Improved Network Traffic Classification Using Ensemble Learning

Isadora P. Possebon*, Anderson S. Silva*, Lisandro Z. Granville*, Alberto Schaeffer-Filho*, Angelos Marnerides†
* Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
Email: {ippossebon, assilva, granville, alberto}@inf.ufrgs.br
†School of Computing and Communications, Lancaster University, United Kingdom
Email: angelos.marnerides@lancaster.ac.uk

***Abstract*—Despite the large number of research efforts that applied specific machine learning algorithms for network traffic classification, recent work has highlighted limitations and particularities of individual algorithms that make them more suitable to specific types of traffic and scenarios. As such, an important topic in this area is how to combine individual algorithms using meta-learning techniques in order to obtain more robust traffic classification metrics. This paper presents a comparative analysis among meta-learning approaches and individual classifiers to classify network traffic. We investigate and evaluate a range of meta-learning techniques, including *Voting*, *Stacking*, *Bagging* and *Boosting*. We then propose a new experimental analysis of different meta-learning techniques - also known as *ensemble learners* - and compare them with their own base classifiers when used individually. Finally, considering the emerging popularity of Neural Networks, we analyze this scenario using the Multi-layer Perceptron classifier. The experiments were performed with data provided by the UCI Machine Learning Repository. The best performance was obtained by an ensemble technique (Bagging), which obtained accuracy of 99.972% and false positive rate of 0.00018%.**

I. Introduction

One of the main challenges in automating the detection and classification of anomalies in modern computer networks is the fact that different anomalies present diverse spatio-temporal network traffic characteristics; as such, a single detection and classification process is unlikely to be effective [1]. It is also resource intensive and actually infeasible to precisely describe all anomalies of a domain, since the set of anomalies does not remain the same; new anomalies typically emerge when system domains evolve with new features, enhancements, and fixes. Still, because new features continue to appear over time, anomaly detection systems should be flexible enough to accommodate new conditions, instead of being restricted to a steady set of predefined anomalies. One of the approaches that have been used to cope with this scenario is the use of machine learning-based classifiers [2].

Research in the area shows that anomalies can be detected, to some extent, by base classifiers individually [1]. In order for traffic classifiers to acquire new skills and adapt to different environments, they should learn from previous experiences rather than considering each isolated classification task. This learning-to-learn (*meta-learning*) [3] approach is a critical step for achieving versatile traffic classifiers. This topic has become especially attractive based on the premise that meta-classifiers are often more accurate than the individual classifiers that make them up [4]. The area of meta-learning is also known as ensemble learning [5].

Ensemble learning includes a wide range of research efforts that seek to find the best methods to build combinations of classifiers [6]. Works such as [7] and [8] started to investigate meta-learning techniques in the context of network traffic classification. However, they considered a more limited set of meta-learning techniques. In this paper, we present a comparative study between different meta-learning techniques and individual classifiers within the scope of network traffic. Thus, we can determine the best technique to be used in this context. The classifiers are employed to distinguish normal from attack traffic from a data set containing real traffic data. For this, we selected four meta-learning techniques commonly presented in the literature, defined the base classifiers to be used, compared the performance of these techniques among each other, and also the execution of the same base classifiers when used individually.

The main contributions of this paper are: *(i)* a study of the state-of-the-art and an investigation of the main existing techniques for meta-learning, *(ii)* an architecture that classifies network flows using different techniques of ensemble learning, and *(iii)* a comparative study between the application of meta-learning techniques and base classifiers.

This paper is organized as follows. In Section 2, we present the theoretical basis for this work. In Section 3, we propose an architecture for network traffic classification using ensemble learners. In Section 4, we present experimental results and associated analysis. Finally, in Section 5, we discuss the conclusions and final considerations of this work.

