A cascade-structured meta-specialists approach for neural network-based intrusion detection

Maxime Labonne, Alexis Olivereau, Baptiste Polve, Djamal Zeghlache

A Cascade-structured Meta-Specialists Approach for Neural Network-based Intrusion Detection

Maxime Labonne
Institut LIST, CEA,
Université Paris-Saclay
F-91120, Palaiseau, France
maxime.labonne@cea.fr

Alexis Olivereau
Institut LIST, CEA,
F-91120, Palaiseau, France
alexis.olivereau@cea.fr

Baptiste Polvé
Institut LIST, CEA,
F-91120, Palaiseau, France
baptiste.polve@cea.fr

Djamal Zeghlache
Institut Télécom
Télécom SudParis
Évry, France
djamal.zeghlache@telecom-sudparis.eu

***Abstract*—An ensemble learning approach for classification in intrusion detection is proposed. Its application to the KDD Cup 99 and NSL-KDD datasets consistently increases the classification accuracy compared to previous techniques. The cascade-structured meta-specialists architecture is based on a three-step optimization method: data augmentation, hyperparameters optimization and ensemble learning. Classifiers are first created with a strong specialization in each specific class. These specialists are then combined to form meta-specialists, more accurate than the best classifiers that compose them. Finally, meta-specialists are arranged in a cascading architecture where each classifier is successively given the opportunity to recognize its own class. This method is particularly useful for datasets where training and test sets differ greatly, as in this case. The cascade-structured meta-specialists approach achieved a very high classification accuracy (94.44% on KDD Cup 99 test set and 88.39% on NSL-KDD test set) with a low false positive rate (0.33% and 1.94% respectively).**

Keywords—intrusion detection, ensemble learning, neural networks, data augmentation, NSL-KDD, KDD Cup 99

I. Introduction

In recent years, network security has become a major concern for businesses and states. High-profile attacks with negative impacts have captured the attention of the public and decision makers. This is why multi-layer security is now more necessary than ever to defend a network against inevitable attacks.

Intrusion detection within a network or system is provided by an Intrusion Detection System (IDS). Network-based IDS monitors the traffic generated by network entities in order to identify malicious activity. They traditionally rely on the detection of known attack signatures. They compare the signature of an activity with a signature database, and raise an alert when they find a match. However, the rapid rise in the number of attacks, both in their diversity and complexity, increases the difficulty to detect attacks using signatures.

An ideal IDS must therefore be able to detect attacks never seen before, with a zero false positive rate. Although there is no such IDS yet, anomaly detection is a popular solution to deal with this issue. Instead of detecting attack signatures, anomaly detection identifies unexpected activities. There are several ways to develop a classification or prediction software for anomaly detection, but machine learning is now the most efficient method to do it. Among these machine learning algorithms, neural networks have been highly successful in different areas such as image recognition [1] or natural language processing [2]. Neural networks have been applied to intrusion detection, but without spectacular results; they are indeed still challenged by other machine learning algorithms [3]. This difference can be explained by the constraints of this problem: any error can be very costly, the scope is broad, and current successful neural network architectures are not suitable to intrusion detection.

In this paper, we present an ensemble learning approach with neural networks to obtain the best possible performance for a 5-class classification, as is required by these datasets. The cascade-structured meta-specialists architecture is based on a three-step optimization method: 1/ data augmentation; 2/ hyperparameters optimization and 3/ ensemble learning. This study focuses on the two most popular datasets of this field: KDD Cup 99 and NSL-KDD. In order to show that our optimization method consistently gives better results than the state of the art, we compare it to other algorithms proposed in the literature.

To the best of our knowledge, the maximum classification accuracy obtained for NSL-KDD on test set is 85.016% with a two-level classifier ensemble [4]. Until now, the best neural network achieved a classification accuracy of 79.10% with self-taught learning [5]. Using the cascade-structured meta-specialists approach that we propose in this work, we obtained a classification accuracy of 88.39%. We also achieved superior results on KDD Cup with 94.44% accuracy on test set, against 92.70% for the winning entry of the competition [6]. We have ensured that these results were comparable, using 2 training processes for each dataset (training set only or training + test sets).

