Enhancing training time and sustainability in Intrusion Detection Systems on Machine Learning

Isabella Marasco[1,*,†], Karina Chichifoi[1,†], Silvio Russo[1,†] and Claudio Zanasi[1,†]

[1]Department of Computer Science and Engineering, University of Bologna, Italy

Abstract

The increasing complexity and volume of network traffic in the Internet of Things (IoT) environments, coupled with the rapid evolution of cyber threats, has rendered traditional Intrusion Detection Systems (IDS) less effective. In response, there is an urgent need to develop a more efficient IDS that can not only detect a wider range of attacks but also adapt quickly to new, previously unknown threats. This study addresses the issues of prolonged training times and high computational resource consumption in IDS, with a particular focus on achieving sustainability without compromising performance.

We put forth a solution to reduce the intrusion detection pipeline, with an emphasis on training time, that employs a novel machine learning (ML) model, PerpetualBooster, which has not previously been utilized in cybersecurity. This model is designed to minimize training times and computational resource consumption while maintaining high detection performance. The CIC Modbus dataset was used to evaluate the performance of our approach. PerpetualBooster was trained in a few seconds, demonstrating a reduction in training time compared to other ML algorithms. These results illustrate the potential of the proposed model as a sustainable, high-performance solution for real-time and energy-efficient IDS in IoT environments, addressing critical challenges in both cybersecurity and environmental sustainability.

Keywords

Cybersecurity, Machine Learning, Intrusion Detection System

1. Introduction

The rapid increase in network traffic and the growing sophistication of cyber threats, particularly in Internet of Things (IoT) environments, have revealed significant limitations in existing intrusion detection systems (IDS). While traditional IDSs demonstrate efficacy in identifying known threats, they exhibit difficulty in adapting to novel attacks [1]. Additionally, they often require substantial computational resources, raising concerns about their environmental impact.

The state of the art on IDS focuses on achieving high performance and reducing the number of false positives [2] and false negatives [3] generated by these systems. A plethora of research is being conducted with the objective of enhancing the efficiency and accuracy of IDS [4, 5, 6]. However, it is crucial to prioritize not only the performance of the system but also the reduction of training time and the minimization of computational resources, although a variety of approaches have been proposed in this regard in the literature [7, 8, 9]. Nevertheless, the various solutions continue to exhibit constraints with respect to training time and the consumption of computing resources. This is particularly relevant in the context of sustainability, Schwartz et al. [10] highlight that the increasing complexity of Machine Learning (ML) models amplifies their energy consumption. In cybersecurity, where new attacks frequently emerge, retraining these models is necessary to ensure it remains effective in detecting emerging and previously unseen threats. The delay in training time presents a significant challenge, as it hinders the possibility of rapidly deploying new models adapted for the new threat landscape, thereby limiting the ability to provide real-time protection against emerging attacks.

Our aim is to design a solution that can reduce the detection time of new attacks. To achieve this,

Joint National Conference on Cybersecurity (ITASEC SERICS 2025), February 03-8, 2025, Bologna, IT

*Corresponding author.

†These authors contributed equally.

isabella.marasco4@unibo.it (I. Marasco); karina.chichifoi@unibo.it (K. Chichifoi); silvio.russo3@unibo.it (S. Russo); claudio.zanasi4@unibo.it (C. Zanasi)

we are focusing on an intrusion detection pipeline, with a particular focus on reducing the training time of the ML model using PerpetualBooster, a model that has never been used in cybersecurity. In contrast to alternative models, PerpetualBooster does not require hyperparameter optimization, which enhances both operational efficiency and sustainability by reducing environmental impact. The proposed methodology was evaluated on the CIC Modbus dataset [11] and was compared with other models often used in this context including Gradient Boosting, LightGBM, HistGradientBoosting, Random Forest, Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP) in terms of training time and performance. The results demonstrate that our proposed approach is able to reduce training time in comparison to other models while maintaining high performance in the detection of attacks. This is due to the absence of hyperparameter optimization, which not only accelerates adaptation to new attacks in networks, but also reduces environmental impact and resource consumption, while maintaining its efficacy in detecting malicious activities.