II. Background and Related Work

For the purposes of this paper, a network anomaly is a sudden deviation from the normal behavior of the observed network traffic [9]. Different network anomalies can manifest through specific deviations in the traffic features of network flows. For example, DoS attacks might have low interarrival time (time between the arrival of two packets), while port scanning attacks could be identified by multiple requests for different ports sent from the same source. Algorithms based on

978-1-7281-2999-0/19/$31.00 ©2019 IEEE

pre-established models (rule-based techniques that work under some heuristics) are restricted to the detection of pre-defined anomalies and are too sensitive to any changes in the nature of traffic, which may lead the monitored flow to be misclassified. On the other hand, learning algorithms are able to learn the most relevant metrics and to identify possible variations of a stream considered normal.

Previous research efforts already used ensemble learning techniques to detect anomalies. For instance, the work presented in [10] proposes an Intrusion Detection System (IDS) based on ensemble learning. Each base classifier is trained with a distinct set of features that represents patterns of the data set. Then, the results from each base classifier are combined. This is motivated by the observation that experts combine attack characteristics from different sets of features to generate new attacks. Three different combination rules were used: majority voting, mean and belief function call - probabilistic estimates based on observed patterns. The authors evaluated their proposal with the DARPA 1998 data set. The results showed that ensemble learners are able to reduce the overall error rate, but also decrease the generalization capacity of the system.

In the area of network traffic classification, existing work exploring the use of meta-learning is limited to the use of Stacking and Voting techniques [11] [8], or only execute experiments with ensemble classifiers that have the same paradigms (variations of the same base classifier) [12].

He et al. [11] present a new machine learning model that combines meta-learning with co-training techniques (semi-supervised technique, which uses several training subsets, where only some of them are classified). This work compares previous approaches that use individual classifiers with the new proposal. This helps overcome three major shortcomings: limited flow precision rate, low adaptability of machine learning techniques, and the need for a set of classified training data. The data set, like the work proposed here, is represented by a set of features extracted from flow records. Experiments showed the effectiveness of the technique.

In addition, the work of Wang et al. [8] proposes a classification approach based on subflow characteristics, using meta-learning. A flow truncation method was developed for real-time processing, and an aggregation machine learning system based on the accuracy of each classifier for different applications. The authors performed experiments with real data, which verify the effectiveness of the proposed methods. The final result shows that the proposed method performs, on average, 8% better than the base classifiers used, and about 3% better than the best of the base classifiers used.

Finally, Gmez et al. [12] also use ensemble learning for classifying network traffic. The authors compare the performance of seven popular ensemble algorithms based on Decision Trees, focusing on model accuracy, latency and byte accuracy (total number of bytes it transfers). They show that some of these algorithms overcome single Decision Tree in terms of model accuracy and byte accuracy. Additionally, a novel ensemble classifier is presented. This classifier is able to exploit imbalanced populations presented in traffic networks data sets to achieve a faster classification. Results show that ensemble techniques can, in fact, present better accuracy and low latency results.

In contrast to the above works, we aim to study the impact of the use of ensemble learners in different metrics – accuracy, precision, recall, false positives and negatives, true positives and negatives, error rate and F1-score – considering an *extended set of ensemble learning techniques*. Differently from the research efforts described above, the *performance of the base classifiers* when used individually and *Multi-layer Perceptron classifier* was also measured.

III. Designing a meta-learning based system for classifying network traffic

In the context of this work, the anomaly detection process is performed offline, *i.e.*, based on previously collected data and using supervised classifiers. The process consists in, given a set of labeled data, composing a training set for the system. This training set is used for training base classifiers and some of the ensemble learning techniques. Next, the validation set is classified by all classifiers in the system. The final stage consists in analyzing the results to compare the performance of the classifiers. The following subsections present a brief description of the meta-learning techniques considered, and the proposed architecture for classifying network traffic using ensemble learning.

