Applying Hoeffding Tree Algorithms for Effective Stream Learning in IoT DDoS Detection

João Gabriel Andrade de Araújo Josephik*, Yaissa Siqueira*, Kétly Gonçalves Machado*,
Routo Terada*, Aldri Luiz dos Santos†, Michele Nogueira†, Daniel Macêdo Batista*
*Department of Computer Science, University of São Paulo (USP), Brazil
joao.gabrielaaj@usp.br, {yaissa.siqueira,ketly,batista,rt}@ime.usp.br
‡Department of Computer Science, Federal University of Minas Gerais (UFMG), Brazil
{aldri, michele}@dcc.ufmg.br

2023 IEEE Latin-American Conference on Communications (LATINCOM) | 979-8-3503-2687-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/LATINCOM59467.2023.10361862

Abstract—**The constant evolution of botnets to cause DDoS and the non-existence of storage devices in IoT environments with high capacity to save the packets traveling on the network claim an IDS capable of constantly learning about new attacks without the need to save all the IoT traffic. A good solution to this scenario is to use stream learning, but it is important to guarantee that the learning model will be adaptable to concept drift. In this sense, Hoeffding Trees can be employed as the core of an IDS. This paper evaluates the effectiveness of using different Hoeffding Tree algorithms to detect DDoS against IoT devices. The work advances the state of the art by comparing different algorithms, adapting the basic algorithm to the considered scenario, evaluating a recently created dataset, and showing how the algorithms react to concept drift. Results show that the basic Hoeffding Tree algorithm is the most effective option when compared against the Hoeffding Adaptive Tree and ensembles of these trees (0.96/0.12 of average/standard deviation F1-Score vs. 0.90/0.26 and 0.91/0.24 respectively). As an additional contribution, all the code and data produced in the investigation are shared as open-source software and open data.**

Index Terms—**DDoS, IoT, Stream Learning, Hoeffding Tree, Ensembles.**

I. Introduction

In the last years, the number of "non-traditional" devices connected to the Internet has grown in an exponential way. The expectation is that the Internet of Things (IoT) market will value around 1.6 trillion dollars by 2025[1]. Due to the growing dependency we have on IoT devices and the economic forecasts in the area, the concerns about IoT security issues are an increasing discussion and target of recent studies. Typically we can divide IoT architecture in three layers: Sensor, Network, and Application layers [1]. Each one has its own vulnerabilities and different classes of attack can happen in any of them. Undoubtedly, attacks that draw the most attention are those that succeed, and in terms of IoT, the Mirai botnet is the most remembered [2]. The Mirai botnet attack in 2016 connected a massive amount of devices like IP cameras and routers, adding malicious code to their systems. It created a network of such infected devices and generated a DDoS (Distributed Denial of Service) attack towards important platforms like GitHub and Twitter. This attack has put DDoS and botnet attacks as the top IoT security problems, which brought attention to research in the field to prevent and mitigate future similar attacks.

Torii, a botnet discovered in 2018, is seen as a more sophisticated botnet than Mirai and is still under study by researchers worldwide [3]. It can infect a set of hardware architectures which is the wider set ever seen. Torii's main intention is to access private data, and it comes with several features capable of executing many important commands. Its enormous range of infected devices is probable to be used in a DDoS attack. It is hard to predict when or if Torii will create a DDoS attack or also to predict the emergence of smarter botnets. It is known that IoT devices will continue to be targeted, demanding more research in DDoS detection.

In this sense, Machine Learning techniques have been proposed as the core of Intrusion Detection Systems (IDS) based on anomaly detection and are a powerful asset in generating software to mitigate attacks. Specifically, in IoT environments, IDSs need to be deployed on devices with low storage capacity, as the access point of the network, for instance, which requires learning techniques capable of training and classifying traffic as the packets flow in the network without the need to store all the traffic for a large time interval. Moreover, the possibility of zero-day attacks with new DDoS tools demands the ability to fast adapt to a new traffic pattern. These characteristics are compatible with stream learning [4], where the training data arrives constantly, with no need to revisit data, and requiring low memory resources. Another feature of stream learning is the ability to adapt itself to deal with data distribution variations, known as concept drift [5].

