Online Framework for Intrusion Detection Systems with Explainable Concept Drift Detection and Adaptation Phases

Salvatore Drago[1,*,†], Manuel Lorusso[2,†], Pierluca Ferraro[2,3,†], Alessandra De Paola[2,3,†] and Giuseppe Lo Re[2,3,†]

[1]IMT School for Advanced Studies Lucca, Italy
[2]Department of Engineering, University of Palermo, Italy
[3]Cybersecurity National Lab, CINI - Consorzio Interuniversitario Nazionale per l'Informatica

Abstract

Machine learning-based Intrusion Detection Systems (IDSs) have become increasingly critical for identifying and mitigating network threats by analyzing data flows for anomalous or malicious activity. However, the dynamic nature of network environments makes static training data increasingly obsolete, leading to a reduction in detection accuracy, a phenomenon known as concept drift. In recent years, online learning approaches have been proposed to enable IDSs to detect drift and adapt to changing conditions. However, efforts have primarily focused on promptly detecting drift and developing fast incremental adaptation strategies, while overlooking two fundamental aspects, i.e., labeling cost and explainability. Labeling cost, defined as the time and resources involved in the data annotation process, is critical in an online setting, where domain experts are required to label data in real time resulting in a potentially serious shortage of labeled data. Moreover, explainability of machine learning model decisions is crucial to understand how concept drift affects system performance and how the model adapts its behavior to align with the new data distribution. To address these challenges, this paper proposes an online framework for IDS powered by an unsupervised, explainability-driven concept drift detector. The detector identifies concept drift without requiring labeled data and provides insights into its impact on model decision-making. The framework also includes an online adaptation strategy that minimizes labeling costs. Experimental evaluation on a real-world network dataset with various concept drift scenarios demonstrates the system's effectiveness. The framework detects concept drift and adapts quickly when necessary, achieving performance comparable to state-of-the-art systems while reducing labeling costs. It also provides explainability of the drift's impact and shows how the model's decisions evolve over time.

Keywords

Machine Learning, Online Intrusion Detection System, Threat Detection, Concept Drift, Explainable AI

1. Introduction and Related Work

In the modern cybersecurity landscape, the complexity and volume of network attacks have highlighted the limitations of traditional signature-based defense mechanisms [1]. While such systems remain effective against known threats, they lack the capacity to detect novel zero-day or evolving attack strategies [2, 3, 4]. As a result, ML-based IDSs have become a central paradigm, exploiting their ability to learn discriminative patterns from historical traffic data and to generalize beyond known attack instances [5, 6, 7, 8].

Nevertheless, the deployment of ML models in operational network environments faces challenges due to the inherently non-stationary nature of data streams [9, 10]. Network traffic evolves continuously: the baseline pattern of benign traffic may evolve, for example, due to modifications in network topology, infrastructure upgrades, deployment or decommissioning of services, software updates, or shifts in

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT
*Corresponding author.
†These authors contributed equally.
salvatore.drago@imtlucca.it (S. Drago); manuel.lorusso@community.unipa.it (M. Lorusso); pierluca.ferraro@unipa.it (P. Ferraro); alessandra.depaola@unipa.it (A. De Paola); giuseppe.lore@unipa.it (G. Lo Re)
0009-0009-0367-0484 (S. Drago); 0000-0003-1574-1111 (P. Ferraro); 0000-0002-7340-1847 (A. De Paola); 0000-0002-8217-2230 (G. Lo Re)

employee behavior following new organizational policies. Moreover, zero-day attacks or adversaries that adapt their techniques [11, 12, 13] often produce traffic patterns that differ substantially from previously observed malicious activity.

This phenomenon, referred to as concept drift [14], alters the statistical distribution of the features over time, ultimately degrading the detection accuracy of IDSs if not properly addressed.

To mitigate the performance drop caused by evolving network traffic, the research community has increasingly adopted paradigms of *online learning under concept drift* [15, 16, 17, 18, 19]. Unlike static learning, where models are trained once and deployed indefinitely, online approaches enable IDSs to update their knowledge base incrementally.

The authors of AXGBoost [20] and ARF [21] propose adaptive systems capable of incremental updating, leveraging the strong performance showed by ensemble learning models in the context of network-based IDS [22]. Incremental updating is performed through policies that update the weights assigned to individual learners for the final prediction, and through mechanisms that replace outdated learners with new ones trained on recently collected data. By acting on these aspects, such systems can perform different types of incremental updates to address various kinds and intensities of drift, or to gradually update the system even when no drift is present. It is therefore essential to detect concept drift promptly and effectively in order to activate the subsequent adaptation phase. For this reason, these systems use concept drift detectors based on error rates.

These algorithms, such as ADWIN [23], EDDM [24] and KSWIN [25] monitor a one-dimensional distribution of data that describes system performance over time. In such scenarios, the degradation in system performance is attributed to concept drift.

AXGBoost, ARF, and other state-of-the-art systems for online learning under concept drift have primarily focused on promptly detecting drift and developing fast incremental adaptation strategies. A key limitation is the labeling cost: in an online setting, the scarcity of labeled data becomes even more critical, as domain experts would need to label data in real time. Detecting drift based on error rates and applying incremental learning adaptation for supervised models depends on having real-time access to ground truth labels to compare with system predictions.

To reduce the cost of labeling in drift detection, the authors of ADWIN-U [26] propose an unsupervised distribution-based concept drift detector. This detector does not rely on labeled data to detect changes. Instead, drift is inferred by monitoring the statistical characteristics of the input features. The algorithm stores a batch of new records and computes a scalar statistical summary (such as mean, variance, skewness, or kurtosis) based on the distribution of features in the batch. This produces a one-dimensional signal that is then fed into ADWIN for concept drift detection.

With this approach, the labeling cost for concept drift detection is eliminated, although labeling is still required during the adaptation phase. Furthermore, concept drift detectors act primarily as alarm mechanisms: they report that a concept drift has occurred but do not provide insight into why this impacts the model's performance or which features are responsible for the shift.

Moreover, explainability becomes increasingly important to understand how concept drift affects system performance [27] and how, after adaptation, the model adjusts its behavior to align with the new data distribution.

To address these challenges, this paper proposes an online framework for IDS that integrates an unsupervised, explainability-driven concept drift detector capable of identifying drift without requiring data labeling, and of explaining its impact on the decision making of the machine learning model. The framework also includes an online adaptation strategy aimed at reducing labeling costs. The experimental evaluation, performed on a real-world network dataset with different concept drift scenarios, proves the system's effectiveness in detecting concept drift and adapting quickly when necessary, achieving performance comparable to state-of-the-art systems while minimizing labeling costs, enabling explainability of the drift's impact, and showing how the model's decisions evolve over time.

The principal contributions of this paper are summarized as follows:

1. the proposal of an unsupervised, multidimensional, explainability-driven concept drift detector

(UME_CDD) designed to identify concept drift without labeled data and to explain its impact on the decision-making process of a machine learning model;
2. the proposal of an online framework for Intrusion Detection Systems that incorporates a concept drift adaptation strategy aimed at minimizing labeling costs;
3. a comparison among different concept drift detectors applied to static models, to observe how drift affects performance and how the various detectors identify its presence;
4. a comparison of online IDSs employing various adaptive strategies and concept drift detectors, in terms of performance and labeling cost;
5. a comprehensive validation of the proposed system using multiple real-world network datasets affected by concept drift.