A. Meta-learning

In this section, we review the main meta-learning concepts and the four techniques used in the proposed architecture. Meta-learning can be defined as learning from information generated by other learning systems [3]. The task of a system that applies meta-learning techniques is to combine the different classifiers, *i.e.*, to learn an integration rule based on the behavior of the trained classifiers. The main meta-learning techniques available in the literature are:

1) Voting: Each base classifier is entitled to one vote and the classification with the highest number of votes is the final prediction. For this approach, an arbitrary number of classifiers may be used. If there's an even number of classifiers, a tie-break rule should be determined. The most voted prediction is elected the final prediction for this instance.

2) Stacking: The process works with a layered architecture [13]. Each layer is composed by one or more classifiers. The prediction of a layer is used to extend the original feature vector of the corresponding instance. Figure IV represents the two-tier architecture we used. The first layer consists of 3 classifiers: SVM, KNN and Decision Tree. Each of these classifiers is trained with a subset of instances. Given a new instance to be classified, each classifier will produce a prediction. The predictions of these classifiers are then combined by majority voting. The resulting prediction is used to extend the feature vector of this instance, and this vector is the input for the next layer, which is the Decision Tree classifier. Finally, with this new feature vector, the Decision

Tree will predict a class for this same instance. The prediction of this last layer is the final prediction for the instance.

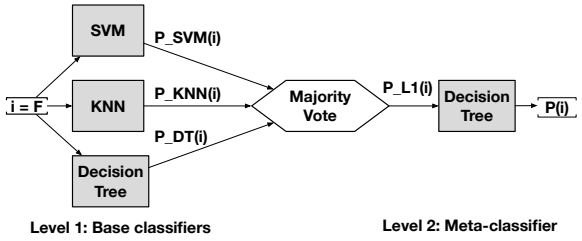

Fig. 1. *Stacking* using SVM, KNN and Decision Tree as the base classifiers at the first level, and Decision Tree at the second level.

3) Bootstrap Aggregating (Bagging): Bagging is a technique that generates a combination of classifiers by manipulating the training set provided to a base classifier. It consists of selecting a single base classifier and invoking it several times, using different training sets [14]. In this method different subsets of training data are created. These subsets are formed by randomly selected instances of the original training set. Each training subset is used to train a base classifier. The predictions of each of the base classifiers are combined by a predefined combination rule.

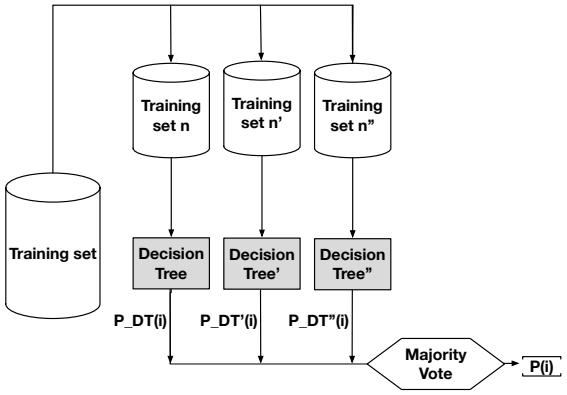

Fig. 2. *Bagging* using *Decision Tree* as the base classifier.

Figure IV shows the process of classifying an instance *i*, represented by the set of features *F*. The meta-classifier consists of a combination rule (in the example, majority voting) to combine these different predictions and provide the final classification *P(i)* of instance *i*.

4) Boosting: Unlike the *Bagging* strategy, *Boosting* creates different base classifiers through a process in which the instances of the data set sequentially receive new weights. In the first step, all instances are initialized with uniform weights. After this initialization, each iteration adapts a base classifier to the training instances with their respective weights. The error is computed and the weight of the correctly sorted instances is reduced, while the weight of the incorrectly sorted instances is increased. The final model obtained by the *Boosting* technique is a linear combination of several base classifiers, weighted for their best performance. This process is repeated until the desired accuracy is achieved or until no improvement is achieved. The version of *Boosting* used in this work is *AdaBoost* [15].

B. Ensemble learning architecture for traffic classification

In this section, we detail the proposed architecture for traffic classification using ensemble learning, considering the aspects presented in the previous sections. The meta-learning techniques used are *Voting*, *Stacking*, *Bagging* and *Boosting*. Figure 3 illustrates the concept and the following subsections provide details about it.