The rest of this paper is organized as follows. Section II describes the two datasets used in this paper and assesses data augmentation techniques. Section III introduces hyperparameters and two optimization techniques to maximize accuracy. Section IV details how the models obtained in the previous step are combined to create a better classifier. Section V provides concluding remarks.

II. Preprocessing and Data Augmentation

A. Datasets and preprocessing

For this work, we selected the two most popular datasets in intrusion detection: KDD Cup 99 and NSL-KDD. KDD Cup 99

was created in 1999 for a classifier learning contest [7]. The purpose of this competition was to accurately classify network connections as legitimate or malicious. Malicious connections fall into four main categories: Denial of Service (DoS), network probing (probe), Remote to Local (R2L) and User to Root (U2R). Each connection has 41 characteristics that allow the classifier to attempt to predict its class correctly (normal, DoS, probe, R2L or U2R). NSL-KDD was created in 2009 to address some of the inherent problems of KDD Cup 99 [8]. Redundant and duplicate connections, which comprised 75% to 78% of the dataset, have been removed. The total number of connections is thus much lower: 148,517 for NSL-KDD compared to 805,050 for KDD Cup 99. NSL-KDD also offers several subsets of different difficulties that we will not use in this paper.

KDD Cup 99 has been repeatedly criticized by the scientific community for its deficiencies. While NSL-KDD corrects the redundancy and duplicating problem, it is based on the same 1998 DARPA Intrusion Detection Evaluation datasets. These data include connections and attacks from 1998, recovered on a simulated U.S. Air Force LAN. The connections and attacks in these datasets are therefore not a good representation of the activity and threats of a modern network. The number of elements in the different classes of the two datasets is strongly imbalanced, which favors the recognition of the most frequent classes. In addition, the distribution probabilities of classes vary significantly between the training set and the test set (TABLE I).

TABLE I. KDD CUP 99 AND NSL-KDD DISTRIBUTIONS OF CLASSES

KDD Cup 99					
	Normal	***DoS***	***Probe***	***R2L***	***U2R***
Train	97278 (19.69%)	391458 (79.24%)	4107 (0.83%)	1126 (0.23%)	52 (0.01%)
Test	60593 (19.48%)	229855 (73.90%)	4166 (1.34%)	16345 (5.26%)	70 (0.02%)
NSL-KDD					
	Normal	***DoS***	***Probe***	***R2L***	***U2R***
Train	67343 (53.46%)	45927 (36.46%)	11656 (9.25%)	995 (0.79%)	52 (0.04%)
Test	9711 (43.08%)	7460 (33.09%)	2421 (10.74%)	2885 (12.80%)	67 (0.30%)

For this work, we did the same preprocessing for both datasets. First, the attack label of each malicious connection was transformed into one of the four attack classes. The values of the numerical features were then normalized between 0 and 1. Categorical features were finally one-hot encoded to be readable by the neural network [9].

B. Data augmentation

Learning from imbalanced data is a classic machine learning problem. The first step in our optimization process is to rebalance the training data, in order to obtain better results on the validation set.

There are two ways to rebalance classes: under-sampling and over-sampling. As the names suggest, under-sampling reduces the populations of the most represented classes, while over-sampling increases those of the least represented classes. In our case, we want to decrease the number of connections in the normal and DoS classes, and increase those in the probe, R2L and U2R classes. Thus, the classifier should not overlearn the most represented classes.

We compared 16 data augmentation methods on the two datasets to determine which ones give the highest Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC). The ROC curve is the plot of the true positive rate against the false positive rate. The AUC ROC score is the probability that a randomly chosen positive is ranked before a randomly chosen negative.

Our classifier is a deep Multi-Layer Perceptron (MLP) with 3 hidden layers, each composed of 128 units with a Rectified Linear Unit (ReLU) activation function. It is trained on 20 epochs with a batch size of 256 with the Adam optimizer. The training is carried out on 80% of the training set, and the validation on the remaining 20% of the training set. The AUC ROC score presented in TABLE II is an average value obtained by repeating this process 10 times per method. We developed our code using Python 3 with the imbalanced-learn package [10] for data augmentation, and Tensorflow [11] with Keras [12] for the neural network.