The rest of the paper is organized as follows. Section 2 describes the proposed framework architecture. Section 3 details the experimental evaluation and reports the results. Finally, Section 4 summarizes the main conclusions and outlines potential directions for future work.

2. Proposed Framework

This section presents the proposed online framework for Intrusion Detection Systems. It consists of two main components: an adaptive machine learning model (*AM*) and an unsupervised, multidimensional, explainability-driven concept drift detector (UME_CDD).

The workflow of the proposed framework is detailed in Algorithm 1. First, an offline period (line 25-30) is dedicated to collecting the first batch of data, which is used as a training set to train *AM*. This batch is also set through UME_CDD as the representative batch of the current concept. At the end of the offline period, the system can be deployed on a server to enable continuous network traffic monitoring, with the aim of identifying and isolating any suspicious activities.

After this offline period, the system enters the online period (lines 4–24), which consists of three main phases. The first is the prediction phase (lines 4–6), performed in real time by *AM* on each new record from the incoming data stream (*DS*). The second is the concept drift detection phase (lines 7–13), which detects changes in the statistical distribution of input data. If a change is detected, the third phase, adaptation (lines 14–21), is triggered, allowing *AM* to integrate the new data that caused the drift.

The concept drift detection and adaptation phases are performed in batches of size *bs*. For each new batch, if UME_CDD does not detect drift, no online adaptation phase is performed. Otherwise, the system enters an adaptation phase, during which *AM* is incrementally trained for ap batches. During the adaptation phase, the concept drift detector is disabled. It is reset at the end of the phase, when the last batch is set as the representative batch of the current concept. The proposed framework reduces labeling costs while allowing the system to adapt in response to concept drift. Labeling costs are reduced in two main ways: first, by employing a concept drift detector that does not require ground-truth labels to monitor system performance, but instead detects drift by comparing changes in the statistical distribution of feature values across batches (Sec. 2.1); and second, by limiting the adaptation phase to a specific period and only triggering it after drift has been detected. Furthermore, the proposed concept drift detector integrates an explainability module that enables continuous monitoring and understanding of the model's decision-making process over time, both during stationary periods and after concept drift detection. It also provides insight into how the adaptation phase and incremental training have influenced the model's behavior.

2.1. UME_CDD

In this work, we propose UME_CDD, an unsupervised multidimensional explainability-driven drift detector based on monitoring the evolution of feature importance over time. The core hypothesis is that changes in the data distribution first manifest as shifts in the relative role of features within the model's decision-making process. The primary advantage of this approach is that such changes can be

Algorithm 1: Proposed framework.

```
Input : DS: data stream of network traffic records;
        bs: size of batches;
        AM: adaptive machine learning model;
        UME_CDD: proposed concept drift detector;
        ap: adaptation period;
Output:
        Ŷ : the list of system predictions.
New_batch ← [ ]; fb = true; cd = 0
for x_i ∈ DS do
    New_batch.append(x_i)
    if fb == false then
        ŷ_i ← AM.predict(x_i)                    ▷ prediction performed on new record
        Ŷ.append(ŷ_i)
        if len(New_batch)==bs then
            if cd == 0 then
                if CDD.detect_drift(AM, New_batch) then
                    AM.partial_fit(New_batch)    ▷ incremental training
                    cd += 1
                end
            end
            else
                AM.partial_fit(New_batch)
                cd += 1
                if cd > ap then
                    CDD.set_concept(AM, New_batch)
                    cd = 0
                end
            end
            New_batch ← [ ]
        end
    end
    else
        if len(New_batch)==bs then
            AM.fit(New_batch)
            CDD.set_concept(AM, New_batch)
            fb = false;
        end
    end
end
```

detected without ground truth labels, by observing how the feature importance profile varies between a reference batch and the current one, based on the model's predictions.

The main modules and the workflow of UME_CDD are shown in Figure 1. It operates on a batch-oriented streaming architecture, where data is processed in batches of size *bs*. The operation can be divided into two main functions: the set_concept function (dotted line in Fig. 1) and the detect_drift function (continuous line in Fig. 1).

During the set_concept function, a representative batch of the training dataset, B_{cc}, is used to compute feature importance for the *AM* model through a XAI_{module}. As output, a feature importance profile I_{cc}, representing the stable concept, is stored for future comparison. It can be seen as a vector of dimensionality equal to the number of features used, containing the importance score for each feature.

For each new batch, the detect_drift function of UME_CDD computes the feature importance profile I_{new} and the dissimilarity score between I_{new} and I_{cc}.

This score, called δ_{score}, is stored in a one-dimensional statistical concept drift detector (i.e., KSWIN), which determines whether concept drift has occurred. If concept drift is detected, the proposed framework uses the set_concept function at the end of the adaptation period to calculate the new I_{cc}. At this stage, KSWIN is also reset to allow the collection of new dissimilarity scores.

The δ_{score} metric, as defined in Eq. 1, focuses on the stability of the magnitude of feature importance. It quantifies the mean normalized difference in importance values. The calculation is constrained to a feature set U, which is defined as the union of the N most important features from I_{cc} and I_{new}.

The union-based approach ensures that the metric includes features that remained within the top N ranking, those that lost significance and dropped out, and those that gained significance and entered the ranking.

$$\delta_{score} = \frac{1}{|U|} \sum_{f \in U} \frac{|I_{new}(f) - I_{cc}(f)|}{\max(|I_{new}(f)|, |I_{cc}(f)|) + \epsilon} \tag{1}$$

Where:

- $|U|$ is the total number of features in the union set;
- $I_{new}(f)$ and $I_{cc}(f)$ are the mean importance values for feature f in the current and reference batches;
- $\max(|I_{new}(f)|, |I_{cc}(f)|)$, normalizes the absolute difference relative to the largest importance value observed for that feature. This ensures the metric measures relative change;
- ϵ is a constant to prevent division by zero.

The dissimilarity value can be:

- $\delta_{score} = 0$: all features in U maintain their importance values between the reference and current batches (no drift in feature importance);
- $\delta_{score} = 1$: all features exhibit maximum divergence. This occurs when features that were highly important in the reference batch have completely lost significance in the current batch, or vice versa;
- $0 < \delta_{score} < 1$: intermediate scenarios where some features remain stable while others change. Notably, when a feature exits the top-N ranking, its residual importance is compared against its historical high value. Conversely, when a feature enters the top-N ranking, its newly acquired importance is compared to its historical baseline.

It is important to note that the modularity of UME_CDD allows the replacement of its main components, such as XAI_{module}, while maintaining the operating logic unchanged.

The framework supports both global and local explainability methods capable of operating without ground truth labels. When a local method is employed, local results must be aggregated to obtain a global representation of the batch. However, this enables the analysis of local explanations, providing security analysts with a valuable tool for a posteriori investigation of individual instances on specific events.

