Dynamic ML Model Updating Strategies for Detection of Evolving Network Attacks

Aleksandra Knapińska[1,2] and Marija Furdek[1]
[1]Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
[2]Department of Systems and Computer Networks, Wrocław University of Science and Technology, Wrocław, Poland
{alekna, furdek}@chalmers.se

2025 15th International Workshop on Resilient Networks Design and Modeling (RNDM) | 979-8-3315-8497-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/RNDM66856.2025.11073800

***Abstract*—Attack detection plays a crucial role in managing the security of communication networks. Thus, advanced methods have been developed to tackle this problem. However, the need for more flexible and adaptable detection systems remains, able to generalize across various attack scenarios and retain model performance over time. In this paper, we address these challenges by proposing two versions of a continuously updating attack detector in the form of binary classifier. Experimental evaluation on diverse datasets and various base classifiers confirms the effectiveness of our methodology.**

***Index Terms*—attack detection, machine learning, data stream**

I. Introduction

Attack detection is a critical component of communication network security management [1]. As the number of connected devices grows, so does the volume of malicious activity targeting these networks. Attacks such as denial of service, unauthorized access, or port scanning can disrupt network operations, leading to severe consequences. Therefore, rapid and accurate identification of such attacks is essential for network operators to ensure continuous and secure service delivery.

Network operators typically collect vast amounts of telemetry data through continuous monitoring of network elements. Machine Learning (ML) algorithms can effectively analyze these large datasets, identifying patterns that are crucial for security management tasks [2], [3]. In particular, deep learning methods have been proven to be powerful for intrusion detection, as they are capable of uncovering complex dependencies in large-scale data [4], [5].

However, advanced deep learning models often come with significant challenges. They tend to be computationally intensive and require extensive training datasets to achieve satisfactory performance. Moreover, pre-trained models may not always be suitable, especially when malicious communication behaviors are sporadic and lack stable patterns [6]. Consequently, it is essential to dynamically evaluate and update attack detection models to incorporate new knowledge gained during their operation. Despite this, many existing studies fail to assess the long-term performance of their models, and dynamic methods are often not directly compared to static approaches, leaving the theoretical benefits of dynamic models unverified.

This work was supported by the Swedish Research Council (2023–05249), and the European Commission's Digital Europe Programme (101127973) through the 5G-TACTIC project.

Additionally, a common limitation in current research is the focus on detecting specific attack types and training specialized detectors. In practice, the real-world network environment is unpredictable, and new attack types can emerge at any time. This highlights the need for more flexible and adaptable detection systems that can generalize across various attack scenarios. A popular solution to this problem is the use of semi- and unsupervised ML models [7], [8], which help the operators distinguish *benign* and *attack* samples. However, those methods often lack any updating capabilities, thus, the possibilities of including up-to-date knowledge about evolving attack types is limited. Moreover, they cannot provide fine-granular diagnostic information, e.g, details regarding the type of an attack.

To address these challenges, we formulate two research questions: ***Q1)*** *Can a binary classifier trained on aggregated data of various intrusions successfully recognize network attacks?* ***Q2)*** *Can an attack detector retain its performance during the network lifetime when new types of attacks emerge?*

In response to these questions, we propose a framework for continuously updating ML models for attack detection formulated as a binary classification task. Treating the network traffic telemetry data as an imbalanced data stream [9], where the traffic samples are categorized as either *benign* or *attack*, we conduct extensive simulation studies using public datasets to demonstrate the necessity of dynamic model updates in attack detection systems. Comparison of our proposed approaches with classical, statically trained models, indicates how the high prediction quality can be retained when new attack types emerge during the network lifetime.

II. Approaches to Network Attack Detection

The existing approaches to network attack detection achieve excellent performance within the environments they are evaluated. However, some shortcomings and limitations persist, leaving the gap for new studies.

Modern network attack detection techniques are usually based on deep learning models [4], [5]. The achieved accuracy scores often approach 100%, demonstrating great robustness of these methods. However, big models face two main limitations: the reliance on specialized classifiers designed for specific attack types and the potential degradation of detection quality as new attack types emerge.

Models tailored to detect particular attacks often fail to generalize to new or diverse attack types, limiting their scalability and effectiveness in real-world, dynamic environments. Consequently, such systems may require frequent retraining or reconfiguration, which can be resource-intensive and impractical for continuously evolving networks. To address the unknown scope of attacks, methods based on semi- or unsupervised learning and anomaly detection can be employed [8], [10]. Recent proposals also include practical implementations, especially related to the physical layer [11], [12]. However, those methods often lack the dynamic update capabilities. Moreover, they cannot provide fine-granular diagnostic information, e.g., details regarding the type of an attack.