1) Data collection and labeling: Considering that we rely on supervised learning, a set of already classified data is needed. The data set used in these project was obtained from Meidan et al. [16]. Originally, the authors aimed distinguishing between benign and Malicious traffic data by means of anomaly detection techniques, but they were able to detect 10 different attacks a part from normal traffic. For the purpose of this work, we focus on the distinction of anomalous traffic into one single class, thus one instance is classified as normal or anomalous according to its features.

Data collection. This data set was obtained from a novel network-based anomaly detection method which extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic emanating from compromised IoT devices. The authors were able to infect nine commercial IoT devices with two of the most widely known IoT-based botnets, Mirai and BASHLITE [16].

Feature extraction and data labeling. For each packet, a behavioral snapshot of the hosts and protocols that communicated this packet is taken. The snapshot obtains the packet's context by extracting 115 traffic statistics over several temporal windows to summarize all of the traffic that has (1) originated from the same IP in general, (2) originated from both the same source MAC and the same IP address, (3) been sent between the source and destination IPs (channel), and (4) been sent between the source to destination TCP/UDP sockets (socket). The 115 attributes for each instance are generated from an extraction of the same set of 23 features from five time windows of the most recent 100ms, 500ms, 1.5sec, 10sec, and 1min. These 23 features include packet size, packet count and packet jitter (the amount of time between packet arrivals). All data used for this research is available at UCI Machine Learning Repository[1].

2) Classification stage: The processed data from the previous step is sent to the classification stage, that uses a meta-learning system and the set of individual base classifiers. The division of the data set between training and validation sets is performed according to user-set parameters. Also parameterized by the user is the configuration of the individual classifiers and ensemble learners of the meta-learning system.

Base classifiers. The base classifiers adopted in our work were *Multi-layer Perceptron* (MLP), *Decision Tree* (DT) and *K-Nearest Neighbors* (KNN). The choice of these classifiers

[1]https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

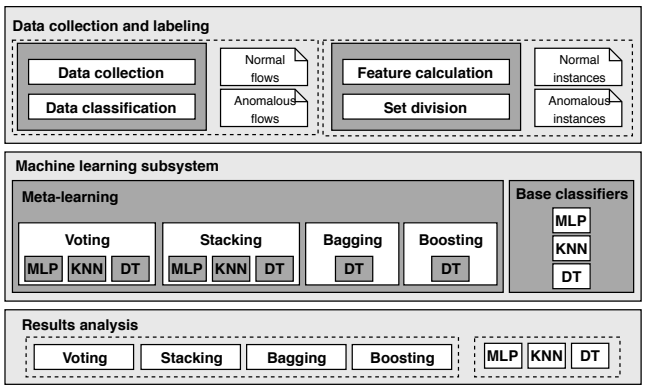

Fig. 3. Ensemble learning architecture for traffic classification.

was due to the fact that they are unrelated algorithms - with different error rates for the same data set and because they are algorithms widely used in the literature in the context of meta-learning.

For the purposes of evaluating the prototype, the behavior of the individual classifiers was also analyzed. That is, the classifiers MLP, DT and KNN were applied alone to a set of data with the same settings. The prototype is based on the *scikit-learn*[2] library version 0.19.1 and Python version 3.6.

Meta-learning techniques. Again *scikit-learn* was used, because it also provides an implementation for the *Voting*, *Bagging* and *AdaBoost* techniques. To ensure a fair evaluation, the base classifiers used for ensemble learning were created with the same parameters as when they were used individually.