In the work [4], the authors proposed an IDS based on stream learning to detect DDoS attacks in IoT networks. This detection model was based on decision tree learning, more specifically, on the Hoeffding Anytime Tree (HATT), an improved version of the Hoeffding Tree stream learning algorithm that uses the Hoeffding Bound to perform the splits of the decision tree [6]. Though, some research questions arose from that previous work: **(RQ1)** What is the performance of different Hoeffding Tree Algorithms when applied to IoT DDoS detection?; **(RQ2)** Is there any advantage of using ensembles of different trees?; and **(RQ3)** How to evaluate the adaptation to concept drift of the different trees? In the present

[1]https://www.statista.com/statistics/976313/global-iot-market-size/
Accessed on August 1, 2023

work, we will answer these questions considering a specific dataset and focusing on detecting DDoS attacks by analyzing the Network layer data, in other words, the data produced by the exchanged packets through the network. To evaluate the capacity to adapt to concept drift, we track the performance of the detection in moments where the attack traffic changes abruptly. To measure this capacity, we calculate the standard deviation of the F1-Score for each algorithm. The lower the standard deviation, the better the capacity.

Experiments with the ToN_IoT dataset show that the basic Hoeffding Tree algorithm is the most effective option when compared against the Hoeffding Adaptive Tree and ensembles of these trees (0.96/0.12 of average/standard deviation F1-Score vs. 0.90/0.26 and 0.91/0.24 respectively). As an additional contribution, we share all the code and data produced in the investigation as open-source software and open data[2].

The remainder of the paper is organized as follows. Section II summarizes related works, emphasizing how our paper advances the state of the art. Section III describes our proposal for using different types of Hoeffding Tree Algorithms for stream learning in IoT DDoS Detection. Section IV describes our testbed. Section V presents the results and the answers to the three research questions. Conclusions and future works are presented in Section VI.

II. Related Work

Horchulhack *et al* [7] discuss how the IDSs proposed in the literature fail to support the unexpected changes in the network traffic pattern, making them unsuitable for the real world. Using this as motivation, they propose a stream learning intrusion detection procedure focused on dealing with concept drift in the traffic. The core of the procedure is the Half-Space Tree algorithm. To serve as a baseline, the authors compared their proposal with the Very Fast Hoeffding Tree algorithm. We observe that the performance of the Hoeffding Tree algorithm was near the performance of the proposal. Different from [7] we evaluated the performance of more than one Hoeffding Tree algorithm and did not use time as the parameter to define how much data must be used for training (Since we are concerned about the amount of data to be stored, we consider the number of packets more important than the amount of time). Besides, to allow the reproduction of our experiments, all the code is publicly available, which is not the case in [7].

Krawczyk *et al* [8] gather a deep study about using machine learning ensemble techniques towards stream data. Ensemble methods can use more than one classifier or learning algorithm to decide the final prediction. The generated model is then an ensemble of all selected classifiers that uses a pre-defined function to output the response, which is a combination of the classifiers' responses. Krawczyk *et al* discuss the difference between static data and transient data streams, and how ensemble algorithms can obtain greater efficiency than single classifier algorithms when applied to stream data. Motivated by their findings, in this paper we evaluated the performance of ensemble techniques combined with Hoeffding Tree algorithms.

Arbex *et al* [4] proposed a Network Intrusion Detection System (NIDS) to detect DDoS attacks from botnets using IoT devices. Their work created a detection model as part of the NIDS, using stream learning techniques applied to the Bot-IoT Dataset [9]. The Hoeffding Anytime Tree Algorithm (HATT) [6], chosen by Arbex *et al* to develop their model, uses the Hoeffding Bound to decide the best branches it should generate during the classification tree creation. A gap left by the authors was the evaluation of different types of Hoeffding Tree algorithms and the combination of them by using ensembles. Also, there was no discussion about the detection of concept drift and how it affected the performance of the proposed model. In this paper, we extend the work in [4] by proposing different algorithms, combining these algorithms with ensembles, considering a more recent dataset, and showing how the algorithms react to concept drift. Like [4], all the code and data produced in the paper are also publicly available.

III. Employing Hoeffding Tree Algorithms to Detect IoT DDoS

Arbex *et al* [4] have proposed a Network IDS that can be found in [10]. The proposed system includes three modules: Traffic Collection, Anomaly Detection, and Mitigation Actions. Traffic Collection and Mitigation Actions modules will not be addressed in this paper. Our work proposes to reassure the use of stream learning applied to IoT DDoS detection by evaluating different Hoeffding Tree algorithms to generate new models to be used on the Anomaly Detection module.