To maintain high performance over extended periods, dynamic methods that continuously update models are necessary. However, existing methods often lack efficient mechanisms for incorporating new knowledge into the detection process without significant computational overhead or loss of accuracy. Current approaches focus mainly on speeding up the convergence to achieve fast re-training [13]. In such a scheme, after a sufficient amount of new data is available, the model is built from the ground-up. Another solution is the addition of a windowing mechanism [10], [14]. However, some of these models are still tested with the traditional train-test split, leaving their real-world performance uncertain. Finally, stream-based dynamically updated methods [15], [16] are a solution allowing to inject up-to-date information into the model to keep its good performance over time. However, they are often not directly compared to static approaches, leaving the theoretical benefits of their constant updating unverified.

III. The Proposed Approach

The limitations of classical methods significantly hinder the long-term effectiveness and adaptability of network attack detection systems. To address this gap, we propose a solution based on dynamically updated binary classifiers. In this approach, the attack detection model is continuously updated with new knowledge derived from attacks encountered over time. The overview of the system is illustrated in Fig. 1. We explore two methods for updating the model. The first, referred to as the *dynamic single* approach, leverages the *partial fitting* capabilities of certain classifiers, usually those with incremental training capabilities. This method allows the model to be gradually trained and updated with new batches of data. The second approach, referred to as *dynamic ensemble*, utilizes the Streaming Ensemble Algorithm (SEA) [17], which combines multiple classifiers into an ensemble to improve overall detection performance. In this method, small member classifiers are trained on newly acquired data, and predictions are made by averaging the responses of the ensemble. Each time a new data batch arrives, a new classifier is trained, and the ensemble size is maintained through a pruning procedure that removes the worst-performing classifier from the pool.

In a practical setting, both proposed approaches operate with a pre-set data batch size. Specifically, incoming data samples are classified as either *attack* or *benign* as they come. Once a predefined number of them has been processed (the chosen batch size is reached), the classifier is updated using the known labels obtained for batch. For the *dynamic single* approach, this involves partial fitting, where the existing model is updated with the new data portion. For the *dynamic ensemble* approach, a new member classifier is trained on the new data batch, and the ensemble is pruned based on a specified metric. A potential tradeoff of these solutions is the introduced overhead, which should guide the decisions on the frequency of training. However, since batch sizes are typically small (around 250 samples), these updates can be performed quickly, without introducing significant computations. Moreover, the savings are still substantial when compared to a full model retraining.

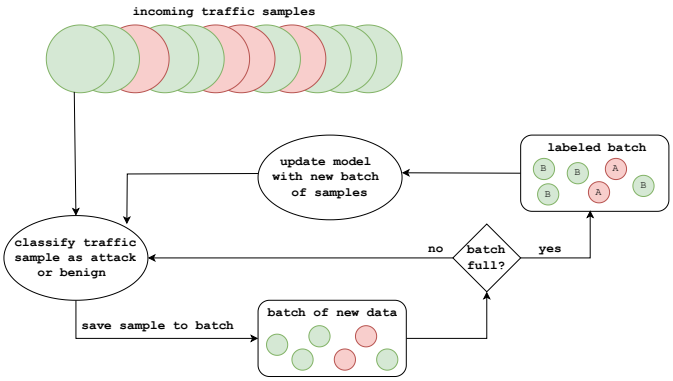

Fig. 1. Proposed framework for dynamic model updates.

IV. Simulation Environment and Assumptions

This section describes the main settings and assumptions for the experimental evaluation we conducted. We further describe the datasets and metrics.

A. Main assumptions

In our experiments, we try to mimic a realistic communication network environment where the scope of possible attacks is unknown and evolving. For this reason, we consider attack detection as a binary classification: for each sample, the model decides whether it represents an attack or not. The particular attack type is not specified, only an alarm is raised. Various existing studies from the literature focus on using separate detectors for predefined attack types. However, as our focus is on detecting previously unseen attacks, we consider a general scenario where only the normal network operating conditions are known.

We consider three base classifiers: Multilayer Perceptron (MLP) [19], Gaussian Naïve Bayes (GNB) [20], and Logistic Regression with Stochastic Gradient Descent training (SGD) [21]. For the most direct comparison, we evaluate each of them in three contexts: *static*, where the classifier is trained on a large amount of data and then put into operation, *dynamic single*, where the classifier is trained on a small amount of data and dynamically updated, and *dynamic ensemble*, where a number of classifiers trained on different data subsets create a dynamically updating ensemble. For dynamically updated approaches, we use the *test-then-train* experimental protocol [22]. In a nutshell, for each batch of data, the model first makes