Since *Stacking* was not directly supported by the *scikit-learn* library, we further developed a custom implementation of this technique. At the first level, each classifier provides, for each instance, an associated classification. The base classifiers used at this level were the same as those used by the other meta-learning techniques to validate the comparison process. Thus, each instance of the set has three associated classes, which correspond to the classification provided by each of the base classifiers. The combination rule used is majority voting as well. The final classification is appended to the feature vector of the instance. Then, at the second level, we receive the new vector of features associated with the instance (now with the classification obtained in level 1). Level 2 consists of a single classifier. For the developed prototype, the classifier chosen for level 2 was a *Decision Tree*. It receives the feature vector from each instance and, based on it, provides the final classification.

3) Result analysis: The following metrics were extracted from the obtained results: number of false positives, number of false negatives, number of true positives, number of true negatives, recall, precision, f1-score and mean accuracy for a given number of experiment repetitions [17]. A positive classification corresponds to an anomalous flow, whereas a negative classification corresponds to a normal flow. Therefore, the number of false positives corresponds to the number of instances considered normal flows that were classified as anomalous flows. Likewise, the number of true positives corresponds to the number of instances considered anomalies that were classified as, in fact, anomalies. Analogously, we calculated the number of false negatives and true negatives.

The recall metric corresponds to the number of positive cases (anomalous flows) that the classifier was able to identify. Also, the precision metric corresponds to the number of positive predictions that were correct, that is, anomalous flows that were, in fact, anomalies. Ultimately, F1-score is the weighted average of precision and recall. Therefore, this score takes both false positives and false negatives into account.

As stated by Boutaba et al. [6], supervised learning tends to yield high classification accuracy, due to a priori information about the characteristics of the classes of interest. Moreover, one of the main challenges in classifying network traffic is to correctly identify anomalies previously unseen. Therefore, we focus on the false positives rate and the precision metric. These, allied to the accuracy metric, provide better insights about the obtained results.

IV. Experimental analysis

In this section we present an experimental analysis of the implemented meta-learning system. Our results describe *(i)* the performance metrics of the ensemble learners considering data derived from real network traffic traces. Further, we present results on *(ii)* the memory consumption and execution time of the algorithms. Finally, *(iii)* results are reported both for the ensemble learners and for the base classifiers used individually.

For evaluation purposes, we implemented a *repeated K-fold cross-validation process* [18]. For the experiments presented in this section, we used $K = 5$ and repeated the K-folding process five times, with shuffled data. This is done to avoid overfitting. The experiments were executed on a 2,6 GHz Intel Core i5 processor. The settings for each base classifier are as follows. The optimization of their parameters has been also evaluated in order to achieve better results. The following parameters correspond to the combination that yields the best results.

- **KNN**: Suppose an instance with coordinates (x, y). This will be assigned to the most common class amongst its K nearest neighbors measured by a distance function. For this work, euclidean distance is used, calculated as shown in Equation 1.

$$Distance = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \quad (1)$$

  In addition, *K = 3* because of the small set of anomalous instances (low values should be used).
- **MLP**: Given a training dataset of *n* points of the form $(\vec{x_0}, y_0), ..., (\vec{x_n}, y_n)$, where perceptron is a linear classifier; that is, it is an algorithm that classifies input by separating two categories with a straight line. A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the

[2]http://scikit-learn.org/stable/

input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP. The activation function used was RELU - Rectified Linear Unit function. For weight optimization, we used the solver LBFGS to converge faster and perform better, suggested by scikit-learn documentation. We configured the network with 8 hidden layers, with 110, 100, 50, 20, 15, 10, 5 and 2 perceptrons respectively. Additional experiments with 21 hidden layers (with 110, 1000, 900, 800, 700, 600, 500, 450, 400, 350, 300, 350, 200, 150, 100, 50, 20, 15, 10, 5 and 2 perceptrons respectively) were also performed but did not yield better results.

- **Decision Tree**: the quality of a split was measured as a function of the entropy for information gained. Entropy is calculated as shown in Equation 3.

$$Entropy = -\sum_{j} p_j log_2 p_j \tag{2}$$

where p_j is the probability of belonging to the class j. The chosen split criterion was *best split* and we did not define a maximum depth.

