Vol. 8, No. 1, February 2026

Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Trinity-Controller ADWIN: An Accuracy Guided Sensitivity Control Framework for Streaming Intrusion Detection

Rodney Buang Sebopelo

North—West University, Potchefstroom Campus, Potchefstroom, South Africa

Received:
November 25, 2025
Revised:
January 14, 2026
Accepted:
January 29, 2026
Published:
February 20, 2026

Corresponding Author:
Author Name*:
Rodney Buang Sebopelo
Email*:
rodney.sebopelo@nwu.ac.za

DOI:
10.63158/journalisi.v8i1.1421

© 2026 Journal of Information Systems and Informatics. This open access article is distributed under a (CC-BY License)

Abstract. Concept drift can severely undermine the reliability of streaming Intrusion Detection Systems (IDS), especially in realistic network traffic where changes are gradual, recurring, and often masked by noise and class imbalance. Widely used statistical drift detectors such as ADWIN provide theoretical guarantees, yet in practice they can exhibit sensitivity oscillations, delayed adaptation under subtle drift, and disruptive reset behavior that leads to prolonged performance dips. This paper presents Trinity-Controller ADWIN, a unified drift-management framework that fuses three complementary signals: a Volatility Controller (VC) for statistically grounded drift detection, an Adaptive Rate Controller (ARC) that dynamically regulates ADWIN sensitivity, and a Performance-Based Controller (PBC) that monitors an Exponential Moving Average (EMA) of online accuracy to detect sustained model degradation. The proposed framework is evaluated using a Hoeffding Adaptive Tree classifier on a time-ordered streaming reconstruction of CICIDS2017, reflecting realistic temporal drift patterns. Across multiple drift regions, Trinity-Controller ADWIN achieves higher long-horizon accuracy stability, faster post-drift recovery, and fewer unnecessary resets than fixed ADWIN, VC-only, and VC+ARC baselines. Notably, in several drift segments the framework preserves post-drift accuracy above 90% of baseline while demonstrating near-zero recovery delay, indicating that adaptation occurs with minimal disruption. Overall, the results show that combining statistical drift evidence with direct performance-aware feedback yields a more robust and operationally reliable streaming IDS under evolving traffic conditions.

Keywords: Concept drift; Streaming IDS; ADWIN; EMA accuracy; Online learning

1. INTRODUCTION

Intrusion Detection Systems (IDS) deployed in operational networks must sustain reliable detection under continuously evolving traffic conditions [1, 2]. In real deployments, changes in user behavior, application usage, routing policies, background services, and attacker strategies introduce multiple forms of concept drift that progressively degrade detection performance. This challenge is amplified in realistic and heterogeneous benchmarks such as CICIDS2017, where the diversity of traffic patterns and attack behaviors creates a highly variable environment that stresses generalization and exposes the brittleness of static or fixed-sensitivity configurations [4, 5]. Under these conditions, drift can be abrupt, gradual, recurring, or partially obscured by noise, meaning that an IDS must adapt continuously without becoming unstable or overly reactive.

Adaptive drift detectors such as Adaptive Windowing (ADWIN) offer a principled way to detect statistical changes in streaming data by monitoring shifts in data distributions over time [3]. However, in IDS settings, statistical drift detection alone often falls short in practice. ADWIN-based methods can suffer from slow post-drift recovery, instability under noisy traffic, and sensitivity to parameter settings, which becomes especially evident when confronting the diverse characteristics present in CICIDS2017 [4, 5]. Even when enhanced with adaptive mechanisms, the underlying reset-and-relearn behavior can lead to performance dips and extended recovery periods after drift events, particularly for subtle or gradual drift that does not produce a strong statistical signal but still undermines classifier effectiveness.

Prior work has sought to improve ADWIN's responsiveness and stability through sensitivity regulation. The Adaptive ADWIN framework, which combines a Volatility Controller (VC) with an Adaptive Rate Controller (ARC), improves adaptability by adjusting drift sensitivity in response to stream behavior [6, 7]. Despite these improvements, notable limitations remain: the detector may still fail to respond adequately to subtle or slowly evolving drift, may become dysfunctional under highly variable streams, and can trigger disruptive resets that degrade accuracy and prolong recovery time [8, 9]. Consequently, there remains a gap for drift management in streaming IDS that is both responsive to diverse drift types and stable under noise, while also avoiding unnecessary resets and maintaining performance over long horizons.

A key motivation of this work is the observation that online performance degradation—such as a sustained drop in accuracy—can provide an earlier and more operationally meaningful signal of model decay than distributional drift indicators alone [10]. While statistical change detection is valuable for identifying shifts in the data stream, it does not necessarily indicate whether those shifts are harming detection performance in a way that warrants intervention. Accuracy-based monitoring can therefore complement ADWIN's statistical checks by signaling when the model's predictive capability is deteriorating, enabling earlier corrective actions and reducing reliance on sensitivity tuning alone as the primary adaptation lever [11]. By integrating performance signals into the drift management loop, adaptive IDS can achieve faster and more stable reactions to both sudden disruptions and gradual degradation, improving robustness across drift regimes [12, 13].