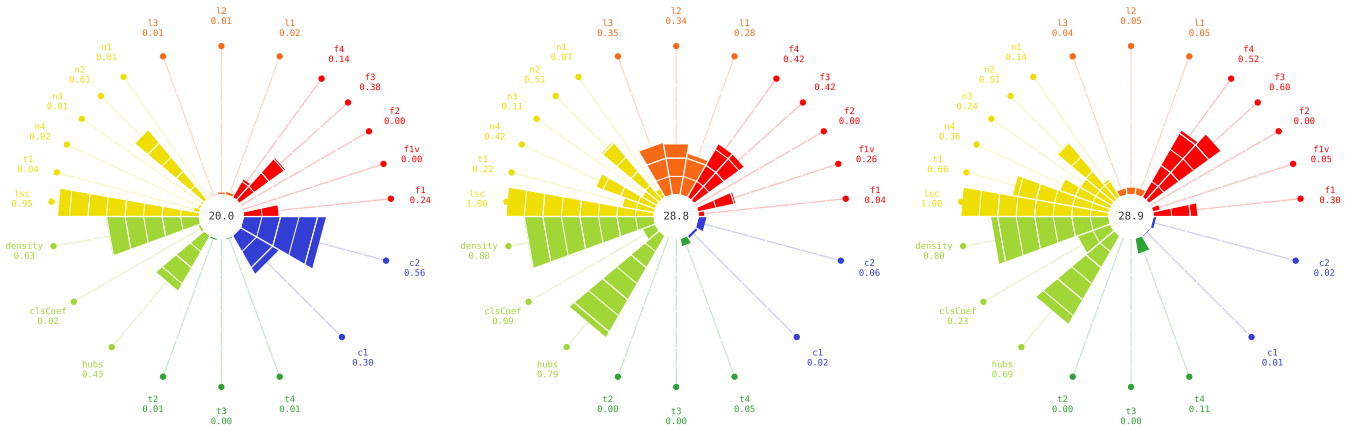

Fig. 2. Dataset complexity assessment with `problexity` [18]. KDDCup dataset (left), RoEduNet dataset (middle), and IoT-23 dataset (right).

the predictions. These predictions are compared to the ground-truth labels later disclosed to the models. The performance of the model is then evaluated by calculating the different metrics, and the model is updated.

B. Datasets

To rigorously evaluate the proposed methodology, we use three publicly available datasets: *KDDCup* [23], *RoEduNet* [24], and *IoT-23* [15] (specifically, the CTU-IoT-Malware-Capture-1-1_0 as in the paper). All of them contain various types of network attacks. For the purpose of experiments, we binarize them: all *benign* samples are labeled as 0 and all *attack* samples are labeled as 1. Despite combining attack samples into a joint class, all datasets remain imbalanced, with the minority (*attack*) class accounting for 20–40% of samples. Importantly, by considering a varied dataset spectrum, we aim to ensure that our approach is not overfitted to a specific network or scenario but instead demonstrates robustness and applicability to a broad range of intrusion detection contexts. To assess the complexity of the classification problems represented by the considered datasets, we use `problexity` [18], which implements various measures collected in [25].

Fig. 2 presents the `problexity` summary plots with the overall score in the center and other metrics denoted with different colors extending radially. The aggregated complexity score is very similar across datasets, but its underlying reasons differ. Let us briefly describe the six different groups of considered data complexity measures and their meaning in a counter-clockwise manner; for more details we refer to [18].

Feature-based measures, depicted with red color in Fig. 2 (*f1–f4*), describe the ability of features to separate classes. They take values between 0 and 1, where a higher value indicates a more complex problem. *f1*, *f1v* and *f2* gauge the overlap between the values of the same feature among different classes. For all considered datasets, there is some overlap (values of *f1* or *f1v* higher than 0.2), but the overall extent is not high (low *f2* value). *f3* and *f4* gauge the feature efficiency in class separation. High *f3* values indicate that there are features with really close values in *attack* and *benign* classes, and their portion is quite high, especially in RoEduNet and IoT-23 (high *f4*).

Linearity measures, depicted with orange color (*l1–l3*), describe the level of linear class separation. Linear class separability is measured by various parameters of a fitted Support Vector Machine (SVM) classifier. The values obtained for the considered datasets indicate that the classes in the KDDCup dataset are quite well separable, the IoT-23 dataset introduces some more challenges, whilst the RoEduNet is quite difficult in that regard. The errors made by the linear classifier are quite distant from one another (high *l1* values), in large portion (high *l2* values), and not resolvable by interpolation (high *l3* values).

Neighborhood measures, depicted with yellow color in Fig. 2 (*n1–n4, t1, lsc*), describe the neighborhood of samples in the feature space. We can see that RoEduNet and IoT-23 contain a noticeable portion of borderline points (*n1*), causing errors and nonlinearity of the nearest neighbors classifier (*n3* and *n4*, respectively). All datasets are also characterized by fairly high distance between the samples and their nearest neighbors of the same class (high *n2* value). In terms of *l1*, which denotes the fraction of hypershperes needed to cover the data, the IoT-23 dataset appears as the most difficult, and the KDDCup as the easiest. All datasets are also characterized by high local set average cardinality, counting sets of points that lie closer to a given sample than their nearest neighbor from the opposing class (high *lsc* value).