A conventional decision tree works like this: each node in the tree makes a decision based on comparing an input feature with a threshold. From this decision, the data is forwarded to a different subtree, where the process is repeated. Upon reaching a leaf, an output is produced. The tree grows from the division of the nodes. The choice of feature and threshold is usually made based on calculating the Information Gain or another similar metric. The problem is that calculating these metrics requires storing the entire dataset. This, in certain cases like IoT environments, may be unfeasible. Thus, Hoeffding Trees use the Hoeffding inequality to guarantee that, with a certain probability, the choice to split a certain node made after looking at n entries, where n is as small as possible, is the same as what would be done after looking at all the data [11].

In Hoeffding Adaptive Trees [11], instead of simply keeping the number of entries in each node, "estimators" are used, such as ADWIN [12]. With this, we can detect changes in the data. If a node detects that there has been a change, it creates a new "alternative" tree with the new best feature at its root. If this alternative tree performs better, the node is replaced by the new tree.

Independent of the learning technique being applied, we can combine more than one aiming to improve the final result. These combinations are called ensembles [13]. Considering

[2]https://github.com/jooag/srp_ddos/tree/latincom2023/ Accessed on August 1, 2023

that, we are also proposing to combine several Hoeffding Trees with two ensemble techniques: Stream Random Patches (SRP) and Adaptive Random Forests (ARF), which are suggested in the literature as good options for stream learning.

SRP is an ensemble technique that can use several estimators as a basis [14]. In our case, we are applying a Hoeffding Tree. A very important concept in ensembles is to induce model diversity. In this way, for the ensemble to be efficient, it requires different classifiers. When all classifiers always vote the same, the ensemble is meaningless. SRP does this in two ways: random sampling of instances and random sampling of features. As we do not have access to the entire dataset, it is impossible to choose a subset of instances. Thus, we define a weight, chosen from a Poisson distribution, for each instance. We also monitor data changes with ADWIN, and the model is restarted when it has degraded its performance too much.

ARF is an ensemble technique very similar to the SRP [15], whose main difference is how features are sampled. In SRP, sampling takes place once per model so that each model receives a subset of features at its creation, allowing the use of a wide range of different classifiers. ARF performs random sampling on each node of each model. Therefore, ARF modifies the inner workings of the classifier, and thus it cannot be used with different classifiers.

IV. Experimental Design

A. Dataset Preparation

The ToN_IoT dataset [16] is a modern public dataset that can be used to evaluate the performance of an IDS. It is less imbalanced than its predecessor (Bot-IoT) [17]. There are $6,165,008$ DDoS attack entries and $796,380$ entries of normal traffic. In our experiments, we filtered the dataset keeping only the packets labeled as DDoS and Benign. After that, we randomly selected 10% of the dataset and evaluated the techniques on this selection. In terms of features, we considered only these connection activity features: *duration*, *src_bytes*, *dst_bytes*, *missed_bytes*, *src_pkts*, and *dst_pkts*. The reasoning behind this selection was to keep the monitoring at the IoT Network Layer and to avoid deep inspection of the packets, which is recommended both to run the training on lightweight devices and to increase the privacy of the IDS [18].

B. Model Training

The library river [19] implements the basics for all the above-mentioned algorithms. Each model was trained using this library in a test-then-train methodology. The dataset entries were given to the model training as a stream simulation. Since we are applying stream learning, the models must be constantly retrained and applied to classify future packets based on the last viewed packets. Preliminary experiments showed that training with the last 10,000 packets to classify the next 40,000 packets provided good results (The study of the impact of different numbers of packets, or window sizes, is held as a future work). So, as the packets progress in the dataset, different models are being built.

C. Computational Environment

We carried out all the experiments on a computer equipped with an AMD Ryzen 5 2600 processor (6 cores @3.4 GHz) and 16 GB of RAM running Arch Linux with kernel 6.4.3. The models and the testbed were developed in Python and executed with version 3.11.3 of the interpreter. Further, the most relevant libraries used were river 0.15.0, numpy 1.24.2, and pandas 1.5.3.