Regarding the ensemble learning techniques, the following architecture and parameteres were used for these experiments:

- **Voting**: SVM, KNN and Decision Tree as base classifiers.
- **AdaBoost**: Decision Tree as base classifier, with 15 base estimators.
- **Bagging**: the architecture shown in Figure . Using Decision Tree as the base classifier and 10 base estimators.
- **Stacking**: SVM, KNN and Decision Tree as level 1 base classifiers, and Decision Tree as level 2 classifier - architecture shown in Figure .

A. Experiments

Due to resource limitations, the data set used in these experiments is a selection data provided by Meidan et al. [16]. We randomly selected 4000 instances –2000 considered normal and 2000 generated from attacks – of the so called Danmini Doorbell device[3].

Table I shows the performance of the algorithms with respect to accuracy, false positive rate, recall, precision and F1-score, respectively. Observing these results, we see that the individual classifiers are not correlated – that is, they do not provide the same results for the same data set. According to the literature, the use of highly correlated classifiers is detrimental to the performance of ensemble learners. For this reason, ensemble learning applied to non-correlated classifiers should yield better results.

Regarding the three base classifiers, Decision Tree presented the best results, while MLP, the worst. Overall, we see that Voting, AdaBoost and Bagging were able to improve the performance of its base classifiers. Stacking, on the other hand, was not. Considering the Voting approach, we can clearly see the majority voting heuristic: despite MLP bad performance, KNN and Decision Tree classifications prevailed.

[3]https://archive.ics.uci.edu/ml/machine-learning-databases/00442/Danmini_Doorbell/

TABLE I
RESULTS FOR THE PERFORMED EXPERIMENTS. $\bar{x}$ DENOTES THE MEAN VALUE, AND σ DENOTES THE STANDARD DEVIATION. THE HIGHEST VALUES ARE HIGHLIGHTED.

		Voting	AdaBoost	Bagging	Stacking	KNN	Decision Tree	MLP
False negatives	$\bar{x}$	0.84	0.76	0.76	0.44	0.88	0.72	**970.24**
	σ	1.1893	0.9912	0.8616	0.8040	0.9086	0.96	1009.9182
False positives	$\bar{x}$	0.88	0.84	0.36	**1050.6**	1.04	0.72	1050.28
	σ	1.0323	0.7838	0.5571	1008.8193	0.7736	0.7756	1009.1522
True negatives	$\bar{x}$	1998.92	1998.96	**1999.44**	949.2	1998.76	1999.08	949.52
	σ	25.9583	26.4340	26.3940	988.0181	26.8034	26.4407	988.3533
True positives	$\bar{x}$	1998.96	1999.04	1999.04	**1999.36**	1998.92	1999.08	1029.56
	σ	25.9438	26.1862	26.3249	26.0521	26.5223	26.1807	989.2455
Recall	$\bar{x}$	0.9996	0.9996	0.9996	**0.9998**	0.9996	0.9996	0.52
	σ	0.0006	0.0005	0.0004	0.0004	0.0005	0.0005	0.4996
Precision	$\bar{x}$	0.9996	0.9996	**0.9998**	0.7373	0.9995	0.9996	0.2574
	σ	0.0005	0.0004	0.0003	0.2521	0.0004	0.0004	0.2473
Accuracy	$\bar{x}$	**0.9996**	**0.9996**	0.9997	0.7372	0.9995	**0.9996**	0.4948
	σ	0.0003	0.0003	0.0002	0.2521	0.0003	0.0003	0.0041
Error rate	$\bar{x}$	0.0004	0.0004	0.0003	0.2628	0.0005	0.0004	**0.5052**
	σ	0.0003	0.0003	0.0002	0.2521	0.0003	0.0003	0.0041
F1-score	$\bar{x}$	0.4998	0.4998	**0.4999**	0.4121	0.4998	0.4998	0.1722
	σ	0.0001	0.0001	0.0001	0.0843	0.0001	0.0001	0.1654

AdaBoost uses unplaced sampling to create subsets of training data, used to train classifiers sequentially. Then, new classifiers are trained focusing on the error of previous classifiers. Because this base classifiers have different errors, there is a gain of information and therefore, a performance improvement.

