Automated Machine Learning Configuration to Learn Intrusion Detectors on Attack-Free Datasets

Omar Anser, Jérôme François, Isabelle Chrisment

2024 IEEE 49th Conference on Local Computer Networks (LCN) | 979-8-3503-8800-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/LCN60385.2024.10639690

Automated Machine Learning Configuration to Learn Intrusion Detectors on Attack-Free Datasets

Omar Anser*, Jérôme François††, Isabelle Chrisment*
*Inria, Université de Lorraine, CNRS, LORIA, Nancy, France
†SnT - University of Luxembourg, Luxembourg
Emails: firstname.lastname@inria.fr, firstname.lastname@uni.lu

***Abstract*—Intrusion detection systems have benefited from Machine Learning (ML) to alleviate the problem of building and maintaining accurate signatures. Nevertheless, ML solutions face issues like overfitting or insufficient training data, which may necessitate retraining or adjustments to maintain long-term efficiency. From data collection to model training, all efforts are crucial for deploying a robust ML-based intrusion detector. Among these efforts, optimizing model hyperparameters, a time-consuming task, can be automated by existing methods.**

Yet, such methods require a validation set, making them unsuitable for training a detector on an attack-free dataset, as in anomaly-based intrusion detection. Additionally, setting the anomaly detectors' threshold, usually beyond hyperparameters configuration, requires knowledge of attacks. To overcome these challenges, this paper presents an automated solution to infer the hyperparameters and the threshold jointly from an attack-free training dataset. Pre-learned optimal configurations are transferred and fine-tuned across datasets.

Our method minimally impacts model accuracy detection performance (4% degradation), while dramatically reducing configuration time by a factor of 160 across the IDS2017 and IDS2018 datasets.

I. Introduction

Machine Learning (ML) has become crucial for operational security, as noted by MIT Technology Review [1], and demonstrated by the increasing number of ML-based Intrusion Detection Systems (IDS) explored in recent studies [2]. This paper focuses on network-based IDS, designed to detect attacks by monitoring network traffic.

Although they are effective in tests with predefined datasets, deploying these systems in real-world environments remains challenging [3]. Issues like overfitting or insufficient data complicate the training of efficient detectors capable of adapting to various environments with different data profiles. As a result, re-adapting or retraining the ML models is essential [4], [5].

Configuring hyperparameters (HPs), along with choosing algorithms, selecting features, and preprocessing data are crucial preliminary tasks in the re-adaptation of ML models. Related works [6], [7], [8] emphasize the importance of regularly reconfiguring HPs and updating the ML model to enhance its robustness. The HPs setting can be automated using Hyper-Parameter Optimization (HPO) methods such as grid search and Bayesian Optimization (BO) but their iterative and time-consuming nature prevents frequent reconfigurations [9].

Furthermore, determining the optimal configuration through HPO requires a validation set with both benign and attack samples [10]. This makes it impractical in the context of Anomaly Detection (AD) [11], in which AD-based Intrusion Detection Systems (AD IDS) are trained on attack-free datasets.

To highlight the main challenges of applying common HPO to AD IDS, a naive supervised configuration approach is illustrated in Figure 1. This approach divides the dataset into three sets (K-folding can also be used). A configuration loop utilizes HPO to derive a configuration ①, which is then employed in the AD IDS model training (with the training set) ② and in validation (with the validation set) ③. At the end of the loop, the AD IDS's final performance is assessed with a test set and is indicative of its real-world effectiveness.

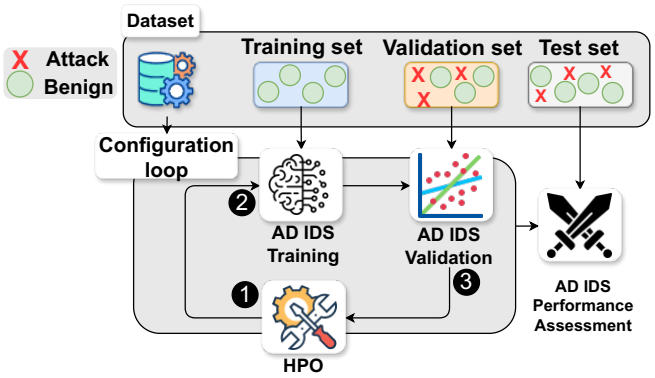

Fig. 1. Supervised HPO for AD IDS

Thus, Figure 1's workflow pose two major challenges:

C1—Necessity of validation set in the configuration loop. The AD IDS training uses only benign samples, but, as previously mentioned, the configuration process (including HPO) assumes a pre-existing validation set with attack samples. This is impractical for regular reconfigurations, as it requires knowledge of future attacks that are inherently supposed to be unknown in the context of AD.