Network measures, depicted with bright green color (*density, clsCoef, hubs*), describe an epsilon–Nearest Neighbors graph generated from the data samples. This group of metrics is another representation of class separability and instance distinctness. The most complex classification problem in terms of both *density* and *hubs* is represented by the RoEduNet dataset, followed by IoT-23. The remaining clustering coefficient metric (*clsCoeff*) indicates that IoT-23 is the most complex of the three datasets.

Dimensionality measures, depicted with darker green color (*t2–t4*), analyze the relation between the number of features and the number of instances. The first two metrics do not indicate high dimensionality of any dataset. However, *t4* shows that for RoEduNet and IoT-23, the number of principal component analysis (PCA) components needed to represent

95% of the data variance is noticeably higher than for the KDDCup dataset.

Finally, the *class imbalance measures*, depicted with blue color (*c1–c2*), describe the degree of class imbalance as the entropy of class proportions and imbalance ratio. Although each of the considered datasets contain more *benign* samples than the *attack* ones, the imbalance effect is the most prominent in the case of the KDDCup dataset, which contains 22 types of attacks accounting for 25% of samples.

In summary, each dataset poses different challenges to the network attack detection task. They vary in terms of the portion of *attack* samples, their general separability, the ability of features to distinguish them, and general data variance. These challenges impact the performance of various models and classifiers, requiring a thorough analysis of their applicability to security analysis in dynamic threat scenarios.

C. Metrics

The choice of a reliable performance metric is crucial for understanding and evaluating the performance of different models. In scenarios with high data imbalance, which are in the focus of our study, the learning process guided by global performance metrics such as *prediction accuracy* usually induces a bias towards the majority class, while the rare minority class samples might remain unknown even if the prediction model has high overall precision [26]–[28]. For that reason, in this work, we focus on six measures reliably capturing the model performance in imbalanced settings. These metrics are calculated using the number of *true positives (TP) – attacks* correctly identified as *attacks*, *false positives (FP) – benign* samples incorrectly classified as *attacks*, *true negatives (TN) – benign* samples correctly identified as *benign*, and *false negatives (FN) – attacks* missed by the model. Below, we provide the formulas and brief explanations for each of them.

- *Precision* is the proportion of predicted *attacks* that are actually *attacks*, given by:
$$\text{Precision} = \frac{TP}{TP + FP}$$
- *Recall* is the proportion of actual *attacks* that are correctly identified by the model, given by:
$$\text{Recall} = \frac{TP}{TP + FN}$$
- *Specificity* is the proportion of *benign* samples that are correctly identified as *benign*, given by:
$$\text{Specificity} = \frac{TN}{TN + FP}$$
- *Balanced Accuracy Score (BAC)* accounts for both *recall* and *specificity* and is used with imbalanced data, given by:
$$\text{BAC} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right)$$
- *F1 Score (F1)* is the harmonic mean of *precision* and *recall*, providing a balance between the two metrics, given by:
$$\text{F1} = 2 \times \frac{TP}{2 \times TP + FP + FN}$$
- *Geometric Mean (G-Mean)* is the geometric mean of *recall* and *specificity*, providing a balance between the model's ability to detect *attacks* and avoid false alarms, given by:
$$\text{G-Mean} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}}$$

All of the considered metrics can take on values between 0 and 1, and should be maximized.

V. Experimental Evaluation

Let us evaluate the performance of the proposed stream-based *dynamic single* and *dynamic ensemble* methods in detecting evolving security threats against the baseline *static* approach. In our evaluation, every experiment is repeated 100 times. In each run, we randomly draw a subset of 125.000 data samples from the corresponding dataset, divided into 500 batches of 250 samples. In the static approach, the classifier is trained on the first 100 batches of data (25.000 samples) and then predicts each subsequent batch without updating. The *dynamic* methods only need the first 250-sample batch of data for initial training and start working right away. For SEA, the ensemble size is set to 3 classifiers, which is the value determined through preliminary tuning experiments. In the following part, we report the average metric values of 100 experiment replications in each dataset.

A. Evaluation of overall model performance

To assess the successful network attack recognition rates of binary classifiers trained on aggregated data, tackling RQ1 of this paper, we analyze the overall performance of the analyzed methods across the whole datasets. The overall performance is presented in Fig. 3, where each row corresponds to a base classifier, and each column shows the classifier's performance for a particular dataset. In each plot, dashed lines depict the metric values obtained by the baseline, *static* classifier, solid lines denote the results obtained by the *dynamic single* classifier, and dotted lines represent the results for the *dynamic ensemble* version.