Related work reflects strong progress but also highlights the need for more performance-aware drift handling in IDS. Farah Jemili et al. [4] introduced DDM-ORF, combining the Drift Detection Method (DDM) with Online Random Forest learning, and reported strong accuracy compared with traditional approaches. Usman Ali et al. [14] proposed AEDDM, a semi-supervised autoencoder-based drift detection approach intended to detect drift without requiring truth labels, and showed that it can detect real drift scenarios across synthetic and real datasets. Day Shang et al. [10] studied neural models for streaming data and pointed out that many existing approaches cannot distinguish concept drift from novelty, which can lead to inefficient allocation of maintenance resources and suboptimal adaptation decisions. Alexander Kraus et al. [15] proposed CV4CDD-4D for automated drift detection across sudden and gradual changes using supervised learning on large event logs and demonstrated improvements in robustness and reliability under noise. Piotr Porwik et al. [16] presented a feature-rank-based drift strategy that tracks changes in important feature rankings across chunks, yielding useful drift indicators. However, despite these advances, most existing approaches still remain heavily dependent on statistical distribution changes and rely on static or blunt reset strategies when drift is declared, which can be problematic for noisy network streams and slow-moving drift.

Despite recent improvements in adaptive drift detection, ADWIN-based methods—even when paired with adaptive sensitivity mechanisms—continue to exhibit delayed recovery

under gradual drift and instability under noisy conditions [17, 18]. More importantly, relatively few studies explicitly treat real-time performance degradation as a first-class signal for drift management in streaming IDS, even though operational IDS objectives are ultimately performance-driven. This gap motivates the need for an adaptive framework that fuses statistical drift evidence with direct indicators of classifier degradation, thereby enabling timely adaptation while minimizing unnecessary resets and improving stability.

To address these limitations, this work proposes Trinity-Controller ADWIN, a unified drift-management framework that jointly leverages statistical drift cues, adaptive sensitivity regulation, and accuracy-driven feedback. The novelty of the proposed approach lies in introducing a Performance-Based Controller (PBC) that uses an Exponential Moving Average (EMA) of online accuracy to detect early performance deterioration and to guide drift response more reliably than statistical signals alone. By integrating VC, ARC, and PBC into one coordinated architecture, the framework aims to achieve faster adaptation, reduce false or unnecessary resets, and sustain stable long-term operation under both abrupt and gradual drift scenarios in realistic IDS streams.

This study makes four primary contributions. First, it proposes a novel Performance-Based Controller (PBC) that leverages the EMA of online accuracy to detect early performance degradation and provide a direct, model-centric adaptation signal. Second, it introduces the Trinity Controller ADWIN framework that integrates statistical drift detection via VC, adaptive sensitivity control via ARC, and performance-driven feedback via PBC into a unified adaptive architecture. Third, it demonstrates that the proposed framework can exhibit near-zero sample recovery behavior in several drift regions, supporting faster adaptation than traditional ADWIN-based approaches. Fourth, extensive experiments on time-ordered CICIDS2017 streams confirm improved stability, fewer unnecessary resets, and stronger robustness under both abrupt and gradual drift scenarios.

The remainder of this paper is organized as follows. The Methods section presents the overall research methodology, including the dataset construction, streaming setup, learning model (Hoeffding Adaptive Tree), the proposed Trinity Controller ADWIN framework, controller mechanisms, reset policy, experimental setup, and evaluation

metrics. The Results and Discussion section reports the experimental findings, comparative performance analysis, ablation study, and discusses the implications and limitations of the proposed approach. Finally, the Conclusion section summarizes the key contributions of this work and outlines directions for future research.

2. Methods

2.1. Dataset and Streaming Construction

Intrusion detection must be performed on a continuous, data-evolving stream in real-world network environments. The CICIDS2017 dataset is reorganized into a time-ordered, labelled data stream preserving the natural traffic flow sequence [19, 20]. The streaming representation exposes the system to dynamic network changes, attack intensity, and benign behavior, thereby creating multiple forms of concept drift [21]. This work employs the Hoeffding Adaptive Tree (HAT) algorithm and an incremental decision tree designed for streaming conditions [22]. It integrates drift-aware split mechanisms and provides competitive performance with low computational cost, making it the suitable model for high-velocity IDS data. The model updates incrementally as new instances arrive and adapt to changes without full retraining [23, 24].

2.2. Hoeffding Adaptive Tree (HAT)

The experiment uses HAT as the primary classifier. HAT is selected based on the capability to support the incremental online updates (e.g., one instance at a time), support the ADWIN-based subtree resizing, be capable of studying drift based on different reset policies, and be the widely used model for research in streaming IDS [25, 26]. The splits are performed using the Hoeffding bound, as shown in Equation 1 [27]:

$$\epsilon = \sqrt{\frac{R^2 \ln\left(\frac{1}{\delta}\right)}{2n}} \quad (1)$$

Where R denotes the range reward and is considered the number of observations.