C2—AD IDS threshold determination. Commonly, AD IDS performance is assessed using the comprehensive metric AUC ROC (Area Under the Receiver Operating Characteristic Curve), calculated by varying threshold values and comparing them to the ML model's anomaly scores. A higher AUC ROC signifies enhanced robustness of the AD IDS against threshold variations. Yet, in deployment, a specific value must be set. Many techniques rely on static or predefined thresholds [12],

[13], [14], which often struggle to adapt to dynamic data, thereby leading to scalability and robustness issues [15].

To address **C1**, this paper introduces a meta-learning-based, semi-supervised configuration approach that relies solely on the training set to infer an efficient configuration. Unlike other limited straightforward methods, such as consistently employing a single overall configuration or the direct transfer of an optimal configuration from one dataset to another, our strategy leverages a meta-learning (MtL) approach to enhance the transfer of configurations between datasets in a more contextualized manner. It is capable of deriving a new configuration instead of merely reusing preexisting ones. This approach is inspired by [6], which reduces HPO computational time in supervised scenarios but remains incompatible with **C1**.

In addressing **C2**, our MtL configuration method identifies a configuration that increase the robustness of an AD IDS. Subsequently, it automatically fine-tunes the threshold as part of the overall configuration process, jointly with HPs configuration.

The paper's contributions are summarized as follows:

- A MtL solution to automate the configuration of a robust AD IDS without anticipating the types of attacks to be detected (Section III).
- An extension of this solution to automatically infer an efficient threshold for the AD IDS (Section III).
- A comparative assessment of the proposed method versus the optimal case (supervised HPO). This evaluation demonstrates that our MtL approach closely matches HPO in terms of attack detection with an average difference of less than 4%, but is 160 times faster (Section. V).

II. Related Works

A. AD IDS Configuration

AD IDS configuration focuses on optimizing the model HPs and determining a threshold to effectively differentiate between normal and anomalous behavior. In the literature, this is predominantly done using either manual techniques or conventional HPO methods, often involving the use of a validation set [16], [17].

The authors in [16] present an unsupervised ensemble learning strategy for Industrial IDS that requires benign data in its training process. They propose three configuration methods: a conventional optimal method that needs a validation dataset, a suboptimal manual method, and a manual method that integrates temporal information into the ensemble model—both manual methods do not require validation sets. Although this approach yields satisfactory results, it remains manual and potentially prone to errors.

In the paper [18], the authors introduce a method that refines HPs tuning for the Local Outlier Factor (LOF) model through a statistical technique. This involves conducting a grid search to determine the optimal neighborhood size and contamination parameters. Despite providing a solution based solely on training data for configuration, it is still an iterative process using a pre-set grid, which is time-consuming. Furthermore, this approach requires manual threshold setting.

B. Dynamic Thresholding in AD IDS

Several studies focused on setting the threshold in an AD system, whether it is specific to the application of an IDS or more general. They relied on static methods [19], [20], [21], [22], game theory-based approaches [23], or RL [24].

The papers [25] and [26] both address the AD thresholding problem, where [25] models the problem as a Markov Decision Process, introducing an agent-based dynamic thresholding framework using a deep Q-network, while [26] focuses on AD IDS and employs a mathematical approach to link threshold values to alert frequencies, offering a practical threshold-setting algorithm for dynamic data distributions.

The work in [27] evaluates three advanced deep AD models [28], [29], [30], all utilizing dynamic thresholding. [30] evaluates the current prediction error by using a vector of historical errors, combining an exponential-weighted average and the argmax function. [29] uses a Variational Autoencoder (VAE) with extreme value theory principles to determine the threshold, employing maximum likelihood estimation on the tail probability without assuming any specific data distribution. [28] employs a state-based thresholding technique by combining Long Short-Term Memory and VAE.

This paper [31] proposes an adaptive thresholding method for AD in univariate time series that leverages segmentation and local statistical analysis to dynamically adjust thresholds. This method simplifies time series data into manageable segments, within which it calculates local means and standard deviations to set context-sensitive thresholds.

In [32], the authors use Isolation Forest for online AD in a smart home network. They suggest auto-tuning Isolation Forest's sklearn implementation with RL to optimize the contamination threshold for classifying flows. Unlike our approach, their search space is limited to 11 contamination values, which raises questions about the need for RL in such a small space. Additionally, RL consistently returns nearly the same threshold value in their results, questioning the significance of reconfiguring iForest given the datasets used.

Our method sets itself apart from these mentioned solutions by incorporating the AD IDS's HPs, not just the threshold, during configuration.