TABLE II. COMPARISON OF DATA AUGMENTATION METHODS FOR NSL-KDD

Method	**Normal class AUC ROC score**
No sampling	0.9453
Over-sampling methods	
Random Over Sampling	0.9595
SMOTE	0.9533
Borderline 1 SMOTE	0.9524
Borderline 2 SMOTE	0.9559
SVM SMOTE	**0.9693**
ADASYN	0.9493
Under-sampling methods	
Cluster Centroids	0.9416
Random Under Sampler	**0.9624**
NearMiss	0.9375
Edited Nearest Neighbours	0.9607
Repeated Edited Nearest Neighbours	0.9609
Condensed Nearest Neigbour	0.8818
AllKNN	0.9618
Instance Hardness Threshold	0.7254
Combination methods	
SMOTEENN	0.9561
SMOTE Tomek	0.9602

Results in TABLE II show that combination methods have not performed as well as expected [13]. Therefore, we manually combined the best under-sampling method (Random Under Sampler) with the best over-sampling method (SVM SMOTE). This combination gave the best results for the two datasets, with a ROC AUC score of 1.0000 for normal class on NSL-KDD validation set (processed in 37 minutes and 22 seconds). The

RUS + SVM SMOTE combination is used in the rest of the paper for both datasets.

III. Hyperparameters optimization

A. Hyperparameters and optimization techniques

Hyperparameters are the variables of a neural network set before training. This includes the number of neurons, batch size, optimizer, learning rate, activation functions, etc. Hyperparameters are often tuned manually, or by testing all possible combinations of a set of values (i.e., grid search). We chose two faster automated techniques: random search and Tree-structured Parzen Estimator (TPE) [14].

Random search randomly tests combinations of a range of values, with a fixed number of iterations. According to Bergstra and Bengio [15], random search can find better models than grid search and requires less computational time. The time allocated to this task is also easier to foresee, since the number of iterations is defined in advance. TPE is a Sequential Model-Based Optimization (SMBO) algorithm. Unlike random search, it chooses which parameters to test and converges to an optimal set of parameters. The choice of the optimization algorithm is data-dependent, that is why we test two different algorithms.

Optimization is a long process that is more reliable with few parameters. We manually defined as many parameters as possible that do not need to be optimized. For example, we know that the ReLU activation function [1] or the Adam optimizer outperforms the others. We also realized that any attempt at regularization, however slight, would lead to a decrease in accuracy. With this knowledge, the following hyperparameters have been optimized:

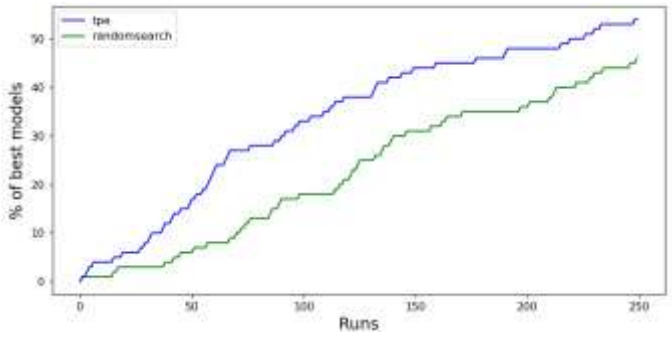
- *Number of hidden layers*: between 1 and 5 (step = 1).
- *Number of units in each hidden layer*: between 1 and 512 (step = 1).
- *Number of epochs*: between 1 and 200 (step = 1).
- *Batch size* = [16, 32, 64, 128, 256, 512, 1024, 2048].
- *Adam's parameters*: learning rate (between 10^{-5} and 0.2), beta 1 (between 0 and 1), beta 2 (between 0 and 1) and epsilon (between 10^{-9} and 10^{-5}).

We used the hyperopt library for its implementation of TPE [16]. Models have been trained on two GTX 1080 Ti GPUs.