Adaptive nodes such that HAT maintains the alternate subtrees based on drift signals and learning behavior with Trinity Controller ADWIN based on the following [28, 29]: 1) VC performs the detection of the distribution drift. 2) ARC dynamically adjusts the ADWIN,

3) PCB triggers a full/partial reset based on the EMA accuracy degradation, and 4) HAT responds by swapping alt subtrees, pruning, or restarting the root nodes, which allows the precise measurement of the adaptation time and stability [30, 31].

2.3. Trinity—Controller ADWIN Framework

As shown in Figure 1, the process starts with an incoming data stream, which is processed using the Hoeffding Adaptive Tree classifier [33]. The predictions from the classifiers are evaluated using the accuracy monitor, which computes the EMA of online accuracy. Then, three controllers operate in parallel: VC uses ADWIN to detect the statistical drift on distributional changes, ARC adjusts the ADWIN's sensitivity based on the drift frequency, and PBC evaluates the EMA accuracy to identify the performance degradation. Each controller triggers a signal directly to a streaming reset policy, which performs the decision based on the model reset, update of, and partial reset. Then the system incrementally updates the HAT and ADWIN, completing the adaptive loop.

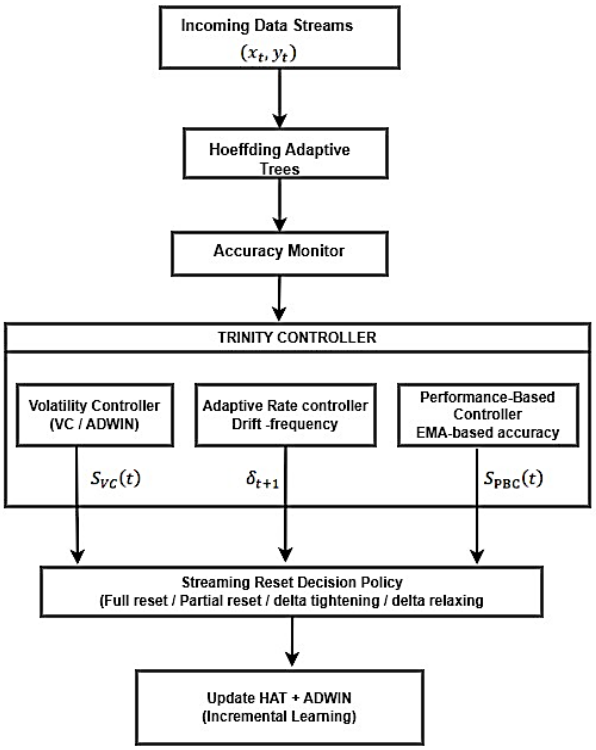

Figure 1. Trinity-Controller ADWIN Framework

2.4. Framework Process Operation

The Trinity Controller ADWIN Framework operates as a closed loop, streaming pipeline that is adaptive, where the incoming CIC IDS traffic instances pass through the

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

preprocessing and are classified by the HAT. The outcome feeds two parallel monitoring components: ADWIN's statistical detector employed by the VC and EMA used by PBC. The ARC evaluates the frequency of the ADWIN drift events and proposes the adjustment sensitivity to stabilize. The VC, ARC, and PBC trigger the signal to the decision unit, which is responsible for the corrective actions, such as partial reset, tightening, loosening, or executing a full reset while both accuracy degradation and drift evidence align. The chosen action is applied back to HAT and ADWIN to achieve faster adaptation, smoother drift recovery, and improved robustness against the noise. The loop is continued per instance, providing a continuous self-adjusting intrusion detection pipeline.