Two types of XAI_{module} are used in this work with different AM models. The first is SHAP (SHapley Additive exPlanations) [28], whose local explanations are aggregated via averaging to obtain the global importance representation of the batch. SHAP provides a unified theoretical framework for explaining individual model predictions by attributing the contribution of each input feature in a manner grounded in cooperative game theory. The contribution of each feature is quantified through the average marginal effect that the feature has on the prediction over all possible feature subsets. For models that support optimized, high-fidelity attribution methods (such as tree-based ensembles), we compute SHAP values using the TreeExplainer methodology. When applied to an ensemble, the resulting attributions are aggregated across all internal components to reflect the entire model's behavior.

The second is based on Unsupervised Permutation Importance, offering a global perspective on feature relevance. Conceptually, this method relies on the definition of Model Reliance [29], which assesses the model's functional dependence on a variable rather than its additive contribution. The core principle involves breaking the association between a feature and the model's output via random permutation, while preserving the marginal distribution of the input data. Unlike Fisher's original formulation, which quantifies Model Reliance through the increase in expected loss (which necessitates ground-truth labels), we adopt the Unary Quantitative Input Influence (QII) [30]. Specifically, we

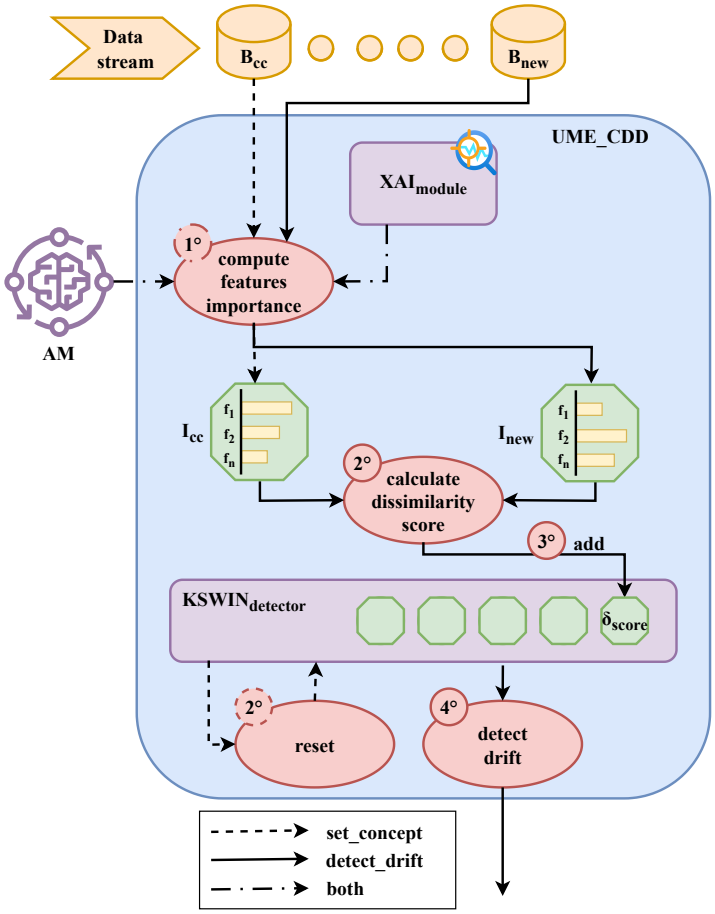


Figure 1: Workflow of UME_CDD.

compute the empirical estimator of the Average QII on Actual Outcomes, quantifying importance as the frequency of prediction divergence over the batch:

$$I_j = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left(\hat{y}_i \neq \hat{y}_{i,\text{perm}_j}\right) \tag{2}$$

Here, I_j is the estimated importance of feature j given a batch size N. The terms $\hat{y}_i$ and $\hat{y}_{i,\text{perm}_j}$ refer to the model's predictions on the original and shuffled data, respectively, where the shuffling of feature j breaks its association with the target. The indicator function $1(\cdot)$ captures discrepancies between these two outputs. Consequently, the equation derives the empirical probability that distorting feature j causes the instance to cross the model's decision boundary.

This formulation allows us to measure the structural reliance of the model on specific features in a fully label-free manner, making it suitable for scenarios where the ground truth is unavailable or delayed.

3. Experimental Evaluation

This section evaluates the capability of the proposed concept drift detector UME_CDD to detect drift by observing feature importance. It also assesses how concept drift and the subsequent incremental adaptation phase influence the decision-making process of the machine learning model. Furthermore,

Feature name	Feature id	Feature name	Feature id
duration	f0	is_guest_login	f21
protocol_type	f1	count	f22
service	f2	srv_count	f23
flag	f3	serror_rate	f24
src_bytes	f4	srv_serror_rate	f25
dst_bytes	f5	rerror_rate	f26
land	f6	srv_rerror_rate	f27
wrong_fragment	f7	same_srv_rate	f28
urgent	f8	diff_srv_rate	f29
hot	f9	srv_diff_host_rate	f30
num_failed_logins	f10	dst_host_count	f31
logged_in	f11	dst_host_srv_count	f32
num_compromised	f12	dst_host_same_srv_rate	f33
root_shell	f13	dst_host_diff_srv_rate	f34
su_attempted	f14	dst_host_same_src_port_rate	f35
num_root	f15	dst_host_srv_diff_host_rate	f36
num_file_creations	f16	dst_host_serror_rate	f37
num_shells	f17	dst_host_srv_serror_rate	f38
num_access_files	f18	dst_host_rerror_rate	f39
num_outbound_cmds	f19	dst_host_srv_rerror_rate	f40
is_host_login	f20		

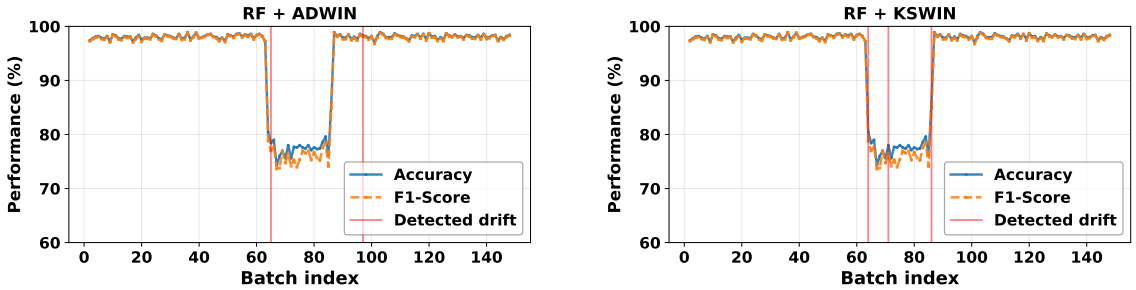
Table 1
Description of NSL-KDD features and their identifiers.

this section evaluates the proposed framework's ability to enable the online IDS to handle concept drift effectively and to accurately detect malicious activity, while minimizing labeling costs.