B. Results

Random search was quickly abandoned in favor of TPE, which consistently obtains better and faster results. Figure 1 shows the distribution of the best models (i.e., the top 20% of models in terms of accuracy) with the two methods. The search space corresponds to the set of values covered by the optimization algorithm (minimum and maximum values of neurons for instance). We started with large search spaces, which were gradually reduced manually by observing the best results to speed up the process.

Fig. 1. Comparison of results obtained with random search and TPE.



Some hyperparameters clearly converged to an optimal value for both datasets. For example, a batch size of 256 obtained much better results than other values. Similarly, the number of layers quickly converged to a value of 1. KDD Cup 99 and NSL-KDD only have 41 features, far from the thousands of features found in image recognition. This may explain a lower need for generalization, and therefore a low number of layers. On the other hand, the numbers of units and epochs never clearly converged to a specific value. On average, they obtain better accuracy for values between 10 and 150, and between 5 and 60 for NSL-KDD respectively. Indeed, even after data augmentation, the distributions of classes in the training and test sets remain different: keeping a small number of epochs prevents overfitting. Likewise, the search of Adam's parameters values has been narrowed manually.

Several models have been built for each configuration of each dataset. In the first configuration, the model is trained with 80% of the training test, validated on 20% of the training test, and tested on the whole test set. In the second configuration, the model is trained on 60% of the training set, validated on 20% of the training set, and tested on the remaining 20% of the training set. This latter configuration gives better results, because the test set adds new attacks as well as a very different class distribution. The best models in each category are presented in TABLE III.

TABLE III. Classification Accuracies for Optimized Neural Networks on KDD Cup 99 and NSL-KDD

Train set	Test set	Number of units	Number of epochs	Adam's parameters	Accuracy
NSL-KDD train (80%)	NSL-KDD test	81	15	lr: 0.075 beta1: 0.213 beta2: 0.850 epsilon: 9.50×10^{-6}	84.70%
NSL-KDD train (60%)	NSL-KDD train (20%)	125	35	lr: 0.128 beta1: 0.125 beta2: 0.958 epsilon: 2.42×10^{-6}	99.29%
KDD Cup 99 train (80%)	KDD Cup 99 test	68	32	lr: 0.001 beta1: 0.9 beta2: 0.999 epsilon: 10^{-8}	93.77%
KDD Cup 99 train (60%)	KDD Cup 99 train (20%)	130	24	lr: 0.001 beta1: 0.9 beta2: 0.999 epsilon: 10^{-8}	99.95%

IV. ENSEMBLE LEARNING

A. Naive ensemble learning

Ensemble learning is a process combining several models to improve the overall predictive performance. This approach has been successful in many machine learning competitions, such as KDD Cup 2009 [13]. The general idea is that a combination of weak learners is more effective than a single strong learner.

We tested the performance of this approach by naively combining our two best models from the previous step. There are several combination rules to create an ensemble classifier: averaging their predictions, keeping only the maximum value, adding them up, multiplying them, etc. [18] We tested in TABLE IV different algebraic combiners to create the final prediction p from p_1 (model 1 with 84.70% classification accuracy on NSL-KDD test set) and p_2 (model 2 with 84.17% classification accuracy on NSL-KDD test set).

TABLE IV. COMPARISON OF DIFFERENT COMBINATION RULES FOR ENSEMBLE LEARNING ON NSL-KDD TEST SET

Combination rule	Prediction p for N models	Accuracy
Mean rule	$p = \frac{1}{N}\sum_{i=1}^{N} p_i$	**84.88%**
Maximum rule	$p = max_{i=1,\ldots,N}\{p_i\}$	84.82%
Sum rule	$p = \sum_{i=1}^{N} p_i$	**84.88%**
Product rule	$p = \prod_{i=1}^{N} p_i$	84.78%

All combination rules work better than the best model of the previous section, especially the mean and sum rules with a +0.18% increase in accuracy. This increase can be explained by the way classifiers make their predictions. On average, a classifier has less confidence in its predictions when they turn out to be false than when they are true. Combining a false prediction with a true prediction thus favors the latter.