C. Meta-Learning in AD IDS

MtL has multiple interpretations. This work adheres to the definition where a ML model learns from numerous datasets across several episodes [33]. This concept is applied in [34] where the authors propose its utilization for automatically select an appropriate anomaly detector algorithm to train. In contrast, in AD IDS literature, MtL is commonly associated with the use of multiple unsupervised algorithms to enhance intrusion detection [35], [36] or to address the challenge of training AD models with limited anomaly samples [37].

The authors in [38] leverage MtL to address high false-positive rates in AD. Rather than relying on manual verification

of top-ranked anomalies, this research integrates long-term performance modeling into the detection process. Using deep reinforcement learning, a meta-policy is developed to optimize query selection. The authors emphasize the meta-policy's ability to adapt to new datasets without re-tuning, attributing this flexibility to meta-learning.

To our knowledge, no previous studies in AD IDS have utilized this definition of MtL to configure and determine the threshold value without the use of a validation set.

III. Meta-Learning for AD IDS Configuration

A. AD IDS Configuration Problem

As depicted in Figure 1, an AD IDS employs a one-class ML technique that learns from normal samples, namely benign network traffic. In many cases, such as Isolation Forest used in this paper, the learned model can infer an anomaly score to be compared to a threshold, τ.

Configuring an AD IDS involves setting the ML algorithm's k HPs and τ in the configuration loop. This process serves two primary objectives:

1) **Enhance the Robustness of the AD IDS**: The aim is to make the AD IDS less sensitive as possible to the value of τ. This objective is defined as $p_{\text{robust}} : D \times C$ and commonly measured using the AUC ROC. p_{robust} assess the overall AD IDS performance applied to the dataset D within the hyperparameter search space C, independent of τ. The goal is thus to tune $c = (c_1, c_2, \ldots, c_k) \in C$ to find c^*, maximizing p_{robust}.
2) **Distinguish Attacks from Legitimate Traffic:** The aim is to maximize $p_{\text{classif}} : D \times C \times \tau \in [0, 1]$ which measures the capability of the AD IDS to correctly identify attack and benign samples. Common metrics for p_{classif} are the precision, recall or F1-score.

Reaching optimal detection score involves determining the optimal configuration by finding sequentially or concurrently both c^* and τ^*. Practically, the main challenge is to identify τ^*. However, also achieving optimal robustness in AD IDS by determining c^* reduces reliance on exact optimal threshold τ^*, easing its adjustment in a robust system.

B. A semi-supervized configuration approach

Figure 2 provides an overview of our approach, which consists of two phases: offline and online. These phases are designed to efficiently reconfigure the AD IDS in response to a new network context, such as changes in traffic profiles, including types of attacks, or alterations in network topology.

The high-level idea illustrated in Figure 2 is to transfer optimal configurations, determined offline, to the online phase via a meta-model. The online phase is the active period during which the AD IDS operates. Details of the meta-model's construction will be provided in Section III-C.

The entire workflow operates on three realistic assumptions. First, we assume that changes in network context are indirectly embedded in the generated data. For example, an increase in the number of hosts or modifications in network topology would not only change the volume but also impact the behavioral patterns of the observed network flows.

Second, we assume access to a collection of datasets that have been characterized offline, including attacks identification. This represents knowledge acquired from in-lab testing or post-deployment analysis. Consequently, as shown in Figure 2, the optimal configurations of each previous dataset, specifically c^* and τ^*, are obtained offline through the configuration loop (Fig. 1). This forms the basis of prior knowledge that feeds the meta-model, which is utilized online, replacing the conventional loop and avoiding time-consuming optimization iterations.

Third, when deploying the AD IDS online in a new network environment, a few attack-free traces can be provided as a new dataset, D_{new}. This is the reason why our configuration approach can be qualified as semi-supervised. Such hypothesis can hold in reality as it is easier for an administrator to white-label expected traffic (for example when deploying new services). In contrast, unexpected or anomalous traffic, which could include sophisticated attacks, is harder to accurately characterize due to its irregular and often covert nature. Thus, thanks to D_{new} and the meta-model, an effective configuration (c, τ) is directly inferred online. The AD IDS can now be retrained with (c, τ) and D_{new} to better adapt to the new network context.

C. Offline Phase: Meta-model Construction

In order to effectively transfer the optimal configurations, already calculated during the offline phase, to the online phase, a meta-model is used. Fig. 3 illustrates its construction.

Conceptually, the idea is to use the meta-model to integrate network contexts details during the transfer, as opposed to conducting an un-contextualized one. The aim is to transfer optimal configurations while taking into account the *characteristics* of both the source network contexts, where these configurations are originally effective, and the target network context, where efficient functioning of the AD IDS is required. It is thus a more informed transfer by channeling multiple optimal configurations alongside their respective network contexts.