The remainder of this paper is organized as follows. Section 2 presents an overview of the state of the art. Section 3 describes the data set and the pre-processing applied. Section 4 presents our proposed methodology. Section 5 illustrates the results. Finally, Section 6 summarizes the conclusions and discusses future work.

2. Related work

The growing number of cyber-attacks underscores the critical need to prevent and detect intrusions in networks. The application of AI has emerged as a highly sought-after solution due to its ability to adapt to evolving threats.

Shahid et al. [12] proposed a hybrid IDS tailored for IoT networks using the ROUT-4-2023 dataset, targeting RPL-based routing attacks. Their system combines statistical traffic analysis with ML and deep learning, achieving 99% accuracy with Random Forest and 97% with a Transformers-based model. Alazzam et al. [13] presented a dual-subsystem IDS trained with OCSVM on normal and attack packets, respectively. The system demonstrated superior performance on KDDCUP-99, NSL-KDD, and UNSW-NB15 datasets in detection rate, accuracy, and false alarms. Hu et al. [14] introduced Graph2vec+RF, leveraging graph embeddings for early detection. By generating flow graphs from initial network packets and classifying them with Random Forest, this method avoids extensive feature engineering and dataset size requirements, outperforming benchmarks on CICIDS2017 and CICIDS2018. While the research in this field aims to enhance the performance of IDS, this is no longer a sufficient goal. The impact of digital technologies on the environment is no longer negligible and must be considered, along with performance, as a primary parameter for evaluating a model. Our work prioritizes the reduction of training time without compromising high performance. This approach addresses a critical bottleneck in IDS development, enabling faster deployment and adaptability in dynamic cybersecurity environments.

Given the dynamic nature of events and the continuous stream of data, it is imperative to not only improve detection accuracy, but also to minimize training time. The authors in [8] proposed LIO-IDS, combining LSTM with an improved One-vs-One (I-OVO) technique. Evaluated on NSL-KDD, CIDDS-001 and CICIDS2017 datasets, it improved detection accuracy and reduced training time to as low as 153.25 seconds on CICIDS2017. Kim et al. [15] presented a hybrid IDS using a C4.5 decision tree and one-class SVMs. Tested on NSL-KDD, it showed better detection rates, fewer false positives, and reduced training time from 76.63 to 56.58 seconds. Gupta et al. [16] introduced CSE-IDS, a cost-sensitive deep learning and ensemble-based NIDS. It achieved competitive accuracy and reduced training times to 120 seconds on NSL-KDD and 430 seconds on CICIDS2017, outperforming traditional methods.

These approaches typically entail the combination of various techniques to reduce training time, which necessitates the regularization of hyperparameters to optimize performance. This, in turn, results in an increase in the time required for training. In contrast, our approach employs PerpetualBooster, a novel method for cybersecurity that has not previously been utilized in this context. This method eliminates the necessity for hyperparameter optimization, enabling a significant reduction in training time without compromising accuracy or other performance metrics. Furthermore, our methodology is

designed to reduce the consumption of computational resources, thereby enhancing the sustainability of IDS.

3. Pre-processing of the dataset

This section presents a description of the dataset, then proceeds to detail the pre-processing and feature selection phases.

3.1. Dataset

The dataset utilized in this research is the CIC Modbus Dataset [11], which was published by the Canadian Institute for Cybersecurity. It was generated from generated traffic captured through Wireshark that contains benign and malicious packets. The benign traffic represents legitimate Modbus communications within the substation network. The malicious network traffic emulates various types of Modbus protocol attacks that are based on the MITRE ICS ATT&CK framework. The original dataset includes traffic related to a series of protocols, such as TCP, Modbus, RMI, ARP, DNS, ICMP, and IGMP, but only those related to Modbus contain labeled samples of benign and malicious traffic. For this reason, we decide to consider only the Modbus network traffic that has two research advantages: it is the most recent publicly accessible dataset on this subject; there are no previous studies on IDS that utilized this dataset.