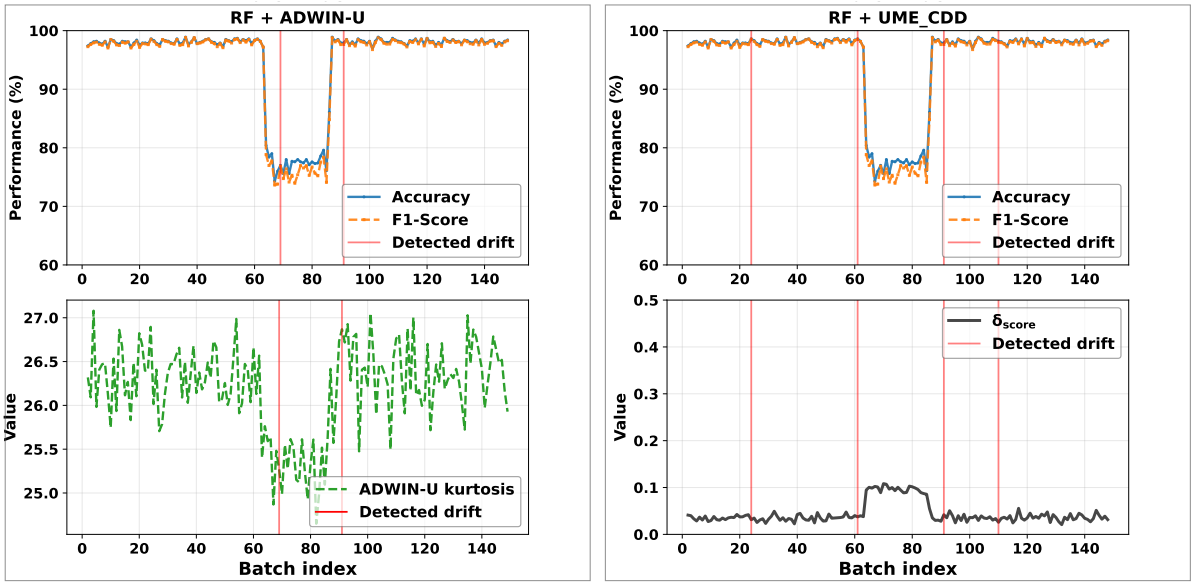
The dataset used to conduct these experiments is NSL-KDD [31], which is an enhanced version of the KDD CUP'99 dataset, designed to overcome its predecessor's main limitations. Specifically, NSL-KDD eliminates the extensive duplicate instances found in the original set and improves class distribution balance, thereby increasing the representation of minority samples in the test set. This improvement facilitates more reliable and comparable evaluations among Intrusion Detection systems, mitigating the risk of results biased by redundancy or the over-representation of specific attack types. The dataset comprises 41 network attributes (detailed in Table 1) alongside a label indicating the nature of the traffic (normal or attack). Although it is not a perfect representation of real-world networks, this dataset includes various types of benign and malicious traffic and is considered one of the few publicly available datasets exhibiting sudden concept drift [14]. These characteristics make it an excellent candidate for testing the performance of ML-based IDSs and evaluating online learning strategies under concept drift. The dataset is released in two parts: a training set and a test set. The test set contains new types of benign and malicious traffic not present in the training set. In this work, the training and test sets are interleaved. The resulting dataset is neither pre-processed further nor shuffled, in order to induce a sudden and recurring concept drift.

First, the performance over time of two static systems is evaluated to assess the performance degradation caused by concept drift. This provides a baseline for subsequent comparisons with adaptive online systems and allows for the evaluation of concept drift detectors in terms of true detections, false positives, and detection delay, as verified by their impact on system performance.

The static models compared are XGBoost [32] and Random Forest (RF) [33] with default hyperparameters. In combination with these models, two error-rate–based concept drift detectors are tested (ADWIN and KSWIN), along with two distribution-based detectors: ADWIN-U and the proposed UME_CDD. The experiments are conducted using the classic online approach, analyzing the dataset as a data stream in batches of size 1000. The static models are trained offline on the first batch of data and then evaluated



(a) Error rate-based concept drift detection: ADWIN on the left, KSWIN on the right.



(b) Distribution-based concept drift detection: ADWIN-U on the left with the kurtosis metric over number of batch, UME_CDD on the right with the δ_{score} over number of batch.

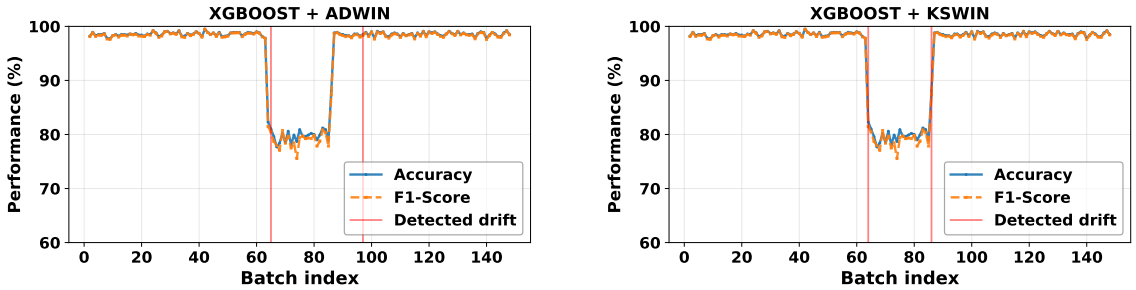
Figure 2: Evaluation of the impact of sudden and recurring concept drift on the performance of Random Forest static ML-model in terms of Accuracy and F1-score over number of batches and evaluation of concept drift detection using different techniques.

online.

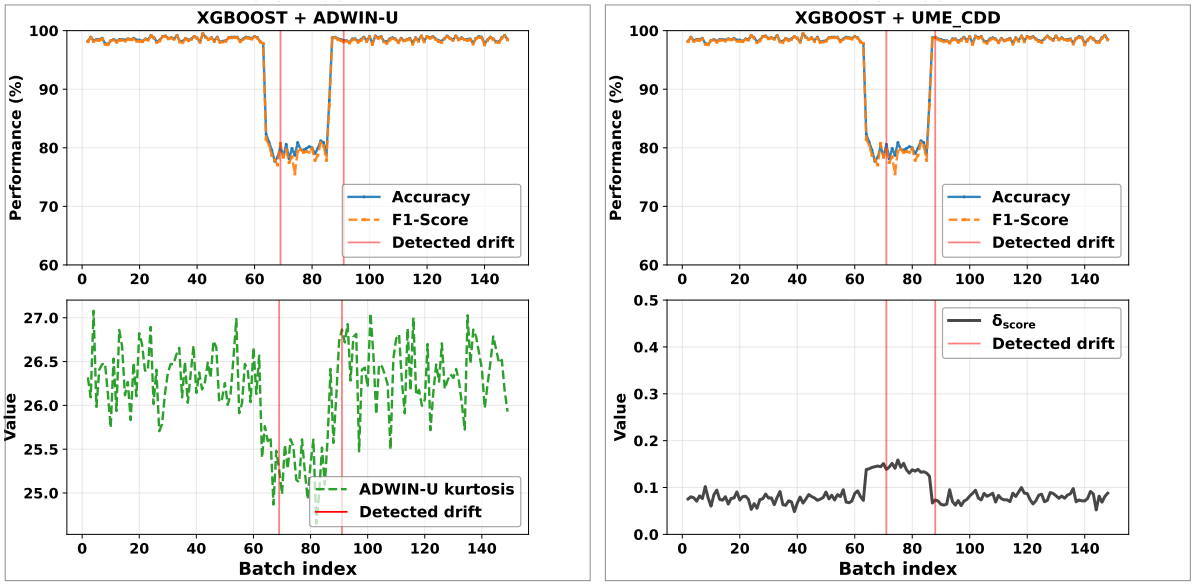
The same batch size (1000) is also used for ADWIN, KSWIN, and ADWIN-U, as well as for the B_{cc} and B_{new} batches of UME_CDD. The rest of the hyperparameters were left at their default values, as suggested by the authors. ADWIN-U is instantiated to analyze the kurtosis metric. As for UME_CDD, the internal KSWIN was instantiated with the default threshold used for the other detectors, a total window size of 15, and a new concept window size of 7. This differentiation is necessary because UME_CDD works by comparing dissimilarity values between batches, rather than operating in error-rate mode on each record within a batch. The choice of these values is empirical and involves a trade-off between correct identification of drift and the number of false alarms. These configurations will be used for the rest of the experiments described in this section.