Practically, as depicted in Figure 3, the meta-model's construction is initiated by using each previous dataset in ① to determine the corresponding optimal configurations ③, specifically c^* and τ^*, through the configuration loop. In parallel, the same datasets are individually characterized in ② using both simple and statistical measures, including measures like instances ratio, kurtosis, mean values, and standard deviations for each feature, which are commonly used in AutoML research [39] (details of these characteristics are provided in Table I). For each dataset, ② and ③ establish a one-to-one relationship, collectively forming new data points. These new data points are utilized to train a Random Forest Regressor as the meta-model ④, capable of inferring efficient c and τ from the characteristics of any new dataset in the online phase.

Optimally configuring an AD IDS in this phase involves a two-stage optimization process:

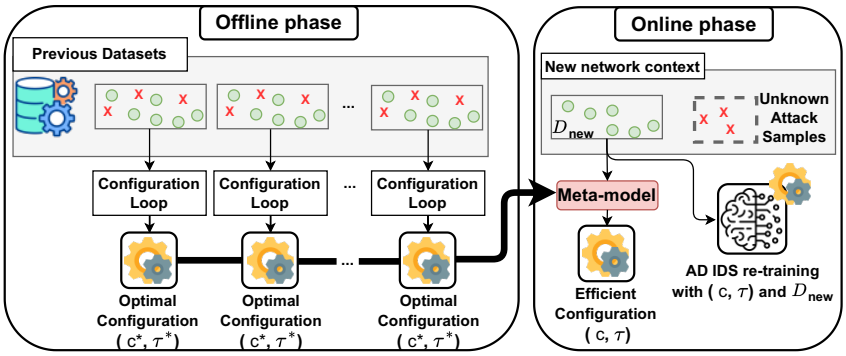

Fig. 2. Semi-supervised HP configuration for AD IDS

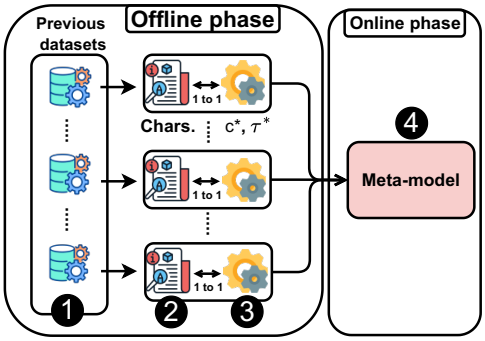

Fig. 3. Meta-model Construction

TABLE I
LIST OF SIMPLE AND STATISTICAL CHARACTERISTICS

Group	Characteristics
Simple	Ratio of features to instances, number of binary attributes, total instances.
Statistical	Geometric mean, harmonic mean, interquartile range, kurtosis, MAD, max/min values, mean, median, range, standard deviation, skewness, sparsity, trimmed mean, variance of each feature.

1) **Derive c^*:** c^* is calculated by maximizing p_{robust}, using BO as the HPO method. BO is an iterative, model-based black-box approach, known for robustness in AI [9]. BO selects the next c for validation based on prior results. It utilizes a surrogate model probabilistically estimates p_{robust} and iteratively refines the search space, unlike grid search's exhaustive exploration.
2) **Derive τ^*:** Once c^* set, τ^* is found through exhaustive testing to maximize the detection accuracy (p_{classif}), based on sample classification score.

During the online stage, where no attack sample is given, all c and τ are inferred jointly in contrast with most common solutions where threshold setting is a distinct task, typically done after optimizing HPs and learning algorithms [16].

D. Online Phase: Efficient Configuration Inference

When a new dataset, D_{new}, is provided online, the same characteristics as those extracted in the offline phase are derived solely from benign traffic. These characteristics are used as input for the pre-trained meta-model to immediately infer the associated efficient c and τ. In fact, these values are approximations of the optimal c^* and τ^* values which could have been calculated with the offline configuration loop if attacks samples were provided, as initially supposed in [6].

Finally, the ML model of the AD IDS is trained using c, with τ as the classification threshold.

IV. Datasets and Evaluation Method

A. Datasets

We leverage the IDS2017 and IDS2018 datasets [40], which are widely utilized benchmarks for IDS evaluation. These datasets mimic internet-based victim-attacker interactions using Windows and Linux systems. IDS2018 is built on IDS2017 by including new attacks and changing the test environment: IDS2018 relies on a virtual AWS setup with 7 networks and 500 machines, while IDS2017 features two networks and 14 hosts. Besides, we use a corrected version [41] where errors, and mislabeled traces have been corrected. Benign traffic, mainly HTTP and HTTPS, was collected through profiling agents trained on real network activities. The datasets are flow-based, segmented into days (5 days for IDS2017 and 10 days for IDS2018), and were created independently without temporal continuity. IDS2017's first day, solely benign traffic, is excluded.