3.2. Pre-processing and feature selection

The initial step was the pre-processing phase. Due to the significant imbalance in the dataset, as shown in Table 1, we achieve a more balanced distribution through a random undersampling technique. This method involves randomly removing a subset of samples from the original dataset.

Table 1
Number of samples for each label class

Class	Number of samples
Benign	8.481.292
Attacks	506.761

To further optimize the pre-processing pipeline, we initially encode the categorical features using the OrdinalEncoder, thereby ensuring that categorical variables are converted into numerical values that ML algorithms can effectively interpret. Then, to enhance the performance of the ML models, we apply the min-max scaling that standardizes the numerical features. The dataset is then split into training and testing subsets with an 80-20 ratio to validate the model's performance on unseen data effectively.

In addition to these pre-processing steps, we incorporate *NeuroFS* [17] for feature selection. NeuroFS is a sparse neural network that is specifically designed to be resource efficient while maintaining high performance. This approach, proposed by Atashgahi et al. [17], dynamically updates the input neurons of the network to identify a set of relevant features from the given input data. Initially, the sparse connectivity of the network is generated randomly as an Erdos-Renyi random graph. This graph theory-based approach ensures that the network starts with a sparsely connected structure, which is critical for maintaining resource efficiency and reducing computational overhead. The dynamic adjustment of input neurons in NeuroFS allows the network to focus on the most pertinent features, thereby enhancing the model's ability to learn and generalize from the data.

During the training phase, the network undergoes a dynamic modification process in which input neurons are gradually removed based on their activity levels. Neurons that remain inactive are systematically pruned from the network, thus reducing the overall complexity. Inactive neurons that are deemed necessary based on certain criteria are re-added to the network, thus ensuring that the model retains

the ability to adapt and learn effectively. The selection process for reactivating neurons is governed by the connections with the highest gradient magnitudes. This criterion ensures that only the most significant neurons, in terms of their contribution to the network's learning process, are retained and emphasized. The gradient magnitude serves as a reliable indicator of the importance of each connection, guiding the dynamic adjustment of the network structure. Upon completion of the training phase, the network identifies K features corresponding to the active neurons with the highest strength in the input layer. These features are considered the most informative and relevant within the dataset, representing the key attributes that significantly contribute to the predictive power of the model. The features in Table 2 are selected using NeuroFS on our dataset.

Feature Name	Feature Description
flow	It indicates if the packet was transmitted or received
modbus_pdu	It specifies the modbus function code
src_port	It denotes the port number utilized by the sender host
dst_port	It indicates the number utilized by the destination host
ip_len	IP packet length
ip_chksum	It checks for errors in the IP packet
modbus_len	Length of modbus package
modbus_start_addr	The address of the initial element to be read
modbus_output_addr	Address of the coil to be turned on or off
modbus_output_value	Value that is overwritten if you turn the coil on (1) or off (0)
modbus_byte_count	Size in bytes of the response
modbus_coil_status_0	Number of states off
modbus_register_val	Sum of the values of all read registers
modbus_register_addr	Address of the request and response packet log to be overwritten
attack	The value can be 0 or 1. A value of 1 signifies an attack, whereas a value of 0 denotes benign network traffic. It is the label

Table 2
NeuroFS feature selection

Figure 1 shows the correlation matrix of the features selected by NeuroFS. This matrix offers an interesting visual representation of the linear relationships between the chosen variables, and the correlations through a heatmap where darker means more correlated. From this matrix, it can be inferred that the features ip_chksum, modbu_register_val, modbus_byte_count and modbus_len are the most correlated with the label attack, and thus the most relevant to the model.