Fig. 2 and 3 show the performance over time, in terms of Accuracy and F1-score, of the static models Random Forest and XGBoost, respectively. The figures also report concept drift detection performed using ADWIN and KSWIN in error-rate mode (Fig. 2a and Fig. 3a) and using ADWIN-U and UME_CDD in distribution-based mode (Fig. 2b and Fig. 3b). In the latter case, the evolution of the metric or score across batches is also reported to provide insight into the rationale behind each detection.

Both static models initially perform very well, until the data distribution undergoes significant changes compared to that of the first training batch (Concept A). Performance drops sharply when a sudden drift occurs around batch 65 (Concept B), then returns to levels similar to the pre-drift period starting from batch 85 (Concept A reappears).



(a) Error rate-based concept drift detection: ADWIN on the left, KSWIN on the right.



(b) Distribution-based concept drift detection: ADWIN-U on the left with the kurtosis metric over number of batch, UME_CDD on the right with the δ_{score} over number of batch.

Figure 3: Evaluation of the impact of sudden and recurring concept drift on the performance of XGBoost static ML model in terms of Accuracy and F1-score over number of batches and evaluation of concept drift detection using different techniques.

In both cases, ADWIN and KSWIN detect drift when the performance of the two models varies sharply. More specifically, ADWIN shows a delay in detection compared to KSWIN. KSWIN detects drift faster than ADWIN but is more prone to false positives, such as a false detection during Concept B when monitoring the RF model. This is also caused by the RF model's more variable performance compared to XGBoost during the drift period. Although their drift detection performance is good, these detectors rely on 100% of the ground-truth labels to monitor model performance.

ADWIN-U and UME_CDD show their ability to overcome the labeling cost problem by detecting concept drift in an unsupervised manner. As shown in Fig. 2b and Fig. 3b, the kurtosis metric values computed by ADWIN-U do not vary over time, as they are derived directly from the statistical distribution of the features and are therefore independent of the machine learning model. The drift is detected correctly with a slight delay. A first limitation is the significant oscillation of this metric, which, combined with its non-normalized value range, makes drift detection unreliable when using a static threshold. A crucial aspect is the multiple statistical testing performed by ADWIN on the one-dimensional distribution of kurtosis values.

Another limitation concerns the explainability of concept drift. Unlike error-rate-based methods, this type of detection is not tied to model performance and provides no indication of the possible underlying causes, reducing the transparency of the process.

UME_CDD overcomes these limitations. The δ_{score} is inherently normalized between 0 and 1 and exhibits stable behavior, as it depends only on the subset of features most influential to the model's