Before diving into details, a general noticeable trend is that in almost all of the cases, the *dynamic* approaches outperform the *static* one in all metrics (recall that the value of each measure should be maximized). For each dataset, there is at least one method that results in high attack detection performance.

The KDDCup dataset, despite the highest imbalance, appears to be the simplest for all approaches and classifiers, which is in line with the insights provided in the complexity analysis in Sec. IV-B. All classifiers perform great across the metrics when used dynamically, compared to the *static* approach. Between the *dynamic* methods, the *ensemble*-based version is noticeably better. In practice, when correct attack recognition is a priority, either of the classifiers would be a good choice, as they excel in *recall*. To additionally minimize false alarms, MLP or SGD should be selected, as they achieve high *precision*.

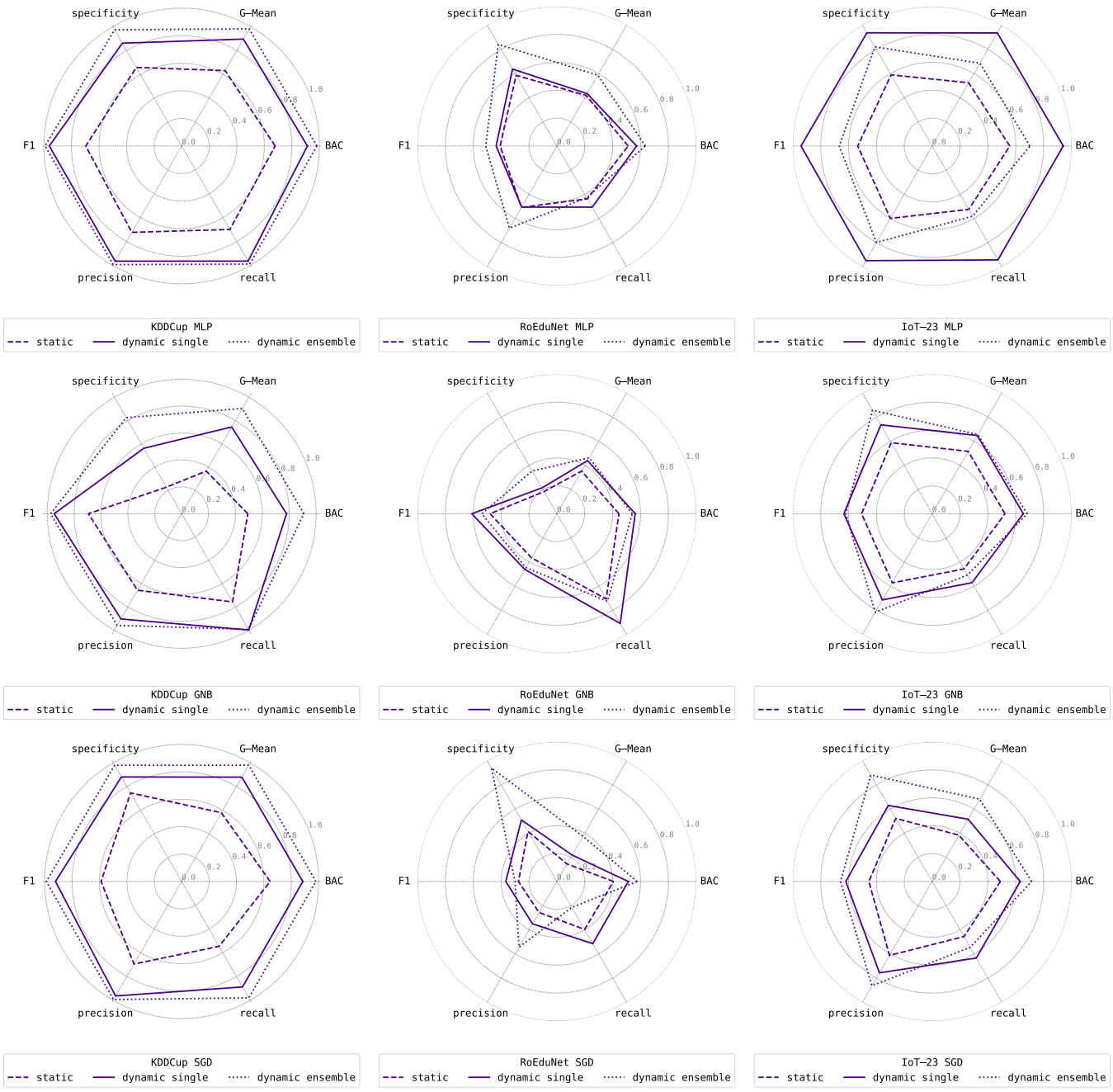

Fig. 3. Average metric values over 100 experiment replications. MLP classifier (top row), GNB classifier (middle row), and SGD classifier (bottom row). KDDCup dataset (left column), RoEduNet dataset (middle column), and IoT-23 dataset (right column).