4. Model

This section introduces PerpetualBooster [18], a recent ML model that was proposed in 2024. This model is designed to achieve high performance with minimal usage of resources but it has not been utilized in cybersecurity. PerpetualBooster is a gradient boosting machine (GBM) algorithm that does not require tuning of hyperparameters, thus obviating the need for hyperparameter optimization. This distinctive feature sets distinguishes it from other GBM algorithms [19] that are supervised learning algorithm that combine multiple weak learners into an ensemble with good predictive performance. They are developed through an iterative process where weak learners, typically decision trees [20], are sequentially added to correct the errors of previous models. During each iteration, a new tree is trained to address the residual errors of the current ensemble. This optimization is gradient-based, where the gradients of a loss function guide the adjustments made by each new tree. The final predictions are obtained by aggregating the outputs of all the trees, with each tree's contribution moderated by a learning rate. This learning rate not only controls the influence of each tree, but also acts as a regularization mechanism to mitigate overfitting. To further prevent overfitting, additional regularization techniques are employed, such as limiting the depth of the trees, pruning, and imposing penalties on model complexity.

PerpetualBooster[21] is different from other gradient boosting-based models due to its lack of

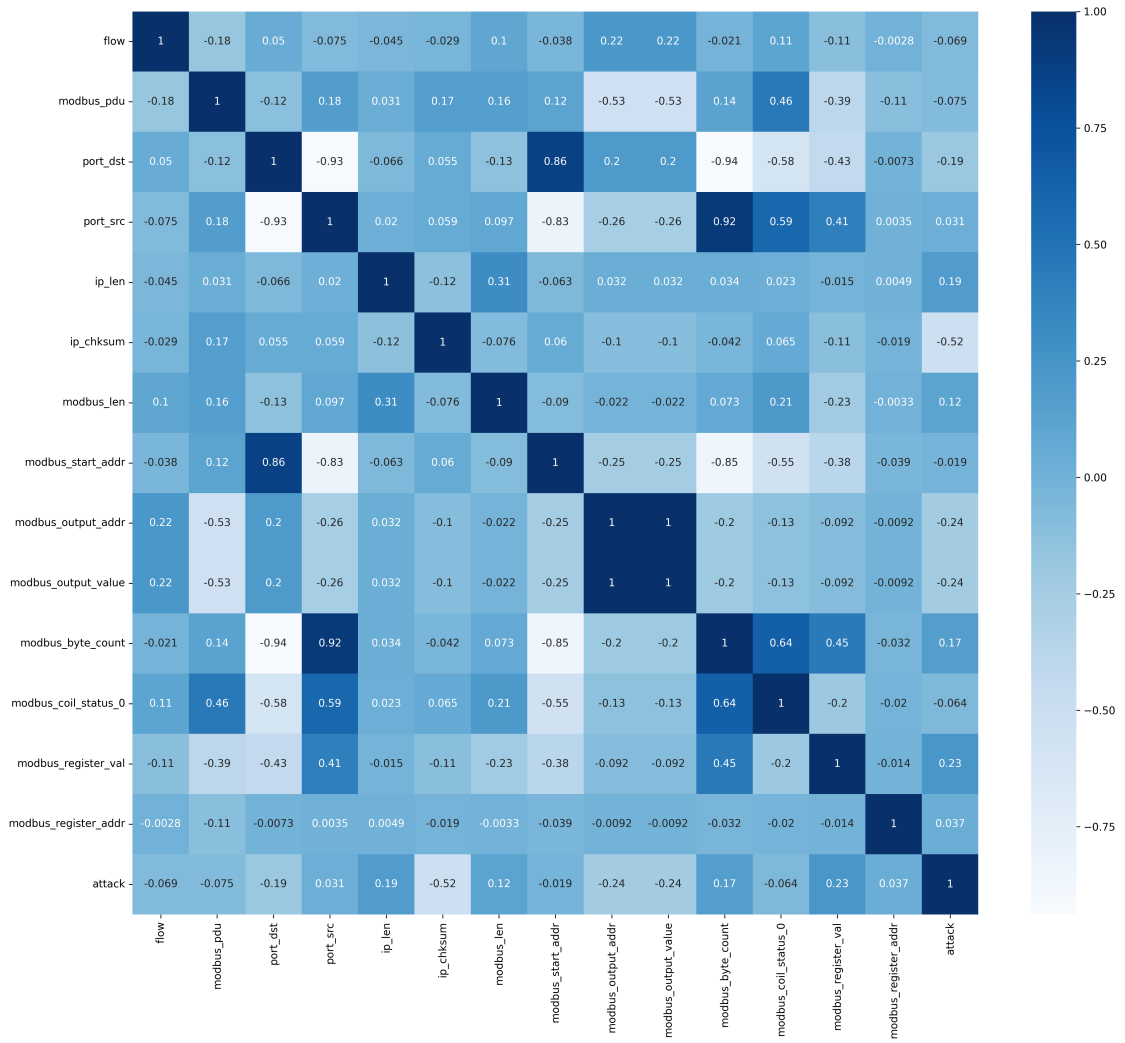

Figure 1: Correlation matrix of NeuroFS feature selection

requirement for hyperparameter optimization. This property is a consequence of a combined use of two techniques: step size control and generalization control. These techniques enable the balancing of data fitting and overfitting avoidance, which is a crucial aspect of decision tree learning.

4.1. Step size control

The step size control is based on the Armijo–Goldstein, also known as backtracking line search, that is a strategy used in optimization to ensure that the loss function f decreases sufficiently at each iteration. In each interation, the Armijo–Goldstein attemps to determinate a sufficiently step size α that satisfies the following condition:

$$f(x + \alpha p) \leq f(x) + \alpha cm$$

In this equation, x represents the current state, p denotes the descent direction, c controls the sufficiency of reducing the objective function and m is the direction derivative of the function f relative to the direction p at the point x.