D. Baseline models

We considered it essential to compare the results obtained by the Hoeffding Tree algorithms with other simpler models not oriented to stream learning. We choose two models usually employed in diverse domains: Logistic Regression and Perceptron. They can serve mainly as a baseline in terms of time since they are not expected to run longer than the models based on Hoeffding Tree algorithms.

Logistic Regression is a linear model. Each input feature is assigned a weight. Features are combined according to these weights and passed through a sigmoid function. In our case, the function used is $f(x) = 1/(1 + exp(-x))$. The output, therefore, is between zero and one. The loss function used is Cross Entropy. With this, the output can be interpreted as a probability. The probability is then converted to a decision (if the probability is greater than half, the output is 1; otherwise, the output is 0). Weights are optimized using stochastic gradient descent, specifically with the Adam algorithm [20]. The learning rate used was 0.1.

Perceptron is also a linear model. Like Logistic Regression, the output is a linear combination of the input features. The classification is then made from a threshold. Unlike Logistic Regression, the outputs cannot be interpreted as probabilities. The optimizer used in this case was stochastic gradient descent with a learning rate equal to 1.

V. Results and Discussion

The objective of this paper is to evaluate the effectiveness of different Hoeffding Tree Algorithms when they are used to detect DDoS attacks. This detection is made by classifying the traffic flowing in the network (In our case, the packets in the ToN_IoT dataset). We are interested in the correct classifications of attack and benign traffic, and this can be measured by the F1-Score using Equation 3. The *Precision* and the *Recall* used to calculate F1-Score are calculated by Equation 1 and Equation 2, respectively. On these last equations, *TP* stands for True Positive, which is the number of attack instances correctly classified as attack, *FP* stands for False Positive, which is the number of benign instances incorrectly classified as attack, and *FN* stands for False Negative, which is the number of attack instances incorrectly classified as benign. The higher the F1-Score, the better.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

Since different models are being built during the simulation, different values of F1-Score are measured as the reading of the dataset advances. A good technique would provide high F1-Score values even in instants with concept drift, which are instants when the distribution of attack and benign traffic have an abrupt change.

Fig. 1 and Fig. 2 show the F1-Score values (lines in blue) obtained with the techniques Perceptron and Logistic Regression, respectively. These two models are our baselines since they are not techniques designed to be applied in stream learning. To track the imbalance of the dataset, a red curve labeled "Ground truth" was added to the plots. Each point in the red curve represents the fraction of the last 10,000 records that were labeled as "attack" (This same configuration of lines will be used in all the figures).

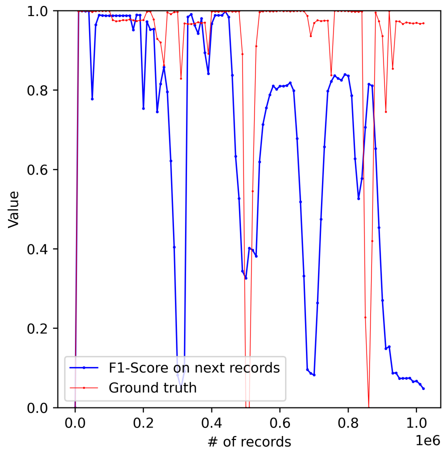

Fig. 1. F1-Score obtained with Perceptron (baseline 1)

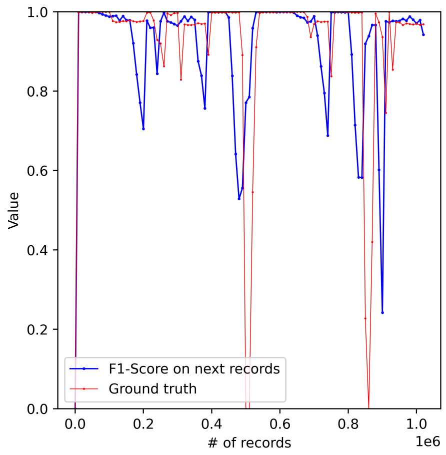

Fig. 2. F1-Score obtained with Logistic Regression (baseline 2)

The fact that the Perceptron and the Logistic Regression are not oriented to stream learning and in conditions with concept drift can be attested by the huge variations in the F1-Score. As can be seen in Fig. 1 in the middle of the dataset (number of packets approximately equal to $0.5e+6$), the abrupt reduction in the amount of attack packets affected the quality of the Perceptron model, which was not able to improve up to the end of the experiment. In the case of the Logistic Regression (Fig. 2), the variation in the middle of the experiment did not affect it as it affected the Perceptron but all the changes in the traffic are followed by worsening in the model.