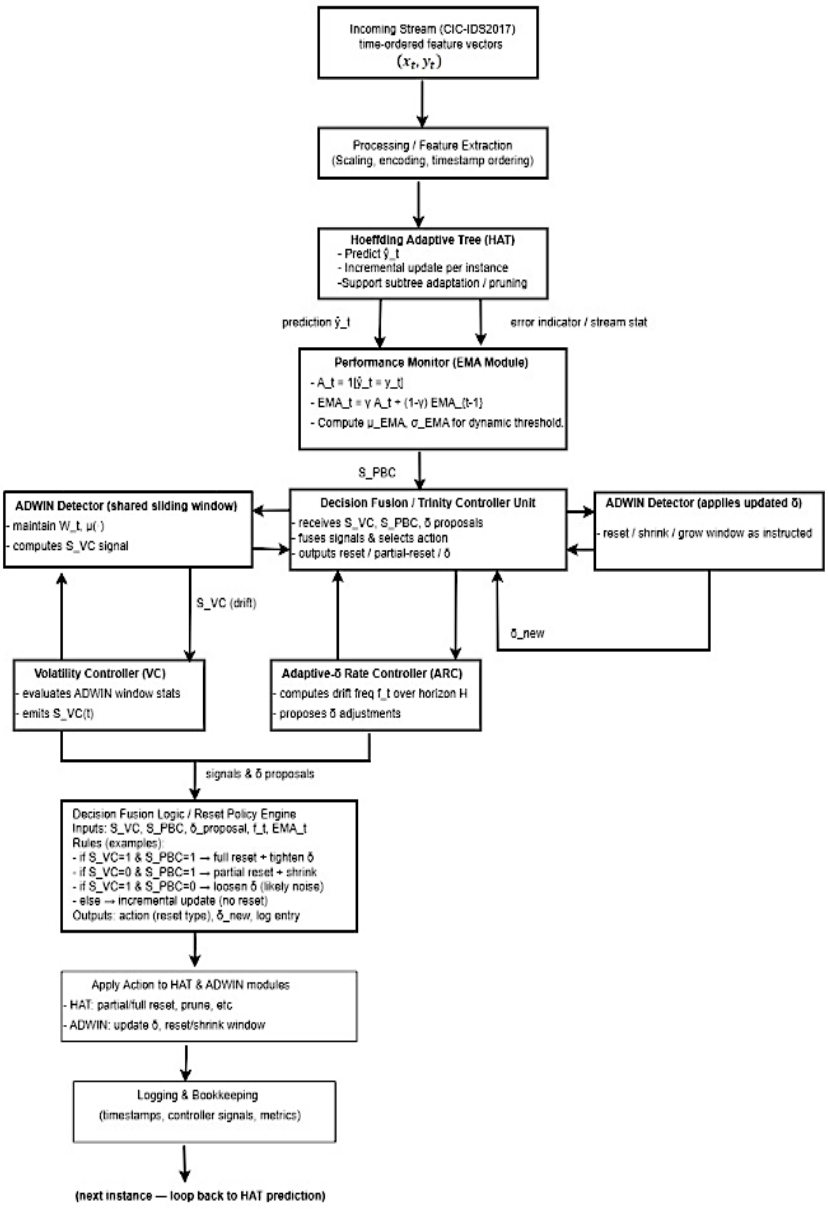

Figure 2. Trinity—Controller ADWIN Workflow

Figure 3 illustrates the overall research methodology employed in this study. The process begins with preparing the CICIDS2017 dataset into a time-ordered streaming format. The stream is then processed using a Hoeffding Adaptive Tree classifier integrated with different drift-handling strategies (Fixed ADWIN, VC only, VC+ARC, and the proposed Trinity Controller). Each configuration is evaluated using online prequential metrics, including accuracy, stability, recovery time, false alarms, and adaptation time. The collected performance streams are subsequently analyzed to compare robustness, responsiveness, and stability across methods.

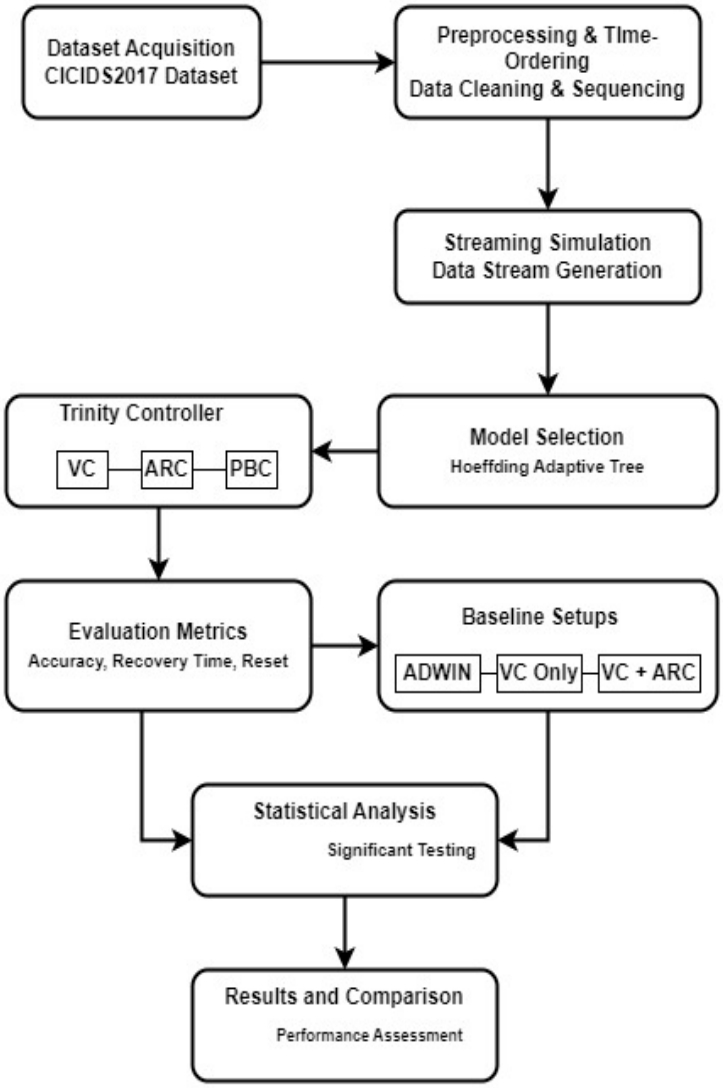

Figure 3. Research Methodology Flow

Algorithm 1. Trinity—Controller ADWIN

```
Initialize classifier M, ADWIN with δ, EMA accuracy A
For each stream instance (x_t, y_t):
    ŷ_t ← M.predict(x_t)
    a_t ← 1 if ŷ_t = y_t else 0
   A ← α·a_t + (1−α)·A
    D ← ADWIN.detect(drift_stat(x_t))
  δ ← ARC.adjust(δ, drift_frequency)
    P ← (A < dynamic_threshold)
    if D and P:
        reset ADWIN and M
       tighten δ
    else if P and not D:
        partial_reset M
        loosen δ
    else if D and not P:
        loosen δ
    M.learn(x_t, y_t)
    End For
```

