work demonstrated that ensemble learners result to better classification models for the NSL-KDD dataset, the most realistic and challenging dataset available for machine learning based intrusion detection. The proposed solution employed diverse base learners using different known algorithms and built different ensemble classifiers for anomaly detection and attack classification. Overall accuracy of 85.81% and 84.25% have been achieved for the binary classification and attack classification, respectively. A deployment architecture was also proposed where anomaly detection was performed in fog nodes to allow faster detection and response, and attack classification in the cloud to benefit from more resources to run a more complex ensemble for better classification that guides the intrusion prevention tasks. Investigating more diverse base learners and various combination methods can improve better these results and is envisioned as future work.

References

[1] C. Perera, C. H. Liu, and S. Jayawardena, "The emerging internet of things marketplace from an industrial perspective: A survey," *IEEE Trans. on Emerging Topics in Computing*, vol. 3, no. 4, pp. 585–598, 2015.

[2] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of things in the 5G era: Enablers, architecture, and business models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, 2016.

[3] "KDD CUP 99 [online] available:," *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*, 1999.

[4] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," tech. rep., Miami Univ Coral Gables Fl Dept Of Electrical And Computer Engineering, 2003.

[5] D. S. Kim, H.-N. Nguyen, and J. S. Park, "Genetic algorithm to improve svm based network intrusion detection system," in *Proc. 19th Int. Conf. Advanced Inf. Netw. And Appl.*, vol. 2, pp. 155–158, 2005.

[6] P. G. Kumar and D. Devaraj, "Network intrusion detection using hybrid neural networks," in *International Conference on Signal Processing Communications and Networking - IEEE-ICSCN*, pp. 563–569, 2007.

[7] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Computational Intelligence for Security and Defense Applications - IEEE-CISDA*, pp. 1–6, 2009.

[8] K. Bajaj and A. Arora, "Improving the intrusion detection using discriminative machine learning approach and improve the time complexity by data mining feature selection methods," *International Journal of Computer Applications*, vol. 76, no. 1, 2013.

[9] L. M. Ibrahim, D. T. Basheer, and M. S. Mahmod, "A comparison study for intrusion database (KDD99, NSL-KDD) based on self organization map (som) artificial neural network," *Journal of Engineering Science and Technology*, vol. 8, no. 1, pp. 107–119, 2013.

[10] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Signal Processing And Communication Engineering Systems (SPACES) - IEEE*, pp. 92–96, 2015.

[11] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[12] F. Alimoglu and E. Alpaydin, "Combining multiple representations and classifiers for pen-based handwritten digit recognition," in *Document Analysis and Recognition. Proceedings of the Fourth IEEE International Conference*, vol. 2, pp. 637–640, 1997.

[13] L. Yu, S. Wang, and K. K. Lai, "Forecasting crude oil price with an emd-based neural network ensemble learning paradigm," *Energy Economics*, vol. 30, no. 5, pp. 2623–2635, 2008.

[14] P. Karvelis, S. Kolios, G. Georgoulas, and C. Stylios, "Ensemble learning for forecasting main meteorological parameters," in *Systems, Man, and Cybernetics (SMC), IEEE International Conference*, pp. 3711–3714, 2017.

[15] N. Paulauskas and J. Auskalnis, "Analysis of data pre-processing influence on intrusion detection using NSL-KDD dataset," in *Electrical, Electronic and Information Sciences (eStream), IEEE Open Conference*, pp. 1–5, 2017.

[16] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.

[17] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.

[18] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Trans. on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2018.