Fig. 3 and Fig 4 present the results with the Hoeffding Tree and the Hoeffding Adaptive Tree algorithms, respectively. It is possible to observe that the variations in the traffic did not affect the quality of the model based on the Hoeffding Tree algorithm in the same way it affected the baseline models. In fact, this was the best result among all the models.

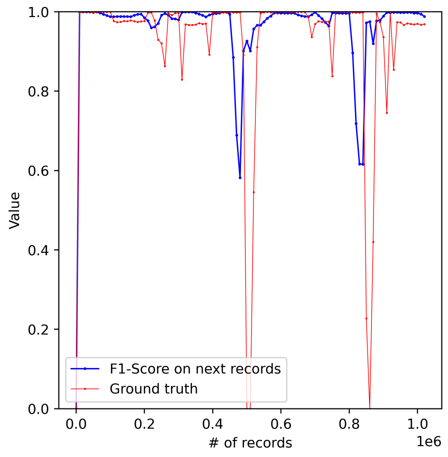

Fig. 3. F1-Score obtained with Hoeffding Tree

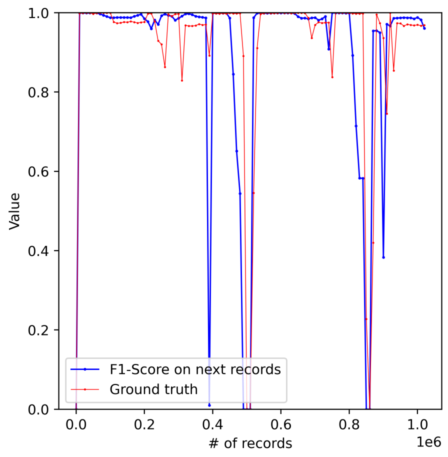

Fig. 4. F1-Score obtained with Hoeffding Adaptive Tree

Considering the performance of the Hoeffding Adaptive

Tree algorithm, the significant changes in the F1-Score in the instants with abrupt changes in the traffic (the instants with valleys in the red line) may be counterintuitive since this algorithm is more suitable for changes in the data streaming than the basic Hoeffding Tree algorithm. The explanation for this is related to the fact that in the traffic used in the experiments, we are dealing with what is called "virtual drift" [21]. In this case, the nature of the attack does not change, but the distribution of the data changes. When the Hoeffding Adaptive Tree algorithm detects the alteration in the traffic pattern, a new tree is used, which is not always a good decision.

Finally, we evaluated the ensemble techniques: Fig. 5, Fig. 6, and Fig. 7 present the results of Stream Random Patches with 1 Hoeffding tree, Stream Random Patches with 12 Hoeffding trees, and Adaptive Random Forest with 10 Hoeffding trees. It is possible to observe that the combination of the trees did not improve the result obtained in the not ensemble techniques. It is also possible to note that the increase in the number of trees (1 to 12) did not improve the performance of the Stream Random Patches. Besides, this came with an impact on the execution time of the model, which increased 11.40 times, which makes sense considering the 12 times increase of the trees. An important disclaimer is that, in practice, one would not use an ensemble technique with just one tree. We also evaluated this configuration to know how the increase in the number of trees affects the quality of the model.

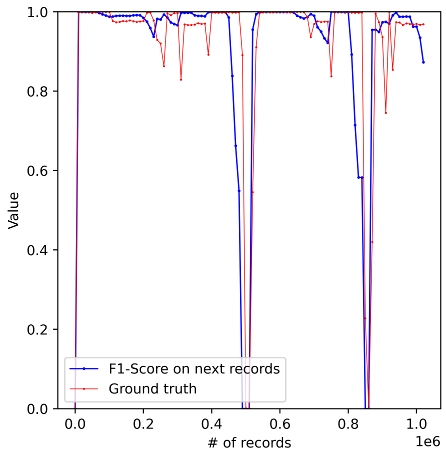

Fig. 5. F1-Score obtained with Stream Random Patches (ensemble of 1 Hoeffding tree)