The performance analysis on the RoEduNet dataset highlights the importance of considering individual metrics including *precision* and *recall*, as the aggregate measures provide limited, sometimes ambiguous insights [29]. Despite quite unsatisfactory values of *BAC*, *F1* and *G-Mean* obtained by the methods, a good performance in terms of *recall* (*dynamic single*–GNB), *specificity* (*dynamic ensemble*–MLP and *dynamic ensemble*–SGD), or *precision* (*dynamic ensemble*–SGD) can be achieved by the dynamic methods. This can be explained with a suspected bias of GNB towards the minority class (the model prioritizes rising an alarm when uncertain) and of MLP and SGD towards the majority class (the model classifies the sample as *benign* when uncertain) in this dataset. Nevertheless, the previously conducted data complexity analysis is reflected in these results. Thus, the choice of classifiers for data with similar characteristics should be guided by the network operator's priority: detection of attacks, at potentially poorer false alarm rate, or false alarm avoidance at a trade-off with lower attack recognition. Finally, the IoT-23 dataset seems quite challenging as well, but the MLP classifier copes with it, especially as a *single* dynamically updating model. The GNB and SGD classifiers perform well, especially in terms of *precision* and *specificity*, when used in an *ensemble*.

In summary, despite the differences among the datasets

and the classifiers, we can conclude that successful attack recognition is possible for methods trained on data that jointly represents various attack types. Furthermore, updating the predictors dynamically, either by *partial fitting* or by using an *ensemble*, improves the attack detection performance gauged by various metrics. To answer ***RQ1***, we can conclude that *binary classifiers trained on aggregated data of various intrusions can successfully recognize network attacks, especially when they are dynamically updated.*

B. Evaluation of model performance over time

The aggregated performance measured over the whole dataset does not, however, provide the necessary insights into the ability of different classifiers to detect attacks. Even if the averaged performance is high, we also need to examine how it changes over time to evaluate the performance retention.

To answer RQ2 posed in this paper, let us analyze a series of representative test cases presented below. First, we check the average values of metrics over consecutive batches of data for the statically trained detector. Fig. 4 presents an example of the performance of the MLP classifier for the IoT-23 dataset according to BAC and F1. The trends for the remaining metrics, omitted here for space constraints, are equivalent. As the first 100 data batches were used for model training, no predictions were available for them and the metric value was undefined. Later, we observe very high prediction quality in the first data batches after the model is freshly trained. However, the metric values quickly start to drop as the data changes and new attack types emerge. This highlights a key limitation of *static* models: they require a substantial amount of initial training data before they can begin operating, and they deliver high prediction accuracy only for a limited period.

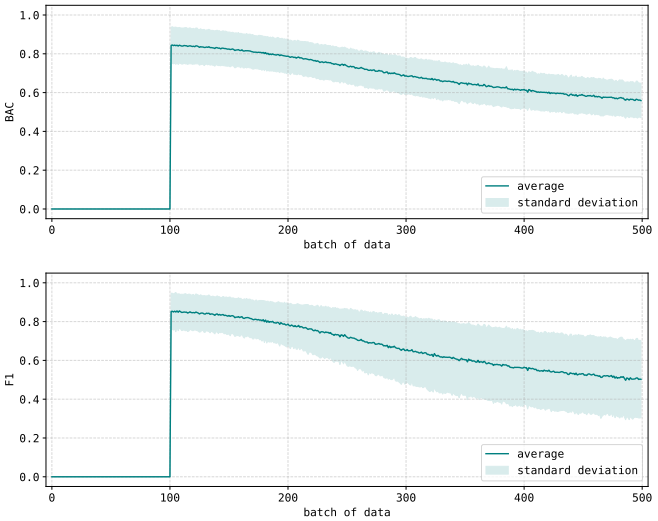

Fig. 4. BAC (top) and F1 (bottom) over consecutive batches of data. Average of 100 experiment replications with standard deviation. IoT-23 dataset, *static* MLP classifier.

A very similar trend can be observed for the remaining classifiers and datasets. Let us, however, investigate the average metric plotted together with the individual runs of the experiment replications in Fig. 5. As the data was randomly sampled each time, there are cases where no new attack types appear and the model retains its good quality over multiple subsequent batches. On the other hand, in cases where unseen attack types occur shortly after training, an almost immediate performance degradation is observed. In the case of a more challenging IoT-23 dataset, we can see that no model is able to retain performance over time across 100 replications, as presented in Fig. 5. This highlights the weaknesses in the level of security defenses provided by statically trained approaches.