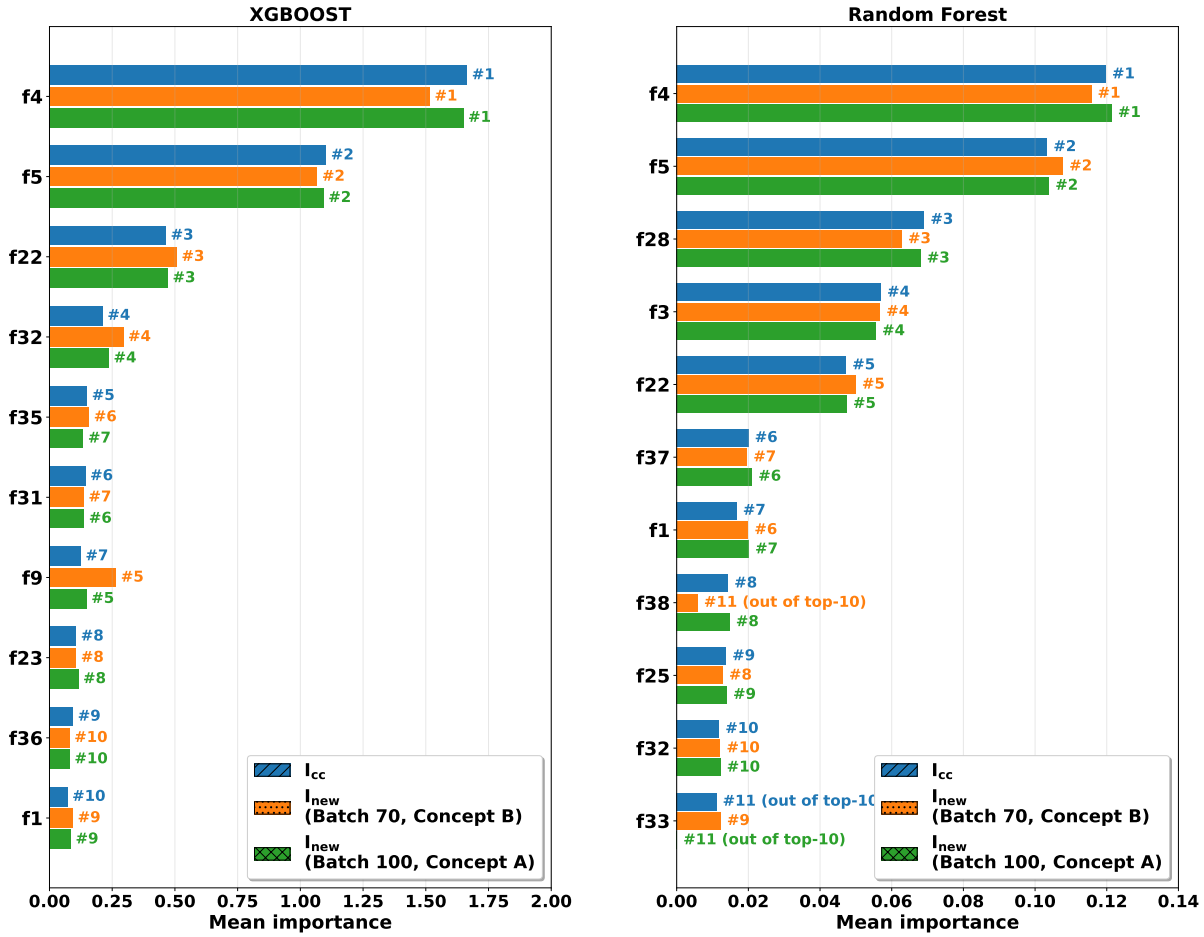


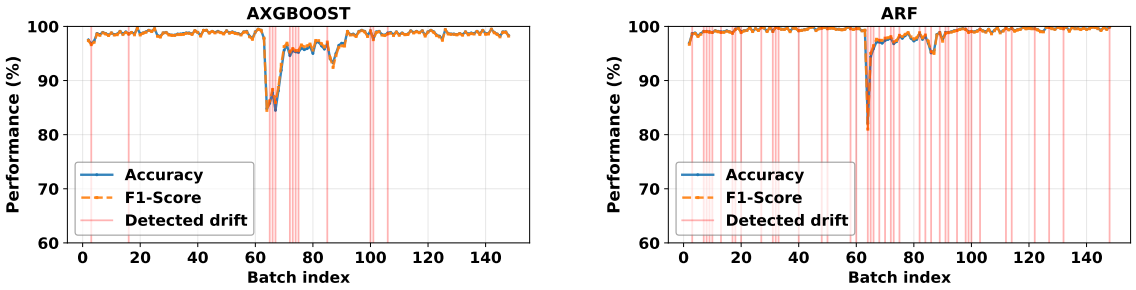
Figure 4: Explanation of the impact of concept drift through comparison of I_{cc} and two different I_{new} features importance profiles of UME_CDD for the two static machine learning models.

decisions. Moreover, noisy records do not significantly affect the computation of I_{new}, as they have limited influence on the feature importance calculated over the entire batch. It is important to note that the temporal evolution of the δ_{score} varies depending on the specific static model used, as shown by the comparison between Fig. 2b and Fig. 3b. This variation reflects not only differences in the data distribution but also how each model computes I_{cc} and I_{new} for each batch. Moreover, UME_CDD enables an explicit characterization of concept drift by examining how the most important features affect the model's decisions. This is achieved by maintaining I_{cc} and I_{new} at each time step. As a result, it becomes possible to track their temporal evolution or to inspect them when UME_CDD detects a drift, in order to understand the nature of the underlying shift.

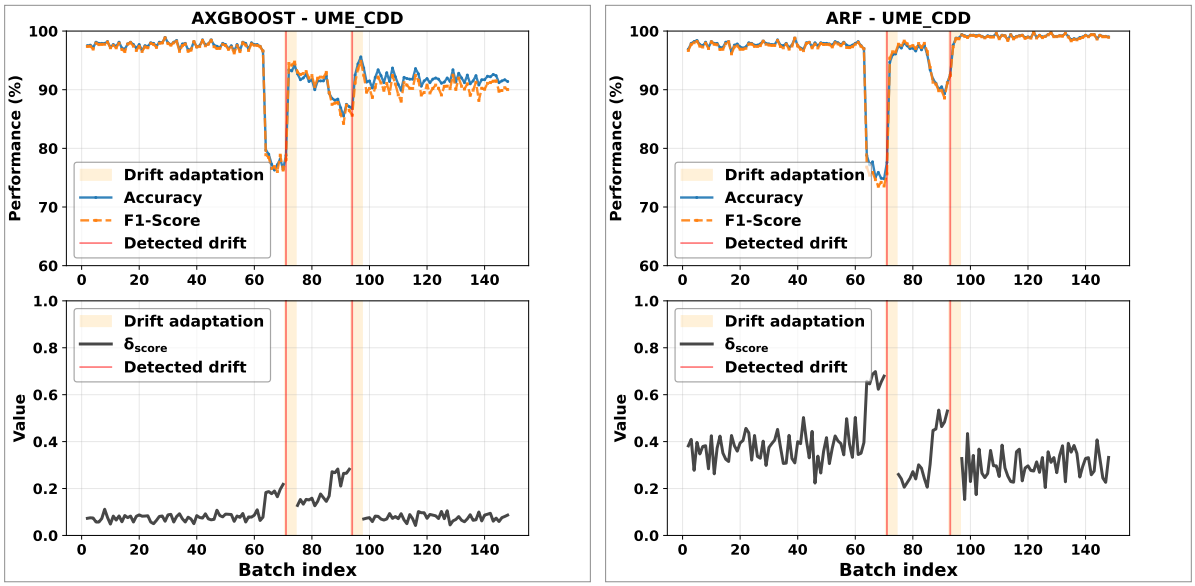
Figure 4 compares I_{cc} with two distinct I_{new} instances, taken from the drift period (Concept B) and the recurring drift period (Concept A), for both machine learning models under investigation. For both models, although on different scales, the feature importance profiles of I_{cc} and I_{new} from batch 100 show substantial similarity in average values and feature importance rankings. In contrast, the differences between I_{cc} and I_{new} from batch 70 are pronounced. In the case of the Random Forest model, it is also evident that, following the drift, some features change so significantly in importance that they no longer appear among the top-N.

The second part of the experiments compares the proposed online framework (Alg. 1) with two adaptive online systems representative of the state of the art: AXGBoost and Adaptive Random Forest.

For a direct comparison with the performance of static systems, the experiments were conducted again on the NSL-KDD dataset using a batch size of 1,000. AXGBoost and Adaptive Random Forest (ARF) were implemented with the default hyperparameters recommended by their respective authors under sudden-drift conditions. The proposed framework was evaluated with both AXGBoost and



(a) AXGBoost (in the left) and ARF (in the right) with their integrated concept drift error-rate detector (ADWIN).



(b) Proposed framework with AXGBoost (in the left) and ARF (in the right) as adaptive machine learning models (AM) with UME_CDD concept drift detetcor.

Figure 5: Comparison between **online adaptive ML model** and **proposed framework** in terms of Accuracy and F1-score over number of batches and evaluation of concept drift detection using different techniques.

ARF as adaptive machine learning models (AMs), using an adaptation period of three batches. The configuration of UME_CDD corresponds to the one previously described. The offline phase consisted of training the models on the first data batch, followed by the online phase.

Figure 5a illustrates the temporal performance of the systems. Both state-of-the-art methods exhibit strong performance in the absence of drift, followed by an abrupt degradation and a subsequent rapid recovery, up until the occurrence of the recurring drift. The second performance drop, however, is less pronounced than the first, as the incremental learning strategy enables the ensemble to retain part of the original knowledge from Concept A, partially integrating it with the knowledge associated with Concept B.

Both models exhibit numerous false drift detections due to two main factors. First, incremental training occurs after every batch, causing slight but continuous fluctuations in performance due to unnecessary updates on batches with varying noise levels. Second, ADWIN is intentionally configured with a highly sensitive threshold to detect drift as quickly as possible. For these systems, the presence or absence of drift merely triggers different adaptation strategies.

This design choice results in a significant limitation with respect to labeling cost, as both models require 100% of the ground-truth labels to operate in real time.