2.5. Controllers

The Trinity Controller ADWIN Framework merges three control strategies—Value Controller (VC), Adaptive Rate Controller (ARC), and Performance Based Controller (PBC)—to reduce time adaptation, enhance drift responsiveness, and stabilize the sensitivity adjustments during IDS streaming operation [32]. Given an incoming stream of data calculated as shown in Equation 2.

$$\mathcal{D} = \{(x_t, y_t)\}_{t=1}^{T}, \tag{2}$$

Where x_t denotes the feature vector and y_t the label. HAT produces a prediction $\hat{y}_t$.

The triple controllers analyze the drift frequency, performance degradation, and distributional changes. Their outputs determine whether to 1) relax the ADWIN sensitivity

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

parameter δ, 2) trigger a full or partial reset model, and 3) incrementally update the classifier.

1) Volatility Controller (VC)

The VC monitors distributional drift by evaluating the difference between ADWIN's two subwindows. Consider the ADWIN's sliding window at time t, and $W_t^{(0)}$ and $W_t^{(1)}$ denotes the right and left during the detection of drift. The ADWIN triggers a drift when calculated as shown in Equation 3.

$$\left|\mu(W_t^{(0)}) - \mu(W_t^{(1)})\right| > \epsilon_\delta, \tag{3}$$

Where ϵ_δ denotes the ADWIN's statistic derived from the confidence bound calculated as shown in Equation 4 and 5.

$$\epsilon_\delta = \sqrt{\frac{1}{2m}} \ln\left(\frac{4}{\delta}\right), \tag{4}$$

and

$$\text{m} = \frac{\left|W_t^{(0)}||W_t^{(1)}\right|}{|W_t|}. \tag{5}$$

The VC is a statistical backbone that detects abrupt or strong events using ADWIN's theoretical guarantees. The volatility controller signal is defined calculated as shown in Equation 6.

$$S_{VC}(t) = \begin{cases} 1, if\ ADWIN\ detects\ drift\ at\ t, \\ 0, \qquad\qquad otherwise. \end{cases} \tag{6}$$

2) Adaptive–δ Rate Controller (ARC)

The ARC dynamically adjusts the ADWIN parameter δ based on the analysed drift frequency. f_t denotes the frequency of the drift over a sliding horizon H represented calculated as shown in Equation 7.

$$f_t = \frac{1}{H}\sum_{i=t-H}^{t} S_{VC}(i). \tag{7}$$

such that α and β denotes the learning rates and ensuring the boundaries, calculated as shown in Equation 8.

$$0 < \delta_{min} \leq \delta_t \leq \delta_{max} < 1. \tag{8}$$

The ARC stabilizes the ADWIN to prevent excessive alarm drift in a noisy period and increases sensitivity when necessary.

3) Performance—Based Controller (PBC)

The PBC analyses the accuracy driven drift management based on the Exponential Moving Average (EMA), calculated as shown in Equation 9 and 10.

$$A_t = 1[\hat{y}_t = y_t], \tag{9}$$

$$EMA_t = \gamma A_t + (1 - \gamma) EMA_{t-1}, \tag{10}$$

Where $0 < \gamma < 1$ denotes the smoothing factor.

4) Dynamic Performance Threshold

The dynamic threshold accuracy is denoted by Equation 11.

$$\theta_t = \mu_{EMA} - \lambda \sigma_{EMA}, \tag{11}$$

such that μ_{EMA} and σ_{EMA} represent the rolling mean and standard deviation of the EMA values, and λ represents factor of sensitivity.

5) PBC Activation Condition

In this manner, the controller triggers performance alerts since calculated as shown in Equation 12.

$$S_{PBC}(t) = \begin{cases} 1, & if\ EMA_t < \theta_t, \\ 0, & otherwise. \end{cases} \tag{12}$$

Compared to VC, PBC react early to the performance decay, especially for gradual drift, subtle distribution shifts, δ instability not caught by ADWIN alone.

2.6. Reset Policy

The policy merges with the VC, ARC, and PBC to achieve corrective actions.