In PerpetualBooster, the step size control has three differences regarding the backtracking line search. The first is that it tries to find the smallest step to satisfy the condition, instead of the largest. It grows the tree and checks the loss decrease after each split. If the loss decrease exceeds a certain threshold, it stops growing the tree and continues with the next boosting round. It tries to take the smallest step that achieves a target loss decrease at each boosting round. This can be called forward tracking tree search,

where α is calculated from the user-defined `budget` parameter, that can be a value between 0 and 1.

$$\alpha = 10^{-budget}$$

The second difference in step size control is the parameter m that is kept constant instead of updating at every step. It is calculated before the fitting process using the base score.

$$m = \frac{1}{n}\sum loss(y_i, \hat{y})$$

The last difference in PerpetualBooster during the step size control is the control parameter c that is calculated with the `bugdet` parameter, using the following formula:

$$r = \frac{10}{\text{budget}}$$

$$\text{Reciprocals of powers (ROF)} = \frac{r}{r-1}$$

$$\text{Truncated series sum (TSS)} = \text{ROF} - \left(1 + \frac{1}{r}\right)$$

$$c = \frac{1}{r} - \text{TSS}$$

Step size control is an efficacious strategy from the outset. The initial split significantly reduces the loss, which is particularly beneficial in the initial boosting rounds.

4.2. Generalization control

Before each node splitting, the data is split into training and validation sets. It calculates not only the training loss but also the validation loss using the separate validation set.

$$\text{loss}_{\text{train}} = G_{\text{train}} w + \frac{1}{2} H_{\text{train}} w^2$$

$$w = -\frac{G_{\text{train}}}{H_{\text{train}}}$$

$$\text{loss}_{\text{valid}} = G_{\text{valid}} w + \frac{1}{2} H_{\text{valid}} w^2$$

In these equations, G represents the gradient values, H the sum of the Hessian values, and w the weight term. Then, it calculates the generalization term using training and validating losses.

$$\text{Generalization} = \frac{\text{loss}_{\text{parent}} - \text{loss}_{\text{train}}}{\text{loss}_{\text{parent}} - \text{loss}_{\text{valid}}}$$

It lets the node split if generalization is greater than 1; it stops splitting if generalization is less than 1. In other words, it checks if the validation loss decreases compared to the parent loss when splitting.