The F1-Score values measured in all the models are summarized in Table I. The average and standard deviation of F1-Score are presented along with the execution time of the models. The results confirm what was observed in the graphs: The Hoeffding Tree algorithm was capable of producing good average F1-Scores with the lowest variation. Its execution time was not the best one but the decision based only on the execution time would not show to be a good one since

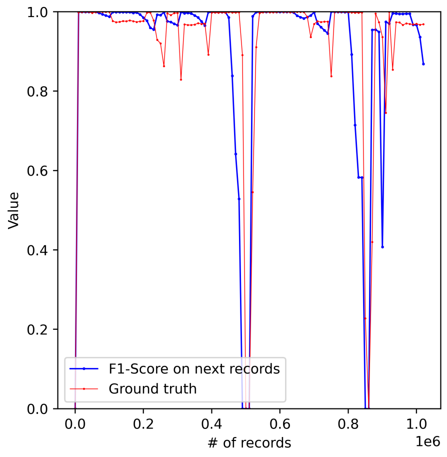

Fig. 6. F1-Score obtained with Stream Random Patches (ensemble of 12 Hoeffding trees)

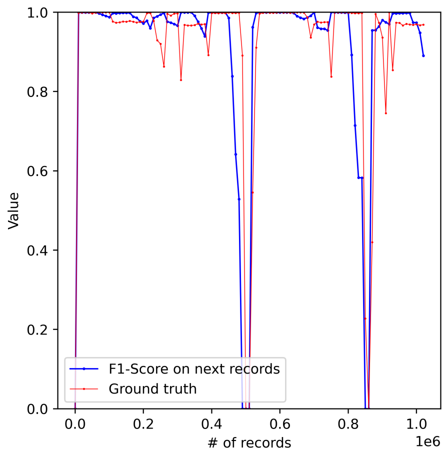

Fig. 7. F1-Score obtained with Adaptive Random Forest (ensemble of 10 Hoeffding trees)

the fastest model (Perceptron) provided the worst average F1-Score with a standard deviation that is 2.73 times higher than the standard deviation obtained with the Hoeffding Tree algorithm.

With these results, we can answer the research questions considering the traffic present in the ToN_IoT dataset:

- **(RQ1)** What is the performance of different Hoeffding Tree Algorithms when applied to IoT DDoS detection? **Answer**: The basic Hoeffding Tree is better than the Hoeffding Adaptive Tree when evaluating F1-Score (average and standard deviation) and the execution time;
- **(RQ2)** Is there any advantage of using ensembles of different trees? **Answer**: No. The increase in the number of trees does not reflect in improvements in the F1-Score, and it increases the execution time;
- **(RQ3)** How to evaluate the adaptation to concept drift of the different trees? **Answer**: By highlighting the moments

Algorithm	F1-Score (avg.)	F1-Score (stddev)	Execution Time (s)
(Baseline 1) Perceptron	0.657890	0.335546	14.92
(Baseline 2) Logistic Regression	0.916740	0.160472	25.37
Hoeffding Tree	0.958675	0.123007	59.05
Hoeffding Adaptive Tree	0.895259	0.262978	129.16
SRP (1 Hoeffding Tree)	0.906999	0.241647	277.37
SRP (12 Hoeffding Trees)	0.903169	0.247619	3162.70
ARF (10 Hoeffding Trees)	0.908579	0.242618	1069.79

TABLE I

F1-Score (average and standard deviation) and execution time for all algorithms. Best results are in the gray cells

when there are abrupt changes in the traffic pattern, for instance by showing them in a graph and measuring the dispersion of the F1-Score, for instance by means of the standard deviation.

VI. Conclusions and Future Works

The limitations in terms of processing and storage in IoT environments allied to the zero-day DDoS attacks justify the usage of lightweight techniques that can update a learning model constantly as the packets flow by. Considering that, this paper evaluated stream learning techniques based on Hoeffding Tree Algorithms. The evaluation of the techniques with the ToN_IoT dataset showed that the basic Hoeffding Tree algorithm was the most effective option when compared against the Hoeffding Adaptive Tree and ensembles of these trees. To allow the reproduction of all the experiments and incentivize the extension of our work, all the code and data produced in the investigation are shared as open-source software and open data. In future works, we plan to evaluate the techniques in a realistic environment by applying the detection in real-time and to evaluate the impact of different window sizes, adapting them to the memory capacity of the device being used to collect the traffic during the training phases.