B. Meta-specialists for ensemble learning

This statement led us to create classifiers specialized in the detection of a single class. These specialists can be 5-class or 2-class classifiers. We tested both approaches and obtained better results for 5-class specialists on normal classes, DoS, R2L and U2R (but not probe), that is why we continue to use 5-class classifiers in the rest of this paper.

We applied the same method as in the first and second sections for the training of these specialists. The preprocessing of the training set depends on the classifier's specialty. Indeed, the class in which the classifier is specialized is over-represented (1:5 to 1:30) compared to the others. First, all other classes are under-sampled with the Random Under Sampler. If the specialty of the classifier is probe, R2L or U2R, this class is then over-sampled around 20,000 connections. The specialist's neural network is optimized with the same search spaces obtained in section II. In addition to the hyperparameters, the class ratio is also optimized by the TPE on a validation set.

We then applied the ensemble learning method to each of the 5 sets of specialists. However, models have different accuracies: some perform better than the others on a dataset. Increasing the contribution of the best models in the final prediction would naturally lead to better results. But poor models should not be systematically excluded from the ensemble. They can indeed be specialized in rare forms of connections that the best models do not recognize. This way of favoring the best models can be implemented by adding weights λ_i to the prediction p_i of each model i in the previous combination rules. These weights are then optimized with the TPE on a validation set to maximize the AUC ROC score of the meta-specialist. In addition to the previous combination rules, we added majority voting, which selects the class that receives the largest total votes. We thus defined a meta-specialist as the composition of several specialists from the same class, and only participates in the classification of its specialty. Results with meta-specialists for NSL-KDD are shown in TABLE V. Mean rule achieved the same accuracy than sum and product rules but is faster to compute (approximately 1 hour and 20 minutes, depending largely on models). This is why we use mean rule for both datasets in the rest of this paper.

TABLE V. COMPARISON OF DIFFERENT COMBINATION RULES WITH META-SPECIALISTS FOR ENSEMBLE LEARNING ON NSL-KDD TEST SET

Combination rule	Prediction p of a meta-specialist for N models	Accuracy
Mean rule	$p = \frac{1}{N}\sum_{i=1}^{N} \lambda_i p_i$	**86.33%**
Maximum rule	$p = max_{i=1,\ldots,N}\{\lambda_i p_i\}$	86.01%
Sum rule	$p = \sum_{i=1}^{N} \lambda_i p_i$	**86.33%**
Product rule	$p = \prod_{i=1}^{N} \lambda_i p_i$	**86.33%**
Majority voting	$p = max_{i=1,\ldots,N} \sum_{i=1}^{N} \lambda_i p_i$	86.17%

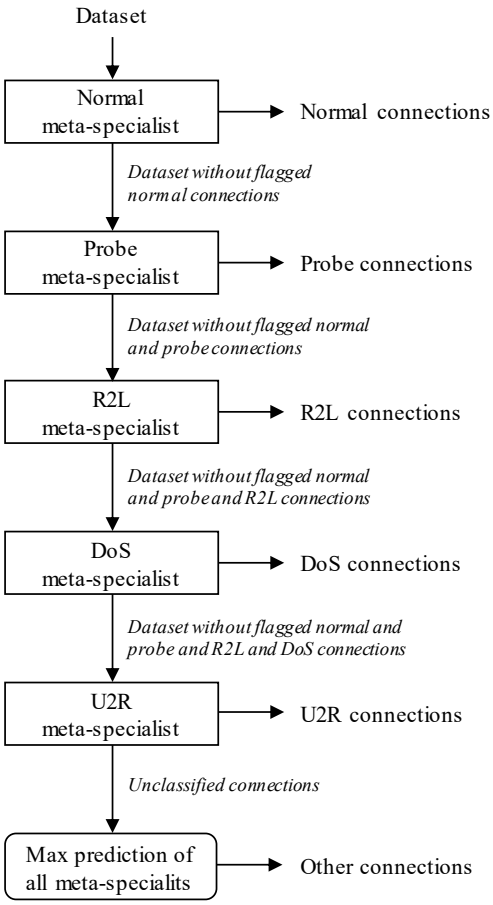
C. Cascade-structured meta-specialists architecture

Meta-specialists tend to over-recognize their own specialty in the connections presented to them. This is a problem for rare and therefore unreliable attacks like R2L and U2R, which can produce many false positives. This problem can be mitigated by presenting successively only the non-classified connections to the different meta-specialists, as shown in Figure 2.