With the *Bagging* approach, on the other hand, diversity is obtained from multiple replicas of the training set (obtained by sampling with replacement). There are two main reasons for its performance improvement: *(i)* the diverse subset generated; *(ii) Bagging* works best with base classifiers with high variance. Since MLP has high variance, there is a clear benefit in using *Bagging*, since we were able to achieve the lowest false positive rate (0.000018%) and highest precision and accuracy results.

Conversely, *Stacking* first level predictions are obtained by majority voting among the base classifiers. Because only MLP classifier was highly non-correlated to the other two (KNN and Decision Tree), there was not much gain of information for the second level. Then, its second level extends the original features set with the gained information on the first level and obtains a new prediction. However, since the gain of information on the first level is very low, there is little performance improvement.

It is also important to consider F1-score because this metric indicates the balance between precision and recall - that is, the overall quality of the classifier. When we look at the results, we see that *Bagging* has the highest score and, therefore, could be considered the classifier with the best quality.

Finally, considering that the main objectives of the classification are to increase accuracy and reduce the number of false positives identified, we see that the meta-learning technique *Bagging* is best suited for this problem. This technique was able to improve accuracy (resulting in 0.9997) and decrease the number of false positives (0.36).

B. Performance Analysis

Next, we analyzed each ensemble learning algorithm and their base classifiers in terms of training time. We evaluated the time needed to train a classifier considering repeated cross-validation, with k=5 and 5 repetitions. Table II shows the results for training time of each technique when classifying

TABLE II
TIME SPENT ON TRAINING PHASE FOR EACH ALGORITHM. $\bar{x}$ DENOTES THE MEAN VALUE, AND σ DENOTES THE STANDARD DEVIATION. THE HIGHEST VALUES ARE HIGHLIGHTED.

		Time spent on training (s)						
Number of instances		**10**	**50**	**250**	**500**	**1000**	**2000**	**4000**
Voting	$\bar{x}$	0.043	0.053	0.108	0.178	0.325	0.623	1.18
	σ	0.011	0.012	0.014	0.027	0.022	0.063	0.053
AdaBoost	$\bar{x}$	0.002	0.005	0.026	0.05	0.091	0.184	0.38
	σ	0.001	0.001	0.006	0.007	0.016	0.021	0.04
Bagging	$\bar{x}$	0.021	0.038	0.112	0.203	0.414	0.868	1.767
	σ	0.007	0.007	0.015	0.024	0.026	0.723	0.129
Stacking	$\bar{x}$	0.034	**0.149**	**0.904**	**2.264**	**6.535**	**21.112**	**67.752**
	σ	0.01	0.023	0.071	0.261	0.388	1.612	4.78
KNN	$\bar{x}$	0.0004	0.001	0.004	0.008	0.016	0.032	0.079
	σ	0.0001	0.0003	0.001	0.0005	0.002	0.002	0.005
DT	$\bar{x}$	0.001	0.004	0.021	0.04	0.084	0.184	0.379
	σ	0.0001	0.001	0.003	0.004	0.01	0.029	0.039
MLP	$\bar{x}$	**0.058**	0.067	0.102	0.147	0.255	0.452	0.795
	σ	0.102	0.104	0.096	0.1	0.128	0.11	0.106

different number of instances. At first, stacking approach stands out with the highest training time needed. In fact, Stacking is a technique that requires two different training phases, which explains the obtained results. More specifically, in our case, there are 4 training processes for this algorithm (its 3 base classifiers on the first level, and the final classifier on the second level). The remaining ensemble techniques require similar effort to its base classifiers. This, allied to their improvement on accuracy and false positives, reinforces its importance.