Acknowledgments

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9. It is also part of the FAPESP proc. 18/23098-0 and FAPESP proc. 21/06995-0.

References

[1] A. Tewari and B. Gupta, "Security, Privacy and Trust of Different Layers in Internet-of-Things (IoTs) Framework," *Future Generation Computer Systems*, vol. 108, pp. 909–920, 2020.

[2] G. Kambourakis, C. Kolias, and A. Stavrou, "The Mirai Botnet and the IoT Zombie Armies," in *Proc. of the IEEE MILCOM*, 2017, pp. 267–272.

[3] H. Somaya and M. Tomader, "Tuning the Hyperparameters for Supervised Machine Learning Classification, to Optimize Detection of IoT Botnet," in *Proc. of the 11th International Symposium on Signal, Image, Video and Communications (ISIVC)*, 2022, pp. 1–6.

[4] G. V. Arbex, K. G. Machado, M. Nogueira, D. M. Batista, and R. Hirata, "IoT DDoS Detection Based on Stream Learning," in *Proc. of the 12th International Conference on Network of the Future (NoF)*, 2021, pp. 1–8.

[5] A. L. Suárez-Cetrulo, D. Quintana, and A. Cervantes, "A Survey on Machine Learning for Recurring Concept Drifting Data Streams," *Expert Systems with Applications*, vol. 213, p. 118934, 2023.

[6] P. Domingos and G. Hulten, "Mining High-Speed Data Streams," in *Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, p. 71–80.

[7] P. Horchulhack, E. K. Viegas, and M. A. Lopez, "A Stream Learning Intrusion Detection System for Concept Drifting Network Traffic," in *Proc. of the 6th CSNet*, 2022, pp. 1–7.

[8] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble Learning for Data Stream Analysis: A Survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[9] N. KoronIoTis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[10] "GitHub - ketlymachado/IoT-IDS-EFDT-DDoS: Codes and other files related to the proposed network intrusion detection system that uses a stream learning algorithm to detect DDoS attacks in IoT," 2020, Accessed on August 1, 2023. [Online]. Available: https://github.com/ketlymachado/IoT-IDS-EFDT-DDoS

[11] D. G. Corrêa, F. Enembreck, and C. N. Silla, "An Investigation of the Hoeffding Adaptive Tree for the Problem of Network Intrusion Detection," in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 4065–4072.

[12] A. Bifet and R. Gavaldà, "Learning from Time-Changing Data with Adaptive Windowing," in *Proc. of the SIAM International Conference on Data Mining (SDM)*, 2007, pp. 443–448.

[13] G. A. Da Silva Oliveira, P. S. S. Lima, F. Kon, R. Terada, D. M. Batista, R. Hirata, and M. Hamdan, "A Stacked Ensemble Classifier for an Intrusion Detection System in the Edge of IoT and IIoT Networks," in *Proc. of the 14th IEEE LATINCOM*, 2022, pp. 1–6.

[14] H. M. Gomes, J. Read, and A. Bifet, "Streaming Random Patches for Evolving Data Stream Classification," in *Proc. of the IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 240–249.

[15] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, and T. Abdessalem, "Adaptive Random Forests for Evolving Data Stream Classification," *Machine Learning*, vol. 106, p. 1469–1495, 2017.

[16] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. d. Hartog, "ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets," *IEEE IoT Journal*, vol. 9, no. 1, pp. 485–496, 2022.

[17] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[18] E. M. d. Elias, V. S. Carriel, G. W. De Oliveira, A. L. Dos Santos, M. Nogueira, R. H. Junior, and D. M. Batista, "A Hybrid CNN-LSTM Model for IIoT Edge Privacy-Aware Intrusion Detection," in *Proc. of the 14th IEEE LATINCOM*, 2022, pp. 1–6.

[19] "GitHub - online-ml/river: Online machine learning in Python," 2023, Accessed on August 1, 2023. [Online]. Available: https://github.com/online-ml/river

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2017, Accessed on August 3, 2023. [Online]. Available: https://arxiv.org/abs/1412.6980

[21] A. Pesaranghader, H. L. Viktor, and E. Paquet, "McDiarmid Drift Detection Methods for Evolving Data Streams," in *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–9.