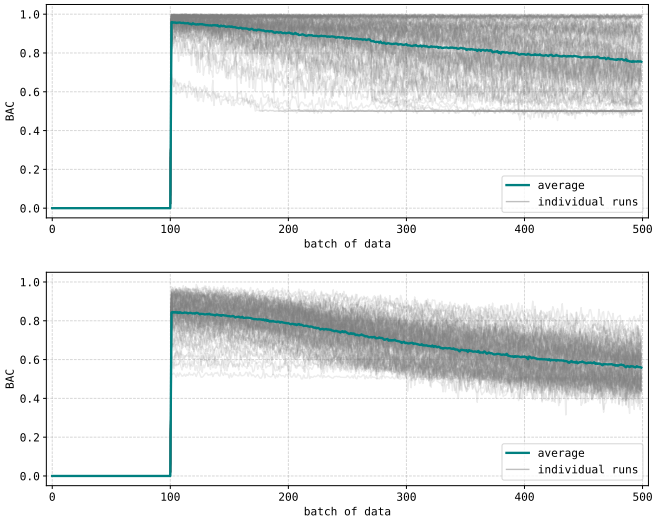

Fig. 5. BAC over consecutive batches of data. Average of 100 experiment replications and individual runs. *Static* MLP classifier; KDDCup dataset (top) and IoT-23 dataset (bottom).

Let us now focus on the dynamically updated models (see representative examples plotted in Fig. 6–7). Contrary to their *static* counterparts, the *dynamic* models are ready for deployment from the beginning, without requiring pre-training or initial collection of large amounts of data. This increases their practical value in realistic network settings. When a *single* classifier is used, a few data batches usually need to be processed before achieving good performance (Fig. 6), while using an *ensemble* allows for an even faster convergence to excellent performance and higher stability (Fig. 7). Throughout network operation, new attacks can go undetected, which creates momentary performance drops present in each experiment replication (see Fig. 8). However, the model quickly learns thanks to the updating, and can continue operating on a satisfactory level.

In summary, updating classifiers with new knowledge as attackers evolve their intrusion techniques is essential for maintaining secure network operations over time. This continual learning also improves the classifier's ability to generalize across diverse attack types, resulting in a more robust and effective detection system. Therefore, to answer ***RQ2***, *attack detectors can retain their performance throughout network operation when new types of attacks emerge if they are dynamically updated.*

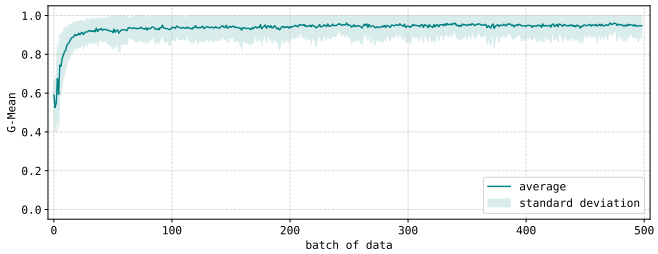

Fig. 6. G-Mean over consecutive batches of data. Average of 100 experiment replications with standard deviation. IoT-23 dataset, *dynamic single* SGD classifier.

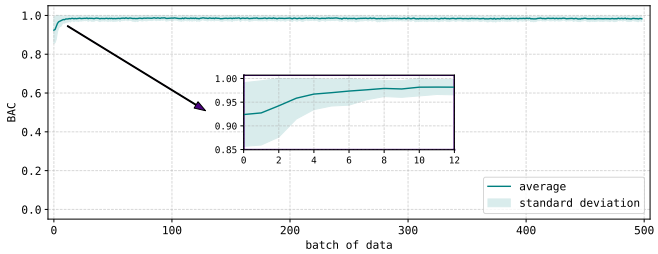

Fig. 7. BAC over consecutive batches of data. Average of 100 experiment replications with standard deviation. KDDCup dataset, *dynamic ensemble* MLP classifier.

VI. Conclusions

In this paper, we tackled the problem of detecting evolving network attacks. We focused on two practical challenges: the ability of ML models to recognize various attack types, unknown in advance, and model performance retention. To solve them, we proposed two approaches based on dynamic model updating. Our experimental evaluation on three public datasets revealed the usefulness of the methodology. The results demonstrated how the same ML classification algorithm, when used differently, can yield significantly different performance outcomes.

These results underscore the importance of continuous model adaptation to handle the evolving nature of emerging threats. The proposed methods offer a flexible and scalable solution for real-time attack detection, making them suitable for deployment in dynamic network environments.

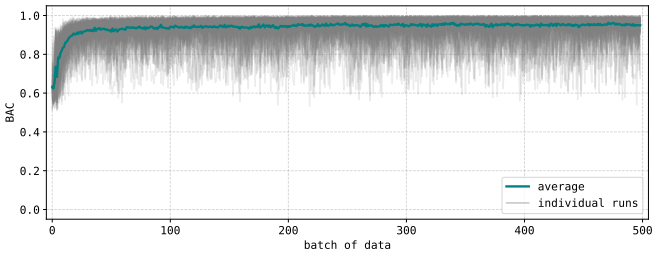

Fig. 8. BAC over consecutive batches of data. Average of 100 experiment replications and individual runs. IoT-23 dataset, *dynamic single* MLP classifier.