The importance of generalization control increases as the boosting process progresses. As algorithmic learning progresses, the trees that result tend to become shallower because there is less data to inform the learning process. The algorithm has a built-in stopping mechanism that is triggered if it encounters a simple tree with poor generalization, less than 1, three times.

5. Experimental results

In this section we evaluate the effectiveness of PerpetualBooster in comparison to alternative models based on gradient boosting, including LightGBM [22], HistGradientBoosting [23] and GradientBoosting. The aim is to ascertain whether the novel gradient boosting-based model, PerpetualBooster, can attain with more rapid execution a superior performance in comparison to the extant models. Furthermore, we also compare PerpetualBooster with other models including Random Forest, SVM and MLP which, as observed in the related work section, typically achieve good performance. For each model, we utilize GridSearchCV in order to identify the optimal hyperparameter configuration.

The results are presented in Table 3. It shows a superiority of the PerpetualBooster in terms of training speed when compared to the other evaluated algorithms. PerpetualBooster completes the training process in a shortest time, equal to 7.39 seconds. The second fastest model, SVM, requires 191.33 seconds, which is approximately 26 times longer than PerpetualBooster. Other algorithms exhibit considerably longer processing times. Despite the considerable reduction in training time, the Perpetual Booster maintains high levels of accuracy and precision.

Model	Training time (s)	Accuracy (%)	Precision (%)
PerpetualBooster	**7.39**	**99.73**	**99.92**
LightGBM	1165.61	99.66	99.91
HistGradientBoosting	3852.0	99.66	99.94
GradientBoosting	43737.91	99.66	99.87
Random Forest	3663.95	99.72	99.91
SVM	191.33	99.51	99.91
MLP	6361.53	99.67	99.95

Table 3
Performance and training time of models

Furthermore, figure 2 shows that, in the absence of hyperparameters optimization, the training time of the models is decreased. In particular, the time required for LightGBM is marginally higher than that of PerpetualBooster. Nevertheless, the use of PerpetualBooster remains advantageous because of its capacity to achieve high results in a shorter time without the need to utilize hyperparameter optimization. Thanks to the elimination of the hyperparameter optimization, PerpetualBooster achieves good accuracy in a considerably shorter training time compared to other models.

6. Conclusions

This paper proposes a novel IDS designed to address the challenges posed by the evolving nature of cyber threats in industrial environments. We consider two main objectives: minimizing the time required for model training and reducing computational resource consumption. In order to achieve these goals, we propose the use of a novel ML model, PerpetualBooster, which has not previously been used in cybersecurity.

The experimental results demonstrate that PerpetualBooster enhances both the efficiency and adaptability of the IDS. It leads to a substantial reduction in training time when compared to conventional ML models, thereby enabling the system to rapidly adapt to new and emerging cyber threats. PerpetualBooster exhibits a marked reduction in computational resource consumption. The proposed IDS employs PerpetualBooster to reduce training times, lower resource consumption, and minimize computational impact while maintaining high efficacy in detecting malicious activities. This work contributes to the field of sustainable cybersecurity and establishes a foundation for future research into the optimization of machine learning models for real-time, energy-efficient, and adaptive intrusion detection systems even in cyberphysical environments.