As shown in Fig. 5b, the proposed framework overcomes this limitation by maintaining performance comparable to that of state-of-the-art adaptive online systems, while achieving a drastic reduction in labeling cost (only 5.4%). When drift occurs, UME_CDD successfully detects it, with a slight delay, once again demonstrating the effectiveness of the proposed approach. This enables the framework to trigger

Online system	Labeling cost	Mean Accuracy	Mean F1-score
AXGBOOST	100%	0.978	0.978
ARF	100%	0.989	0.989
AXGBOOST - UME_CDD	5.4%	0.933	0.928
ARF - UME_CDD	5.4%	0.966	0.965

Table 2
Number of ground truth labels requested and mean accuracy achieved by online systems after pretraining - NSL-KDD dataset.

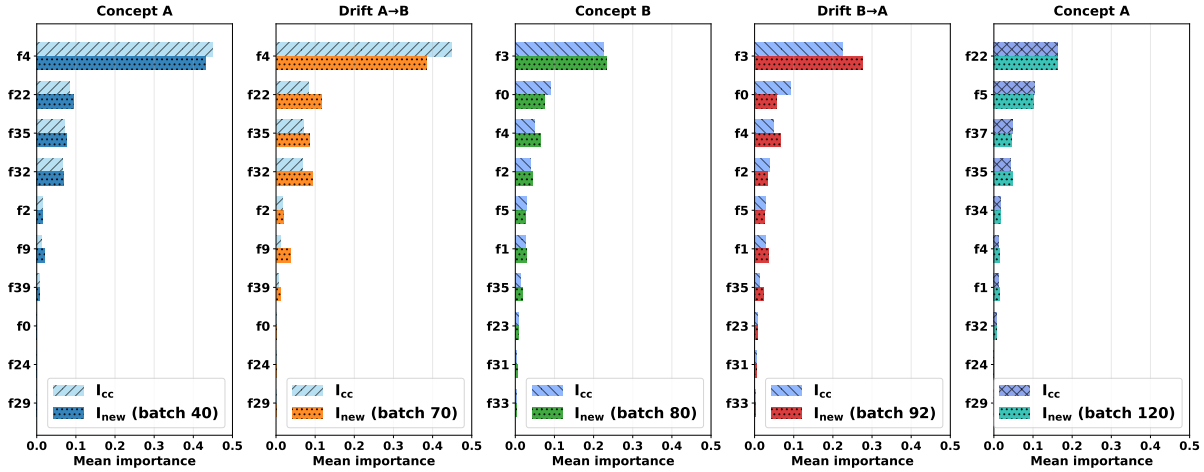


Figure 6: Explanation of the impact of concept drift and adaptation strategy on model decisions (Proposed framework with AXGBoost) through comparison of I_{cc} and I_{new} in different periods: pre-drift (Concept A), during drift (Drift A → B), after first adaptation period (Concept B), during recurring drift (Drift B → A), and after second adaptation period (Concept A).

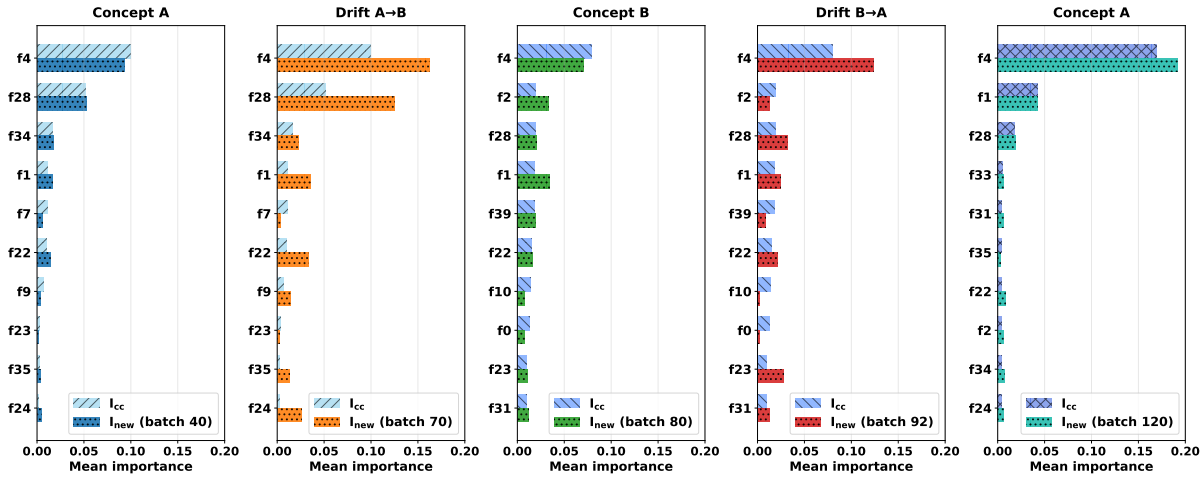


Figure 7: Explanation of the impact of concept drift and adaptation strategy on model decisions (Proposed framework with ARF) through comparison of I_{cc} and I_{new} in different periods: pre-drift (Concept A), during drift (Drift A → B), after first adaptation period (Concept B), during recurring drift (Drift B → A), and after second adaptation period (Concept A).

a temporally bounded adaptation phase that still allows the adaptive models to recover from drift in both scenarios.

Furthermore, the adaptation period parameter can be tuned to achieve the desired trade-off between labeling cost and post-adaptation performance. Table 2 reports the final average performance of the two compared systems and of the proposed framework, along with the associated labeling cost.

In addition, unlike traditional error-rate–based drift detection methods used in the compared systems, the proposed framework allows observation of both the impact of concept drift on model decisions and the changes in the decision-making process after adaptation. Fig. 6 and Fig. 7 illustrate this aspect for AXGBoost and ARF, respectively, by comparing the I_{cc} and I_{new} components of UME_CDD across different time periods.

As observed in Fig. 4, the feature importance profiles remain highly similar before the drift event (Concept A). A sharp divergence occurs immediately after the drift (Drift A → B), when the model has not yet adapted. After the adaptation period, once UME_CDD updates I_{cc}, the profiles become similar again. However, a clear distinction remains between I_{cc} for Concept A and Concept B: the most influential features change, and even features remaining in the top ranks exhibit different impact levels. Following adaptation, the model adjusts its decision-making to reflect the shifted data distribution caused by the drift.

A mirrored behavior is observable after the recurring drift (Drift B → A). Finally, after the second adaptation period, the I_{cc} profile again resembles the initial one, while still reflecting the influence of both concepts integrated within the ensemble.

4. Conclusions and future work

This work presented an online framework for Intrusion Detection Systems (IDSs) designed to address two major challenges in online learning under concept drift for network security: the high labeling cost and the lack of explainability in existing drift detection and adaptation approaches.

To overcome these limitations, we propose the Unsupervised Multidimensional Explainability-Driven Concept Drift Detector (UME_CDD), capable of identifying drift without ground truth labels while providing insights into its impact on the decision-making process of the underlying machine learning model. Additionally, we proposed an online adaptation strategy that minimizes labeling costs, ensuring that the IDS can maintain high detection performance even in dynamically evolving network environments.

Extensive experimental evaluations conducted on multiple real-world network datasets exhibiting different types of concept drift demonstrated the effectiveness of the proposed framework. The results show that our approach achieves detection and adaptation performance comparable to state-of-the-art methods while significantly reducing labeling cost. The UME_CDD successfully detected drift events and provided meaningful explanations of their influence on model predictions. In addition, the explainability-driven analysis enables a deeper understanding of how the model's behavior evolves over time in response to drift, improving transparency and trust in IDS decision-making.