References

[1] M. Furdek *et al.*, "Optical network security management: requirements, architecture, and efficient machine learning models for detection of evolving threats," *Journal of Optical Communications and Networking*, vol. 13, no. 2, pp. A144–A155, 2021.

[2] M. Furdek *et al.*, "Machine learning for optical network security monitoring: A practical perspective," *Journal of Lightwave Technology*, vol. 38, no. 11, pp. 2860–2871, 2020.

[3] C. Natalino *et al.*, "Flexible and scalable ML-based diagnosis module for optical networks: a security use case," *Journal of Optical Communications and Networking*, vol. 15, no. 8, pp. C155–C165, 2023.

[4] T. Yi *et al.*, "Review on the application of deep learning in network attack detection," *Journal of Network and Computer Applications*, vol. 212, p. 103580, 2023.

[5] Y. Wu *et al.*, "Network attacks detection methods based on deep learning techniques: A survey," *Security and Communication Networks*, vol. 2020, no. 1, p. 8872923, 2020.

[6] G. Duan *et al.*, "Practical cyber attack detection with continuous temporal graph in dynamic network system," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 4851–4864, 2024.

[7] T. Zoppi *et al.*, "Unsupervised anomaly detectors to detect intrusions in the current threat landscape," *ACM/IMS Transactions on Data Science*, vol. 2, no. 2, pp. 1–26, 2021.

[8] T. Zoppi *et al.*, "Which algorithm can detect unknown attacks? comparison of supervised, unsupervised and meta-learning algorithms for intrusion detection," *Computers & Security*, vol. 127, p. 103107, 2023.

[9] G. Aguiar *et al.*, "A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework," *Machine Learning*, vol. 113, no. 7, pp. 4165–4243, 2024.

[10] C. Natalino *et al.*, "Root cause analysis for autonomous optical network security management," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2702–2713, 2022.

[11] C. Natalino *et al.*, "Microservice-based unsupervised anomaly detection loop for optical networks," in *OFC Conference*, 2022.

[12] P. Lechowicz *et al.*, "Trade-offs in implementing unsupervised anomaly detection with TAPI-based streaming telemetry," in *HPSR Conference*, 2024.

[13] K. Alrawashdeh *et al.*, "Fast activation function approach for deep learning based online anomaly intrusion detection," in *BigDataSecurity Conference*, 2018.

[14] Z. Shi *et al.*, "Deepwindow: An efficient method for online network traffic anomaly detection," in *HPCC Conference*, 2019.

[15] P. Zyblewski *et al.*, "Cyber-attack detection from IoT benchmark considered as data streams," in *CORES Conference*, 2021.

[16] P. Mulinka *et al.*, "Stream-based machine learning for network security and anomaly detection," in *Big–DAMA Workshop*, 2018.

[17] W. N. Street *et al.*, "A streaming ensemble algorithm (SEA) for large-scale classification," in *SIGKDD Conference*, 2001.

[18] J. Komorniczak *et al.*, "problexity—an open-source python library for supervised learning problem complexity assessment," *Neurocomputing*, vol. 521, pp. 126–136, 2023.

[19] G. E. Hinton, "Connectionist learning procedures," in *Machine Learning*. Elsevier, 1990, pp. 555–610.

[20] J. R. Anderson *et al.*, "Explorations of an incremental, bayesian algorithm for categorization," *Machine Learning*, vol. 9, pp. 275–308, 1992.

[21] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *ICML Conference*, 2004.

[22] G. Ditzler *et al.*, "Learning in nonstationary environments: A survey," *IEEE Computational Intelligence Magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[23] S. J. Stolfo *et al.*, "Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection: Results from the jam project." [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/task.html

[24] M.-E. Mihailescu *et al.*, "The proposition and evaluation of the roedunet-simargl2021 network intrusion detection dataset," *Sensors*, vol. 21, no. 13, p. 4319, 2021.

[25] A. C. Lorena *et al.*, "How complex is your classification problem? a survey on measuring classification complexity," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–34, 2020.

[26] G. Haixiang *et al.*, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[27] G. M. Weiss, "Mining with rarity: a unifying framework," *ACM SigKDD Explorations Newsletter*, vol. 6, no. 1, pp. 7–19, 2004.

[28] G. M. Weiss *et al.*, "Learning to predict extremely rare events," in *AAAI Workshop on Learning from Imbalanced Data Sets*, 2000.

[29] W. Węgier *et al.*, "Multicriteria classifier ensemble learning for imbalanced data," *IEEE Access*, vol. 10, pp. 16 807–16 818, 2022.