Table II details the daily distribution of executed attacks and benign traffic, along with bidirectional flow counts. The number of flows per day ranges from 322K to 7.4M, with IDS2018 recording more flow counts than IDS2017 by up to tenfold. The two datasets feature 31 attacks, including DoS, DDoS, Botnet, and Bruteforce attacks, among others. Across the days, the attacks vary in percentage (0.01% for Web attacks to 47.30% for DDoS attacks), but the majority of flows remain benign, often exceeding 80%, which reflects realistic traffic patterns.

Since each day is different from all others based on the type of attacks or the topology used, a given day is representative of a unique network context in our evaluation. Two exceptions are in IDS2018, where Web attacks occur on both days 9 and 10, and Infiltration attacks are noted on days 11 and 12.

Based on the dataset authors' analysis [40] and commonly used features in IDS solutions [42], we have chosen 14 features from the original set of 83. These are detailed in Table III.

B. AD IDS design and implementation

Isolation Forest Although our approach is applicable to any algorithm used in AD IDS, our evaluations are based on the Isolation Forest algorithm, called iForest, demonstrated to be efficient in the context of AD [43]. It isolates outliers rather than profiling normal data by constructing binary decision trees (iTrees) from the training set. iTrees are formed by randomly selecting features and split values, creating partitions. A group of iTrees collectively forms an iForest. The path length to

TABLE II
DAILY ATTACK AND BENIGN TRAFFIC DISTRIBUTION

Day	Attacks	% Attacks	% Benign	#Flow
0	FTP, SSH Patator	2.16	97.84	322K
1	DoS GoldenEye, DoS Hulk, DoSSlowhttptest, DoS Slowloris, Heartbleed	35.74	64.26	497K
2	Infiltration, NMAP Portscan, Web Attacks	19.11	80.89	362K
3	Botnet Ares, DDoS-LOIC, Infiltration	47.30	52.70	548K
4	FTP, SSH Patator	4.88	95.12	5.9M
5	DoS GoldenEye, Slowloris	0.70	99.30	5.4M
6	DoS Hulk, FTP Patator	25.83	74.17	7.4M
7	DDoS-LOIC-HTTP, LOIC-UDP	4.79	95.21	6.1M
8	DDoS-HOIC, LOIC-UDP	15.57	84.43	7.0M
9	Web Attack: Brute Force, SQL Injection, XSS	0.01	99.99	6.1M
10	Web Attack: Brute Force, SQL Injection, XSS	0.01	99.99	6.0M
11	Infiltration: Communication Victim Attacker, Dropbox Download, NMAP	0.76	99.24	6.6M
12	Infiltration: Communication Victim Attacker, Dropbox Download, NMAP	0.61	99.39	6.6M
13	Botnet Ares	2.27	97.73	6.3M

isolate a data point in an iTree indicates its anomaly score. An overall score is then defined based on the iTrees which can be ultimately compared to the threshold τ to identify attacks. Extensions to the original algorithm have been proposed to isolate outliers faster and/or find clustered outliers [44], [45], which we leverage in our approach.

Configuration Search Space Based on existing work [45], [43], we focus on configuring seven HPs of iForest: the number of columns used for a split in the model; the size of the data subsets used for constructing each binary tree; the quantity of binary trees to be constructed for the model; the maximum growth depth for the binary trees; the probability of choosing a threshold for splitting a single variable or multiple variables; the minimum required gain threshold to proceed with a node splitting; and the number of variables or linear combinations to test for the best gain determination. Thus, C is the product of all these HPs.

TABLE III
FEATURES CONSIDERED IN EACH DATASET

Feature Name	Description
Src Port	Source service port numbers (one-hot encoded).
Dst Port	Destination service port numbers (one-hot encoded).
Protocol	Network protocol of the flow (one-hot encoded).
Flow Duration	Duration of the flow.
Total Fwd/Bwd Packets	Packets in the forward/backward direction.
Total Length of Fwd/Bwd Packet	Size of packets in forward/backward direction.
Flow Bytes/s	Bytes transferred per second.
Flow Packets/s	Packets transferred per second.
Fwd/Bwd IAT Total	Time between packets in fwd/bwd direction.
Fwd/Bwd Packets/s	Forward/backward packets per second.

Evaluation method The functions p_{robust} and p_{classif}, which determine the AUC ROC and F1-score, respectively, both apply a 5-fold cross-validation. For p_{robust}, the positive class is identified as attacks. Given the minimal number of attack instances on most dataset days, we opted for cross-validation to merge both validation and testing. In each validation stage, four folds are used for training while the fifth fold is combined with attack data for validation.

V. EVALUATION

We assess our method in terms of anomaly detection performance by determining p_{classif} with τ, thereby evaluating MtL's ability to directly infer the threshold value (**C2**). Due to space limitations, we omit the part where c is derived and the calculation of p_{robust}, focusing only on p_{classif}.