In this architecture, the entire NSL-KDD dataset is first presented to the normal meta-specialist. This classifier only classifies normal connections. Connections flagged as "normal" are subtracted to the dataset, which is then presented to the probe meta-specialist. This process is repeated for R2L, DoS and U2R attacks. The order of meta-specialists was determined by selecting the one that gave the best AUC ROC score on the

validation set. All remaining connections, those that have not been recognized by any meta-specialist, are then classified. The class of each of these connections is determined by the meta-specialist with the highest probability.

Fig. 2. Cascade-structured meta-specialists architecture for NSL-KDD.



Unlike naive ensemble learning models, specialists have never been trained or validated on the test set, in order to avoid data leakage. Their performance was measured on a validation set (20% of the training set), despite its important differences with the test set. Indeed, validating the weight optimization of specialists on the test set would greatly improve the results: 92.66% classification accuracy on NSL-KDD test set.

TABLE VI presents the final performance for our architecture on KDD Cup 99 (with max rule combination) and NSL-KDD (with sum rule combination).

TABLE VI. CLASSIFICATION ACCURACIES FOR CASCADE-STRUCTURED META-SPECIALISTS ARCHITECTURE ON KDD CUP 99 AND NSL-KDD

Train set	Test set	Accuracy
NSL-KDD train (80%)	NSL-KDD test	88.39%
NSL-KDD train (60%)	NSL-KDD train (20%)	99.91%
KDD Cup 99 train (80%)	KDD Cup 99 test	94.44%
KDD Cup 99 train (60%)	KDD Cup 99 train (20%)	99.95%

These results are detailed by class for test sets in TABLE VIII according to the following metrics:

TABLE VII. CONFUSION MATRIX

		Predicted result	
		Negative	***Positive***
Actual result	***Negative***	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Accuracy is the ratio of correctly identified results.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- True Positive Rate (TPR) is the proportion of positives that are correctly detected.

$$TPR = \frac{TP}{TP + FN} \quad (2)$$

- False Positive Rate (FPR) is the proportion of negatives incorrectly flagged as positives.

$$FPR = \frac{FP}{TN + FP} \quad (3)$$

- Precision is the proportion of predicted positives that are indeed positives.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

- F1 score is the harmonic mean of precision and TPR

$$F1\ score = 2 \times \frac{precision \times TPR}{precision + TPR} \quad (5)$$

- The AUC ROC score is the probability that a randomly chosen positive is ranked before a randomly chosen negative.

TABLE VIII. SUMMARY OF TEST RESULTS FOR CASCADE-STRUCTURED META-SPECIALISTS ARCHITECTURES

KDD Cup 99 (classification accuracy = 94.44%)					
	Normal	***DoS***	***Probe***	***R2L***	***U2R***
Accuracy	98.10%	94.77%	99.73%	96.32%	99.97%
TPR	97.54%	98.74%	90.06%	36.28%	28.57%
FPR	0.33%	6.19%	0.14%	0.35%	0.01%
F1 score	0.9870	0.8803	0.8986	0.5089	0.2985
AUC ROC	0.9861	0.9628	0.9496	0.6797	0.6428
NSL-KDD (classification accuracy = 88.39%)					
	Normal	***DoS***	***Probe***	***R2L***	***U2R***
Accuracy	95.02%	92.37%	96.86%	93.40%	99.12%
TPR	88.87%	96.10%	85.38%	64.92%	35.82%

FPR	1.94%	10.46%	1.75%	2.42%	0.69%
F1 score	0.9220	0.9156	0.8540	0.7158	0.1943
AUC ROC	0.9347	0.9282	0.9181	0.8125	0.6756

Finally, TABLE IX shows a comparison study on NSL-KDD between our model and previous results in terms of classification accuracy and FPR. Our solution performs better than the best 2-class classifiers in the literature in both metrics.