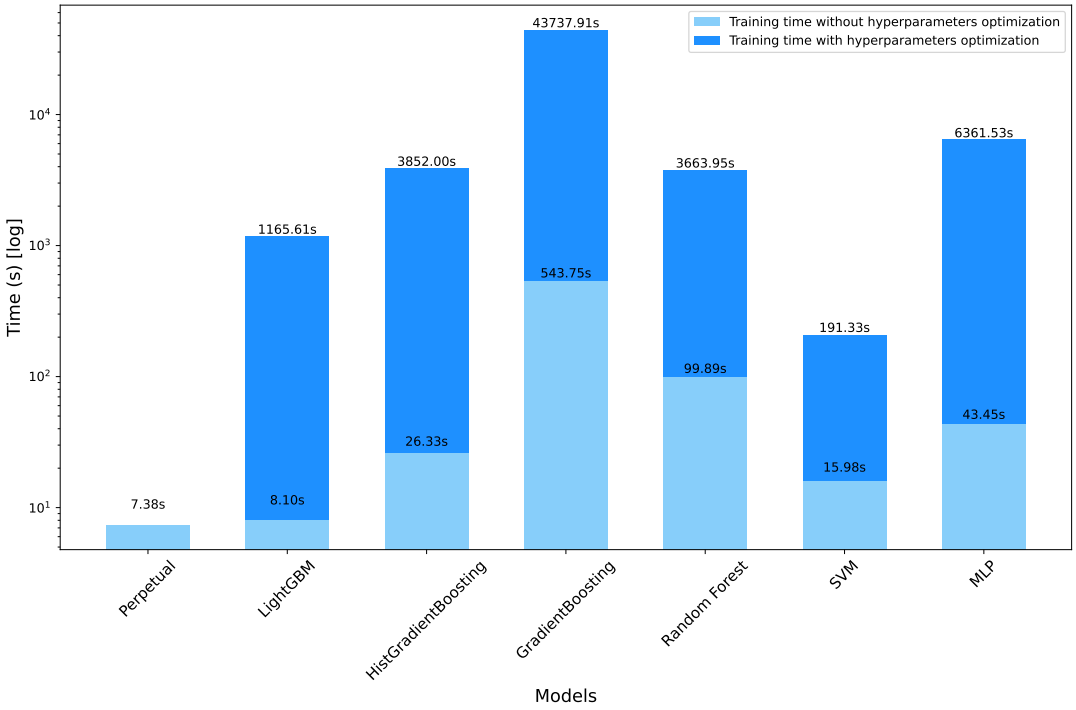

Figure 2: Training time with and without hyperparameter optimization

Acknowledgments

This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU and by the project C4SI funded by the PR-FESR ER 2021-2027

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Cantelli-Forti, M. Colajanni, Adversarial fingerprinting of cyber attacks based on stateful honeypots, in: 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018, pp. 19–24. doi:10.1109/CSCI46756.2018.00012.

[2] J. Liu, Y. Gao, F. Hu, A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm, Computers & Security 106 (2021) 102289.

[3] M. AlSlaiman, M. I. Salman, M. M. Saleh, B. Wang, Enhancing false negative and positive rates for efficient insider threat detection, Computers & Security 126 (2023) 103066. URL: https://www.sciencedirect.com/science/article/pii/S0167404822004588. doi:https://doi.org/10.1016/j.cose.2022.103066.

[4] M. Sajid, K. R. Malik, A. Almogren, T. S. Malik, A. H. Khan, J. Tanveer, A. U. Rehman, Enhancing intrusion detection: a hybrid machine and deep learning approach, Journal of Cloud Computing 13 (2024) 123.

[5] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, J. Ahmad, Enhancing iot network security through deep learning-powered intrusion detection system, Internet of Things 24 (2023) 100936.