We compare our method to the optimal configuration using Bayesian Optimization (BO), an ideal scenario that requires attack traces. In this case, τ_i^* is determined by maximizing the F1 score after BO has been applied to the hyperparameters. As a reminder, τ is treated as a hyperparameter to be inferred with our solution. To configure each day of the datasets, our method uses data from the previous 13 days to build the meta-model, applying it in a rotating manner to yield 14 independent AD IDS configurations.

p_{classif} is evaluated using the F1 score and precision. While not presented, it is worth noting that recall is always above 0.97.

In terms of the F1 score shown in Figure 4, where the x-axis represents the days of the dataset as presented in Table II, we observe that the optimal configuration using BO ($p_{\text{classif}}(D_i, c_i^*, \tau_i^*)$) demonstrates high performance, with values ranging from 0.94 to 1. Although our method does not rely on a validation set with attack samples to infer τ, the F1 score is noticeably lower only on days 2 and 5-8. The worst result occurs on day 6, with an F1 score of 0.84 compared to 0.98 obtained with the optimal configuration.

Figure 5 highlights similar findings for precision, with MtL values generally lower than the optimal and exhibiting greater variability.

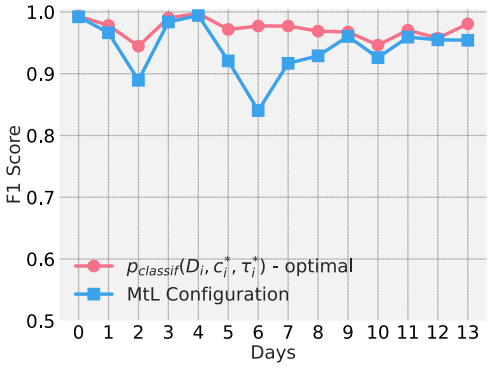

Fig. 4. F1 score with a daily configuration

Generally, our approach is almost equivalent to BO for both the F1-score and precision over 8 days (out of 14). It is slightly

less accurate on 2 days (days 8 and 13) and leads to noticeable degradation on only 4 days (2, 5, 6, and 7).

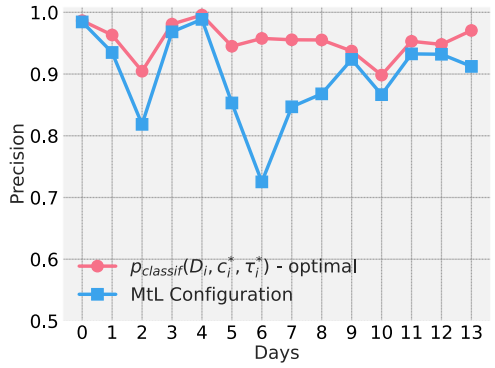

Fig. 5. Precision with a daily configuration

Setting τ is challenging because the optimal threshold τ_i^* tends to vary each day. For instance, τ_0^* is 0.31, while τ_4^* is 0.10. This variability makes it difficult to develop an efficient ML model capable of predicting it without knowledge of the attack traffic characteristics. This suggests that even if the inferred τ_i (MtL) differs from τ_i^* (optimal), MtL compensates by optimizing all HPs jointly, a task challenging to perform manually. Hence, configuration is an interplay of all HPs, whereas most work considers setting the threshold and the HPs of the ML algorithm in separate phases.

A. Configuration Generation Time

We also assessed, on a daily basis, the time it takes for MtL to generate configurations compared to BO.

For MtL, the time for learning the meta-model is completed offline, and thus the generation time for MtL is the time taken to infer c_i, including the extraction of the dataset's characteristics. Once done, the AD IDS requires training with the chosen configuration and application on the test set to identify malicious flows. This process contrasts with BO, where training and prediction run concurrently. Hence, for a fair comparison, we also include the time taken by MtL for training and testing the model in the online phase.

In Figure 6, our solution always returns a configuration in less than 1200 seconds, with extremely fast results on days 9 and 10 (less than 1 second). Conversely, c_i^* requires considerably more time, with durations varying from day to day and occasionally exceeding 100,000 seconds. Overall, MtL is faster by about a factor of 160 across all days.

B. Meta-learning Relative Performance

Figure 7 highlights the daily relative performance (gain) of MtL for p_{robust} (ROC AUC) and $p_{classif}$ (F1 score) compared to the performance achieved by BO when the number of BO iterations is reduced to match MtL's configuration generation time. Even without considering attacks for configuration, MtL performs as well as BO. Specifically, the gain does not show any major negative values and is even better in certain cases. In terms of p_{robust}, we note that on 5 out of 14 days, the gain is nearly or exceeds 0.25, while on 2 out of 14 days, it surpasses 0.5. For $p_{classif}$, , although the gains are typically lower, it's worth noting that for 4 out of 14 days, they approach or surpass 0.25. This demonstrates the superiority of our approach over BO at an equivalent computational cost and without requiring attack samples.