V. CONCLUSIONS AND FUTURE WORK

Although the literature on the use of machine learning for traffic classification is vast [6], there are a number of limitations and particularities of individual algorithms that make them more suitable to specific types of traffic and scenarios. As such, an important research topic in this area is how to combine individual algorithms using meta-learning techniques in order to obtain more robust classification metrics.

In this work, a comparative analysis was performed between ensemble learning techniques and individual classifiers to classify network traffic. Based on experiments with real data, meta-learning techniques presented clear benefits when compared to their base classifiers, mainly because the base classifiers were little correlated. Overall, the ensemble learners were able to reduce the number of false positives (Bagging achieved 0.36 false positives out of 4000 instances), except for Stacking, due to its few gain of information on its first level. Future work includes obtaining more representative data sets to conduct new experiments. It is also worth highlighting the possible in-depth study of the best individual classifiers to be used in this context, as well as the associated parameters. Additionally, emerging machine learning approaches like Deep Learning could be studied for this context [19].

ACKNOWLEDGMENTS

Alberto E. Schaeffer-Filho would like to thank CNPq for research grants ref. 407899/2016-2 and 312091/2018-4. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and also by NSF CNS-1740911 and RNP/CTIC (P4Sec) grants.

REFERENCES

[1] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016.

[2] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. E. Solano, and O. M. C. Rendon, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018. [Online]. Available: https://doi.org/10.1109/MCOM.2018.1700560

[3] P. K. Chan and S. J. Stolfo, "Experiments on multistrategy learning by meta-learning," in *Proceedings of the Second International Conference on Information and Knowledge Management*, ser. CIKM '93. New York, NY, USA: ACM, 1993, pp. 314–323.

[4] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine Learning*, vol. 54, no. 3, pp. 255–273, Mar 2004.

[5] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. Springer Publishing Company, Incorporated, 2012.

[6] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, Jun 2018.

[7] J. M. Reddy and C. Hota, "P2p traffic classification using ensemble learning," in *Proceedings of the 5th IBM Collaborative Academia Research Exchange Workshop*, ser. I-CARE '13. New York, NY, USA: ACM, 2013, pp. 14:1–14:4.

[8] C. Wang, X. Guan, and T. Qin, "A traffic classification approach based on characteristics of subflows and ensemble learning," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 588–591.

[9] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, April 2016, pp. 27–35.

[10] L. Didaci, G. Giacinto, and F. Roli, "Ensemble learning for intrusion detection in computer networks," in *AI*IA, Workshop on "Apprendimento automatico: metodi e applicazioni"*, Siena, Italy, 11/09/2002 2002.

[11] H. He, X. Luo, F. Ma, C. Che, and J. Wang, "Network traffic classification based on ensemble learning and co-training," *Science in China Series F: Information Sciences*, vol. 52, no. 2, pp. 338–346, Feb 2009. [Online]. Available: https://doi.org/10.1007/s11432-009-0050-8

[12] S. E. Gmez, B. C. Martnez, A. J. Snchez-Esguevillas, and L. Hernndez Callejo, "Ensemble network traffic classification," *Comput. Netw.*, vol. 127, no. C, pp. 68–80, Nov. 2017. [Online]. Available: https://doi.org/10.1016/j.comnet.2017.07.018

[13] R. Vilalta, C. Giraud-Carrier, and P. Brazdil, *Meta-Learning - Concepts and Techniques*. Boston, MA: Springer US, 2010, pp. 717–731.

[14] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, Aug 2000.

[15] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997. [Online]. Available: http://dx.doi.org/10.1006/jcss.1997.1504

[16] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, Jul 2018.

[17] S. Yingchareonthawornchai, D. N. Nguyen, V. T. Valapil, S. S. Kulkarni, and M. Demirbas, "Precision, recall, and sensitivity of monitoring partially synchronous distributed systems," *CoRR*, vol. abs/1607.03369, 2016.

[18] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statist. Surv.*, vol. 4, pp. 40–79, 2010. [Online]. Available: https://doi.org/10.1214/09-SS054

[19] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2432–2455, Fourthquarter 2017.