1) Case 1: VC & PBC

If both controllers are achieved as shown in Equation 13.

$$S_{VC}(t) = 1 \; and \; S_{PBC}(t) = 1 , \tag{13}$$

then the 1) ADWIN window is reset, 2) HAT model is considered fully or partial reset, and δ is tightened. That is δ_{t+1} minimizes the post−drift recovery time, calculated as shown in Equation 14 to 19.

$$\delta_{t+1} = max\,(\delta_t - \beta, \delta_{min}). \tag{14}$$

2) Case 2: Performance Collapse Without Statistical Drift

If

$$S_{VC}(t) = 0 \; and \; S_{PBC}(t) = 1 , \tag{15}$$

then the ADWIN increases slightly:

$$\delta_{t+1} = \delta_t - \eta, \tag{16}$$

such that 1) ADWIN window shrinks, 2) the classifier undergoes a partial reset.

3) Case 3: VC Drift but Stable Accuracy

If

$$S_{VC}(t) = 1 \; and \; S_{PBC}(t) = 0 , \tag{17}$$

This represents noise rather than the harmful drift when:

a) δ is loosened

$$\delta_{t+1} = \delta_t + \alpha, \tag{18}$$

b) No classifier reset is triggered and reduces the false alarms caused by the δ noise.

4) Case 4: No Signals

If both VC and PBC are silent:

$$S_{VC}(t) = 0 \; and \; S_{PBC}(t) = 0 \,, \tag{19}$$

The system performs the standard HAT incremental updates.

2.7. Experimental setup

Dataset (CIC—IDS2017) is processed in a chronological order to evaluate the real—time traffic patterns [34, 35]. The processed dataset follows the steps:

1) Time Ordering

In this phase, all the flows are sorted using the Timestamp attribute. No shuffling of the data is applied, preserving the natural day-wise drift (e.g., Monday: benign, Tue-Fri: multiple attacks). The resulting stream of the data reflects a temporal concept drift that includes changes in attack intensity, changes from the normal attack normal intervals, and the variable traffic features per day [36, 37].

2) Class Imbalance

The CIC IDS2017 is imbalanced and dominated by the benign flow during early days and DDoS, brute force, and SYN, which appear in bursts. The classes that are rare, such as infiltration and web attack, appear sparsely, often less than 1%. This imbalanced dataset challenges the accuracy of statistical drift detection and motivates the accuracy-guided controllers [38, 39].

3) Attack Diversity

The streaming preserves all the major attack families [40, 41], as shown in Table 1.

Table 1. Major attack families

Category	Examples	Behaviour in Stream
DoS / DDoS	GoldenEye, Slowloris, Hulk	High-volume bursts, causing abrupt drift
Brute Force	SSH, FTP	Small clusters, subtle drift
Web Attacks	XSS, SQLi, Brute Force	Very sparse, difficult to detect
Port Scan	PortScan	Large waves, strong statistical shift

Category	Examples	Behaviour in Stream
Infiltration	Heartbleed	Extremely low frequency, near-zero prior probability

The irregularity of the dataset distribution, diversity, and temporal patterns make it an ideal for evaluating drift—aware streaming systems [42, 43].

4) Implementation Environment

The framework was implemented utilizing relevant libraries and environments for dashboard interaction data streaming, as shown in Table 2.

Table 2. Implementation Environment

Component	Details
Simulation language	Python 3.11
Frameworks	River (online ML & drift detection), scikit—learn for metrics, Plotly (visualization)
Dataset Loader	Pandas
Execution Platform	Ubuntu 22.04 Intel i7 CPU, 16 GB RAM
Visualization	Streamlit Dashboard with real-time plots

2.8. Evaluation Metrics

To assess performance in a streaming setting, we rely on standard *online prequential* evaluation metrics [44-46]. In prequential evaluation, each incoming instance is first used to test the current model and then (optionally) used to update it, which makes the metrics reflect real-time behavior under non-stationary data.

1) Online Accuracy

Online accuracy measures the cumulative proportion of correct predictions up to time t. It is computed as the running average of an indicator function that equals 1 when the prediction matches the true label and 0 otherwise:

$$Acc_t = \frac{1}{t}\sum_{i=1}^{t} 1(\hat{y}_i = y_i) \quad (20)$$

This metric provides a straightforward view of overall predictive correctness as the stream evolves.

2) F1-Score

Because streaming data can be imbalanced (e.g., one class appearing far more frequently than others), accuracy alone may be misleading. The F1-score is therefore used to better reflect performance under class imbalance by combining precision and recall into a single measure. In practice, the F1-score highlights whether the model is correctly identifying minority-class instances rather than simply benefiting from majority-class dominance.

3) Drift Reaction Time

Drift reaction time quantifies how quickly the system responds once a true concept drift occurs. Specifically, it is defined as the time difference between the *ground-truth* drift point and the first detection raised by the drift detection component (e.g., VC/ADWIN). Smaller values indicate faster detection and, typically, less performance degradation following drift.

4) Adaptation Time

Adaptation time evaluates how quickly the model recovers after drift has been detected and adaptation begins. It is defined as the earliest time $t$at which the online accuracy returns to (or exceeds) a stable reference accuracy level:

$$\tau = \min\{t : Acc_t \geq Acc_{stable}\} \quad (21)$$

This metric captures the resilience of the learning system—i.e., how rapidly it can regain stable predictive performance after the environment changes.