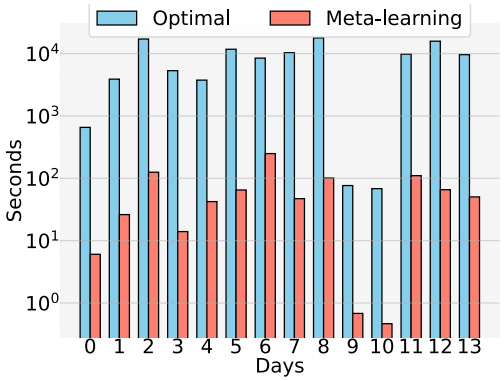

Fig. 6. Configuration Generation Time

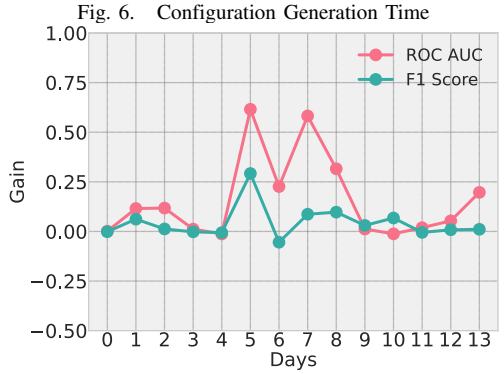

Fig. 7. Performance gain of semi-supervised MtL computed as the difference between the performance of MtL and BO

VI. Conclusion

This paper explores the problem of configuring an ML solution for an AD IDS. We tackle the problem of relying exclusively on an attack-free dataset to automate the configuration of the detector. This contrasts with known automated methods in the area of AutoML, which require a validation set with attack samples for HPO. To alleviate this need, we propose a semi-supervised MtL-based framework to learn a model from previous offline experiences where full knowledge is available. It is then capable of directly inferring a configuration based on benign samples only. This approach is more practical than hypothetically knowing attack details in advance. Compared to this baseline, the performance degradation is low and the method is significantly faster. For future work, we aim to enhance the entire pipeline with MtL, including algorithm selection.

VII. Acknowledgment

This work has been partially supported by the French National Research Agency under the France 2030 label (Superviz ANR-22-PECY-0008). The views reflected herein do not necessarily reflect the opinion of the French government.

REFERENCES

[1] I. F. Sridhar Muppidi and C. I. Security, "Ai in Cybersecurity: Yesterday's Promise, Today's Reality," MIT Technology Review, Tech. Rep., May 2023.

[2] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Communications Surveys Tutorials*, 2019.

[3] R. Xie, J. Cao, E. Dong, M. Xu, K. Sun, Q. Li, L. Shen, and M. Zhang, "Rosetta: Enabling robust TLS encrypted traffic classification in diverse network environments with TCP-Aware traffic augmentation," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[4] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[5] E. K. Viegas, A. O. Santin, V. V. Cogo, and V. Abreu, "Facing the unknown: A stream learning intrusion detection system for reliable model updates," in *International Conference on Advanced Information Networking and Applications*, 2020.

[6] O. Anser, J. François, and I. Chrisment, "Auto-tuning of hyper-parameters for detecting network intrusions via meta-learning," in *IEEE/IFIP Network Operations and Management Symposium*, 2023.

[7] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time url spam filtering service," in *2011 IEEE Symposium on Security and Privacy*, 2011.

[8] S. T. K. Jan, Q. Hao, T. Hu, J. Pu, S. Oswal, G. Wang, and B. Viswanath, "Throwing darts in the dark? detecting bots with limited data using neural data augmentation," *2020 IEEE Symposium on Security and Privacy*, 2020.

[9] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[10] M. A. Khan, N. Iqbal, Imran, H. Jamil, and D.-H. Kim, "An optimized ensemble prediction model using automl based on soft voting classifier for network intrusion detection," *Journal of Network and Computer Applications*, 2023.

[11] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv. 2019*, 2009.

[12] R. Bhaumik, C. Williams, B. Mobasher, and R. Burke, "Securing collaborative filtering against malicious attacks through anomaly detection," in *AAAI Conference on Artificial Intelligence*, 2006.

[13] S. Lin, R. Clark, R. Birke, S. Schönborn, N. Trigoni, and S. Roberts, "Anomaly detection for time series using vae-lstm hybrid model," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[14] M. Landauer, G. Höld, M. Wurzenberger, F. Skopik, and A. Rauber, "Iterative selection of categorical variables for log data anomaly detection," in *26th European Symposium on Research in Computer Security (ESORICS), 2021*, 2021.