Future work could involve testing the proposed framework and UME_CDD in other cybersecurity applications that use machine learning models on high-dimensional datasets and may suffer from concept drift, such as spam or malware detection. This would verify the generality of the proposed work.

The framework could be improved by incorporating a more complex adaptation strategy to perform different incremental training depending on the type and intensity of drift detected by UME_CDD. In addition, an outlier detection model could be included to filter the most dissimilar records in each batch. This would prevent noisy or adversarial records from biasing the calculation of feature importance and, consequently, the detection of drift.

Acknowledgments

This work was partially supported by the AMELIS project, within the project FAIR (PE0000013), and by the ADELE project, within the project SERICS (PE00000014), both under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Declaration on Generative AI

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

References

[1] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Transactions on Emerging Telecommunications Technologies 32 (2021) e4150.

[2] V. Agate, F. M. D'Anna, A. De Paola, P. Ferraro, G. Lo Re, M. Morana, A behavior-based intrusion detection system using ensemble learning techniques., in: CEUR Workshop Proceedings, 6th Italian Conference on Cybersecurity, ITASEC 2022, 2022, pp. 207–218.

[3] M. Soltani, B. Ousat, M. J. Siavoshani, A. H. Jahangir, An adaptable deep learning-based intrusion detection system to zero-day attacks, Journal of Information Security and Applications 76 (2023) 103516.

[4] A. De Paola, S. Drago, P. Ferraro, G. Lo Re, Detecting zero-day attacks under concept drift: An online unsupervised threat detection system, in: CEUR Workshop Proceedings, 8th Italian Conference on Cybersecurity, ITASEC 2024, 2024.

[5] H. Choi, M. Kim, G. Lee, W. Kim, Unsupervised learning approach for network intrusion detection system using autoencoders., Journal of supercomputing 75 (2019).

[6] A. Augello, A. De Paola, D. Giosuè, G. Lo Re, SF-AE: Split federated autoencoder for unsupervised IoT intrusion detection, in: Proceedings of Tenth International Congress on Information and Communication Technology, Springer Nature Singapore, Singapore, 2025, pp. 371–381. doi:10.1007/978-981-96-6432-0_28.

[7] E. Min, J. Long, Q. Liu, J. Cui, Z. Cai, J. Ma, Su-ids: A semi-supervised and unsupervised framework for network intrusion detection, in: International Conference on Cloud Computing and Security, Springer, 2018, pp. 322–334.

[8] V. Agate, A. De Paola, S. Drago, P. Ferraro, G. Lo Re, Enhancing IoT network security with concept drift-aware unsupervised threat detection, in: 2024 IEEE Symposium on Computers and Communications (ISCC), IEEE, 2024, pp. 1–6.

[9] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, L. Cavallaro, {TESSERACT}: Eliminating experimental bias in malware classification across space and time, in: 28th USENIX security symposium (USENIX Security 19), 2019, pp. 729–746.

[10] V. Agate, S. Drago, P. Ferraro, G. Lo Re, Anomaly detection for reoccurring concept drift in smart environments, in: 2022 18th International Conference on Mobility, Sensing and Networking (MSN), IEEE, 2022, pp. 113–120.

[11] K. He, D. D. Kim, M. R. Asghar, Adversarial machine learning for network intrusion detection systems: A comprehensive survey, IEEE Communications Surveys & Tutorials 25 (2023) 538–566.

[12] F. Concone, G. L. Re, M. Morana, C. Ruocco, Twitter spam account detection by effective labeling, CEUR Workshop Proceedings 2315 (2019).

[13] Z. Lin, Y. Shi, Z. Xue, Idsgan: Generative adversarial networks for attack generation against intrusion detection, in: Pacific-asia Conference on Knowledge Discovery and Data Mining, Springer, 2022, pp. 79–91.

[14] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Transactions on Knowledge and Data Engineering 31 (2018) 2346–2363.

[15] A. Augello, A. De Paola, G. Lo Re, Hybrid multilevel detection of mobile devices malware under concept drift, Journal of Network and Systems Management 33 (2025).

[16] S. C. Hoi, D. Sahoo, J. Lu, P. Zhao, Online learning: A comprehensive survey, Neurocomputing 459 (2021) 249–289.

[17] G. Andresini, A. Appice, C. Loglisci, V. Belvedere, D. Redavid, D. Malerba, A network intrusion

detection system for concept drifting network traffic data, in: International Conference on Discovery Science, Springer, 2021, pp. 111–121.
[18] F. Camarda, A. De Paola, S. Drago, P. Ferraro, G. Lo Re, Managing concept drift in online intrusion detection systems with active learning, in: CEUR WORKSHOP PROCEEDINGS, volume 3962, CEUR-WS, 2025.
[19] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, L. Cavallaro, Insomnia: Towards concept-drift robustness in network intrusion detection, in: Proceedings of the 14th ACM workshop on artificial intelligence and security, 2021, pp. 111–122.
[20] J. Montiel, R. Mitchell, E. Frank, B. Pfahringer, T. Abdessalem, A. Bifet, Adaptive xgboost for evolving data streams, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.
[21] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, Machine Learning 106 (2017) 1469–1495.
[22] V. Agate, A. De Paola, P. Ferraro, G. Lo Re, MIDES: A multi-layer intrusion detection system using ensemble machine learning, International Journal of Intelligent Networks 6 (2025) 204 – 223. doi:10.1016/j.ijin.2025.09.001.
[23] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, 2007, pp. 443–448.
[24] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth International Workshop on Knowledge Discovery from Data Streams, volume 6, 2006, pp. 77–86.
[25] C. Raab, M. Heusinger, F.-M. Schleif, Reactive soft prototype computing for concept drift streams, Neurocomputing 416 (2020) 340–351.
[26] D. N. Assis, V. M. Souza, Adwin-u: adaptive windowing for unsupervised drift detection on data streams: Dn assis, vma souza, Knowledge and Information Systems (2025) 1–30.
[27] Y. Lee, Y. Lee, E. Lee, T. Lee, Explainable artificial intelligence-based model drift detection applicable to unsupervised environments., Computers, Materials & Continua 76 (2023).
[28] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Advances in Neural Information Processing Systems 30 (2017).
[29] A. Fisher, C. Rudin, F. Dominici, All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously, Journal of Machine Learning Research 20 (2019) 1–81.
[30] A. Datta, S. Sen, Y. Zick, Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems, in: 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 598–617. doi:10.1109/SP.2016.42.
[31] R. Bala, R. Nagpal, A review on kdd cup99 and nsl nsl-kdd dataset., International Journal of Advanced Research in Computer Science 10 (2019).
[32] B. S. Bhati, G. Chugh, F. Al-Turjman, N. S. Bhati, An improved ensemble based intrusion detection technique using xgboost, Transactions on emerging telecommunications technologies 32 (2021) e4076.
[33] P. A. A. Resende, A. C. Drummond, A survey of random forest based methods for intrusion detection systems, ACM Computing Surveys (CSUR) 51 (2018) 1–36.