5) False Alarm Rate

False alarms occur when the drift detector signals drift even though no true drift has taken place. The false alarm rate (FAR) measures how often detected alarms are incorrect relative to the total number of alarms:

$$FAR = \frac{Number\ of\ incorrect\ drift\ alarms}{Total\ alarms} \quad (22)$$

A lower FAR indicates more reliable drift detection, reducing unnecessary resets or updates that can harm performance and waste computation.

6) Throughput / Latency

Finally, we measure computational efficiency using throughput and latency. Throughput captures how many instances the system can process per second, which reflects suitability for high-velocity streams. Latency measures the time required to update the model per instance (typically reported in ms/instance). Together, these metrics describe whether the approach can operate in real time while maintaining acceptable predictive performance.

3. RESULTS AND DISCUSSION

This section evaluates the analysis of the proposed framework—VC only, VC+ARC, PBC only, and Trinity controller using the streaming version of CIC—IDS2017. The performance results were assessed based on cumulative accuracy, δ —parameter stability, recovery behavior around the drift points, and case studies for sudden and subtle drift patterns. The experimental simulation was done using prequential evaluation over the full stream.

3.1 Overall Accuracy

The four approaches start with the same accuracy as the result of the cold start condition. The drift is experienced as the stream progresses through multiple regions such as indices 4000, 8000, 12 000, and 15 000; minor deviations are experienced. For example, VC only and VC + ARC experienced large fluctuations after drift points, while PBC-only Trinity configurations maintained smoother accuracy curves. The Trinity controllers maintain stable accuracy over the long period, confirming that integrating EMA performance monitoring mitigates the post-drift instability and maintains extended degradation.

3.2 Around Drift at Index 4000

As shown in figure 5, at around 4000 the feature mean curve increases sharply, reflecting the distributional shift. The Trinity controller stabilizes faster when other methods momentarily dip as the result of sudden change. For example, VC only and VC + ARC experience a delayed realignment of accuracy, while PBC only responds earlier as the

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

result of performance degradation detection. The Trinity Controller uses both the ADWIN trigger and the PBC signal to reset and executes a targeted and early recovery intervention. Thus, the results maintain smoother post-drift accuracy and fewer oscillations in the subsequent window.