TABLE IX. COMPARISON STUDY ON NSL-KDD

Study	***Accuracy***	***FPR***
Our solution	**88.39%**	**1.94%**
Two-level classifier ensemble [4]	85.016%	12.6%
Bagging (J48) + feature selection [19]	84.25%	2.79%
GAR-forest + feature selection [20]	85.05%	12.2%
SVM + feature selection [21]	82.37%	15%

V. CONCLUSION

In this paper, we presented a three-step methodology for optimizing intrusion detection with neural networks. The cascade-structured meta-specialists architecture is based on the creation of specialized classifiers in a single class. Specialists are first trained on a modified training set to over-represent their class. The hyperparameters of these classifiers are then optimized to maximize their accuracy on a validation set. Specialists of the same class are combined into a meta-specialist. Non-flagged connections in the dataset are successively tested by all meta-specialists. This system has proven to greatly improve the quality of detection on KDD Cup 99 and particularly on NSL-KDD, with a classification accuracy of 88.39% and 1.94% FPR. It could be applied to any other labeled dataset for intrusion detection, with a similar performance increase compared to a naive classifier.

This approach could be improved by combining neural networks with other machine learning algorithms (e.g., Random Forest or SVM). These algorithms are more deterministic than neural networks, and could thus compensate for certain deficiencies of the latter. Besides, preprocessing is done on the entire training dataset, but selecting a combination of data augmentation algorithms class by class would make more sense. This would help to extend the classification system to the attacks themselves rather than the categories.

ACKNOWLEDGMENT

This work was supported by the French-German project BERCOM jointly funded by ANR and BMBF under the grant number ANR-14-PICS-0001.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167.

[3] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Commun. Surv. Tutor.*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

[4] B. A. Tama, A. S. Patil, and K.-H. Rhee, "An Improved Model of Anomaly Detection Using Two-Level Classifier Ensemble," in *2017 12th Asia Joint Conference on Information Security (AsiaJCIS)*, Seoul, South Korea, 2017, pp. 1–4.

[5] Q. Niyaz, A. Javaid, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (Formerly BIONETICS), BICT*, 2016, vol. 15, pp. 21–26.

[6] "Results of the KDD'99 Classifier Learning Contest." [Online]. Available: http://cseweb.ucsd.edu/~elkan/clresults.html. [Accessed: 13-Apr-2018].

[7] "KDD-CUP-99 Task Description." [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/task.html. [Accessed: 18-Jun-2017].

[8] 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, 2009. CISDA 2009. IEEE Symposium on*, 2009.

[9] M. Z. Alaya, S. Bussy, S. Gaïffas, and A. Guilloux, "Binarsity: a penalization for one-hot encoded features," *ArXiv170308619 Stat*, Mar. 2017.

[10] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *J. Mach. Learn. Res.*, vol. 18, no. 17, pp. 1–5, 2017.

[11] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *ArXiv160304467 Cs*, Mar. 2016.

[12] F. Chollet, "Keras," 2015.

[13] S. Kudugunta and E. Ferrara, "Deep Neural Networks for Bot Detection," *ArXiv180204289 Cs*, Feb. 2018.

[14] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[15] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012.

[16] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in Science Conference*, 2013, pp. 13–20.

[17] "SIGKDD : KDD Cup 2009 : Customer relationship prediction." [Online]. Available: http://www.kdd.org/kdd-cup/view/kdd-cup-2009. [Accessed: 15-Apr-2018].

[18] R. Polikar, "Ensemble learning," *Scholarpedia*, vol. 4, no. 1, p. 2776, Jan. 2009.

[19] N. T. Pham, E. Foo, S. Suriadi, H. Jeffrey, and H. F. M. Lahza, "Improving performance of intrusion detection system using ensemble methods and feature selection," in *Proceedings of the Australasian Computer Science Week Multiconference on - ACSW '18*, Brisband, Queensland, Australia, 2018, pp. 1–6.

[20] N. K. Kanakarajan and K. Muniasamy, "Improving the Accuracy of Intrusion Detection Using GAR-Forest with Feature Selection," in *Springer*, 2016, pp. 539–547.

[21] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, 2014, pp. 1–6.