[15] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[16] K. Wolsing, D. Kus, E. Wagner, J. Pennekamp, K. Wehrle, and M. Henze, "One ids is not enough! exploring ensemble learning for industrial intrusion detection," in *28th European Symposium on Research in Computer Security (ESORICS), 2023*.

[17] T. Madi, H. A. Alameddine, M. Pourzandi, A. Boukhtouta, M. Shoukry, and C. Assi, "Autoguard: A dual intelligence proactive anomaly detection at application-layer in 5g networks," in *26th European Symposium on Research in Computer Security (ESORICS), 2021*, 2021.

[18] Z. Xu, D. Kakde, and A. Chaudhuri, "Automatic hyperparameter tuning method for local outlier factor , with applications to anomaly detection," in *2019 IEEE International Conference on Big Data*, 2019.

[19] Y. Chae, N. Katenka, and L. DiPippo, "An adaptive threshold method for anomaly-based intrusion detection systems," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, 2019.

[20] A. Krim Rahaoui, T. de Riberolles, and J. Song, "Adaptive threshold for anomaly detection in atm radar data streams," in *Pattern Recognition and Artificial Intelligence: Third International Conference, ICPRAI*, 2022.

[21] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[22] R. Laxhammar and G. Falkman, "Online learning and sequential anomaly detection in trajectories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[23] A. Ghafouri, W. Abbas, A. Laszka, Y. Vorobeychik, and X. Koutsoukos, "Optimal thresholds for anomaly-based intrusion detection in dynamical environments," in *7th International Conference on Decision and Game Theory for Security*, 2016.

[24] M. Yu and S. Sun, "Policy-based reinforcement learning for time series anomaly detection," *Engineering Applications of Artificial Intelligence*, 2020.

[25] X. Yang, E. Howley, and M. Schukat, "Adt: Agent-based dynamic thresholding for anomaly detection," 2023.

[26] R. A. Bridges, J. D. Jamieson, and J. W. Reed, "Setting the threshold for high throughput detectors: A mathematical approach for ensembles of dynamic, heterogeneous, probabilistic anomaly detectors," in *IEEE International Conference on Big Data (2017)*.

[27] D. Minovski, C. Åhlund, K. Mitra, and I. Cotanis, "Anomaly detection for discovering performance degradation in cellular iot services," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*.

[28] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, 2018.

[29] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[30] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[31] M.-C. Dani, F.-X. Jollois, M. Nadif, and C. Freixo, "Adaptive threshold for anomaly detection using time series segmentation," in *Neural Information Processing*, 2015.

[32] R. Heartfield, G. Loukas, A. Bezemskij, and E. Panaousis, "Self-configurable cyber-physical intrusion detection for smart homes using reinforcement learning," *IEEE Transactions on Information Forensics and Security*, 2021.

[33] J. Vanschoren, "Meta-learning: A survey."

[34] D. Schubert, P. Gupta, and M. Wever, "Meta-learning for automated selection of anomaly detectors for semi-supervised datasets," in *International Symposium on Intelligent Data Analysis*, 2023.

[35] T. Zoppi, M. Gharib, M. Atif, and A. Bondavalli, "Meta-learning to improve unsupervised intrusion detection in cyber-physical systems," *ACM Trans. Cyber-Phys. Syst.*, 2021.

[36] W. Lee, S. Stolfo, and K. Mok, "A data mining framework for building intrusion detection models," in *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, 1999.

[37] T. Feng, Q. Qi, J. Wang, and J. Liao, "Few-shot class-adaptive anomaly detection with model-agnostic meta-learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021.

[38] D. Zha, K.-H. Lai, M. Wan, and X. Hu, "Meta-aad: Active anomaly detection with deep reinforcement learning," in *2020 IEEE International Conference on Data Mining (ICDM)*, 2020.

[39] B. Bilalli, A. Abelló Gamazo, and T. Aluja Banet, "On the predictive power of meta-features in openml," *International Journal of Applied Mathematics and Computer Science*, 2017.

[40] I. e. a. Sharafaldin, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *4th International Conference on Information Systems Security and Privacy*, 2018.

[41] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *IEEE Conference on Communications and Network Security (CNS)*, 2022.

[42] A. e. a. Khraisat, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, 2019.

[43] F. T. e. a. Liu, "Isolation forest," in *Eighth IEEE International Conference on Data Mining*, 2008.

[44] ——, "On detecting clustered anomalies using sciforest," in *ECML PKDD*, 2010.

[45] D. Cortes, "Revisiting randomized choices in isolation forests," *arXiv preprint arXiv:2110.13402*, 2021.