[6] O. H. Abdulganiyu, T. Ait Tchakoucht, Y. K. Saheed, A systematic literature review for network intrusion detection system (ids), International journal of information security 22 (2023) 1125–1162.
[7] J. Gu, S. Lu, An effective intrusion detection approach using svm with naïve bayes feature embedding, Computers & Security 103 (2021) 102158.
[8] N. Gupta, V. Jindal, P. Bedi, Lio-ids: Handling class imbalance using lstm and improved one-vs-one technique in intrusion detection system, Computer Networks 192 (2021) 108076. URL: https://www.sciencedirect.com/science/article/pii/S1389128621001675. doi:https://doi.org/10.1016/j.comnet.2021.108076.
[9] M. J. Awan, U. Farooq, H. M. A. Babar, A. Yasin, H. Nobanee, M. Hussain, O. Hakeem, A. M. Zain, Real-time ddos attack detection system using big data approach, Sustainability 13 (2021) 10743.
[10] R. Schwartz, J. Dodge, N. A. Smith, O. Etzioni, Green ai, Commun. ACM 63 (2020) 54–63. URL: https://doi.org/10.1145/3381831. doi:10.1145/3381831.
[11] K. Boakye-Boateng, A. A. Ghorbani, A. Lashkari, Securing substations with trust, risk posture, and multi-agent systems: A comprehensive approach, in: 2023 20th Annual International Conference on Privacy, Security and Trust (PST), IEEE Computer Society, Los Alamitos, CA, USA, 2023. URL: https://doi.ieeecomputersociety.org/10.1109/PST58708.2023.10320154. doi:10.1109/PST58708.2023.10320154.
[12] U. Shahid, M. Z. Hussain, M. Z. Hasan, A. Haider, J. Ali, J. Altaf, Hybrid intrusion detection system for rpl iot networks using machine learning and deep learning, IEEE Access (2024).
[13] H. Alazzam, A. Sharieh, K. E. Sabri, A lightweight intelligent network intrusion detection system using ocsvm and pigeon inspired optimizer, Applied Intelligence 52 (2022) 3527–3544.
[14] X. Hu, W. Gao, G. Cheng, R. Li, Y. Zhou, H. Wu, Toward early and accurate network intrusion detection using graph embedding, IEEE Transactions on Information Forensics and Security 18 (2023) 5817–5831. doi:10.1109/TIFS.2023.3318960.
[15] G. Kim, S. Lee, S. Kim, A novel hybrid intrusion detection method integrating anomaly detection with misuse detection, Expert Systems with Applications 41 (2014) 1690–1700. URL: https://www.sciencedirect.com/science/article/pii/S0957417413006878. doi:https://doi.org/10.1016/j.eswa.2013.08.066.
[16] N. Gupta, V. Jindal, P. Bedi, Cse-ids: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems, Computers & Security 112 (2022) 102499. URL: https://www.sciencedirect.com/science/article/pii/S0167404821003230. doi:https://doi.org/10.1016/j.cose.2021.102499.
[17] Z. Atashgahi, X. Zhang, N. Kichler, S. Liu, L. Yin, M. Pechenizkiy, R. Veldhuis, D. C. Mocanu, Supervised feature selection with neuron evolution in sparse neural networks, arXiv preprint arXiv:2303.07200 (2023).
[18] PerpetualML, Perpetualbooster, 2024. URL: https://github.com/perpetual-ml/perpetual?tab=readme-ov-file.
[19] H. Lu, R. Mazumder, Randomized gradient boosting machine, SIAM Journal on Optimization 30 (2020). doi:10.1137/18M1223277.
[20] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, S.-I. Lee, From local explanations to global understanding with explainable ai for trees, Nature machine intelligence 2 (2020) 56–67.
[21] P. ML, How perpetualbooster works, 2024. URL: https://perpetual-ml.com/blog/how-perpetual-works.
[22] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, Advances in Neural Information Processing Systems 30 (2017).
[23] H. A. Ali, C. Mohamed, B. Abdelhamid, N. Ourdani, T. E. Alami, A comparative evaluation use bagging and boosting ensemble classifiers, in: 2022 International Conference on Intelligent Systems and Computer Vision (ISCV), 2022, pp. 1–6. doi:10.1109/ISCV54655.2022.9806080.