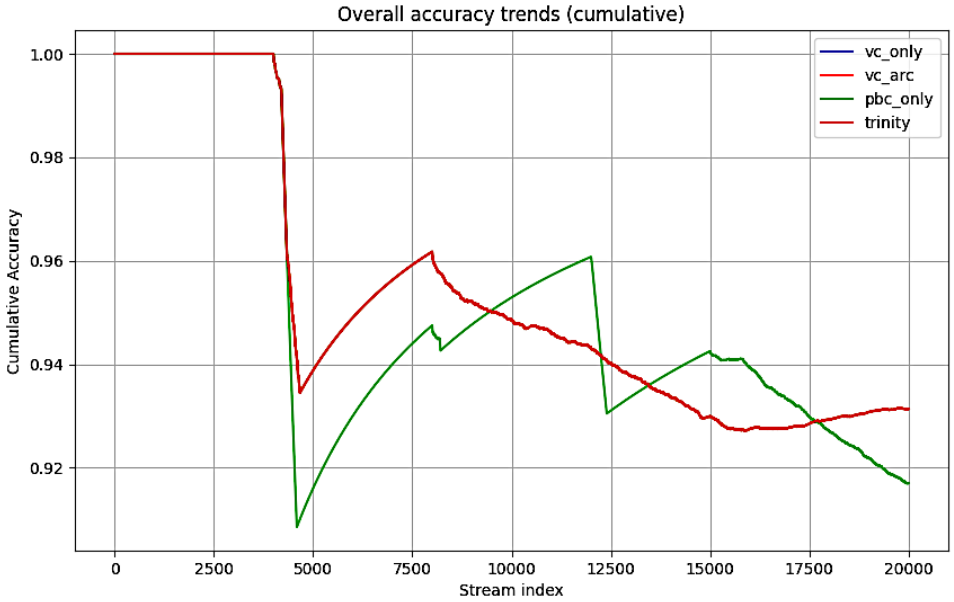

Figure 4. Accuracy Trends

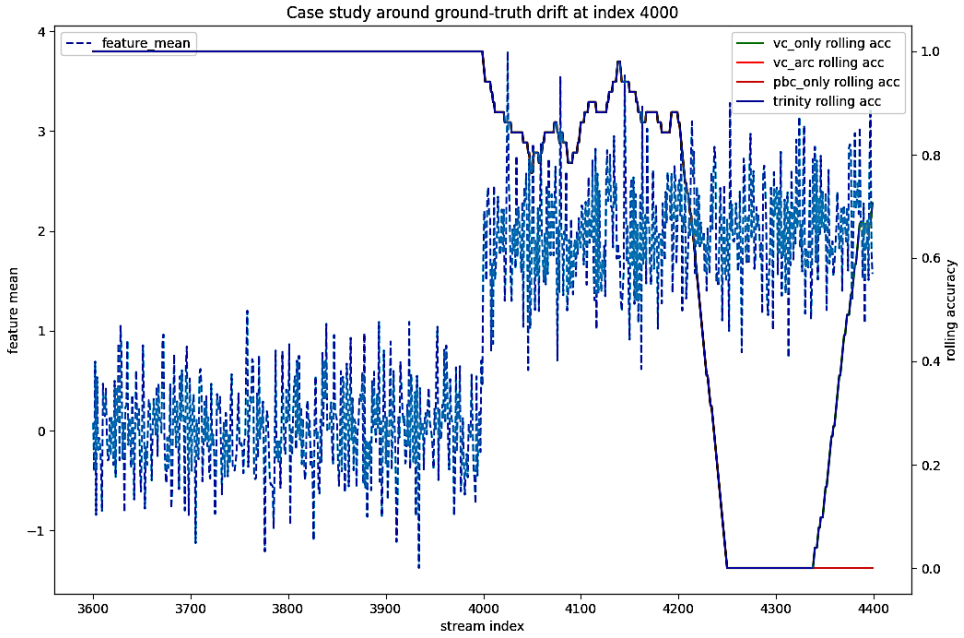

Figure 5. case_study_4000

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

3.3 Around Drift at Index 8000

Figure 6 presents drift behavior around index 8000, where shifts are gradual rather than abrupt. In these scenarios, ADWIN alone struggles to identify the statistical shifts immediately. As shown in figure 3, VC only and VC + ARC manage to track the changes but show prolonged instability. The PBC only manages to reach sooner as the result of the EMA accuracy beginning to fall before the feature means change significantly. The Trinity Controller shows consistent behavior by merging early degradation (PBC) with confirmation of the drift (ADWIN) to alert a reset precisely at the onset of performance collapse. This set the Trinity approach to maintain accuracy closer to the baseline level as compared to other approaches.

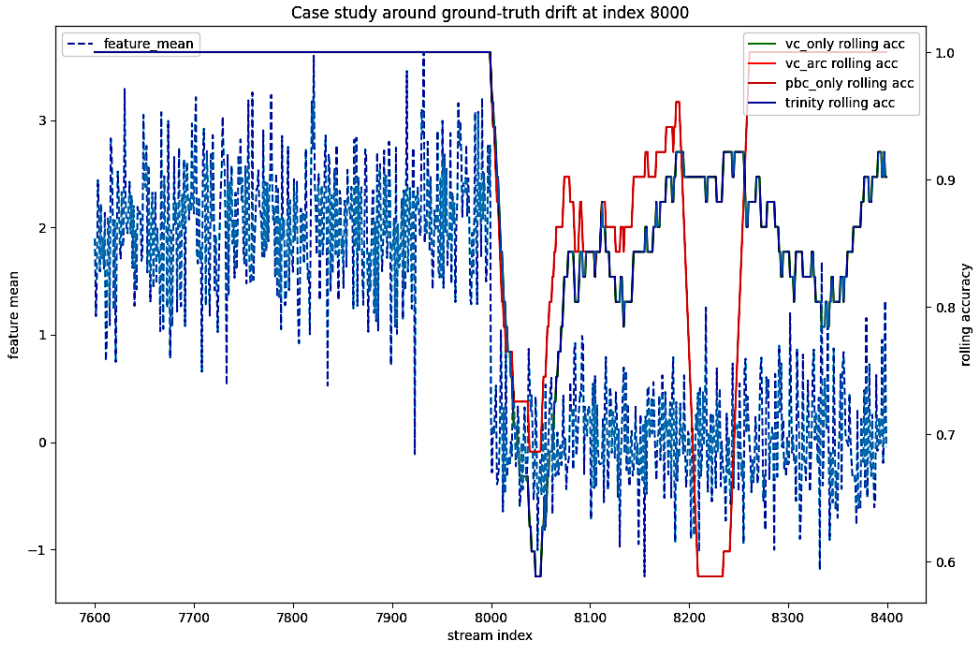

Figure 6. case_study_8000

3.4 Oscillations

To analyze how the ADWIN δparameter evolves during streaming, we examine its behavior under different configurations, with particular focus on adaptive regulation using ARC (as in VC + ARC and Trinity). In these adaptive settings, $\delta$is not fixed; instead, it is updated according to the *recent alert density*, meaning the controller becomes more conservative or more sensitive depending on how frequently drift alerts are being raised. This matters because δdirectly controls ADWIN's sensitivity: smaller values tend to make

detection more conservative (fewer triggers), while larger values generally increase responsiveness in volatile conditions.

In contrast, VC and PBC operate with a fixed δ, which is reflected in the flat, unchanging curves. Since these approaches do not modulate δin response to stream dynamics, they cannot dampen noise-induced fluctuations or selectively increase sensitivity when the data becomes unstable. ARC, however, exhibits clear adaptive modulation: it decreases δduring stable periods to avoid unnecessary alarms, and increases δduring volatile phases to react faster to meaningful changes. This adaptive pattern is visible in Figure 7 (Delta oscillations), where ARC's curve moves up and down in response to the stream conditions rather than remaining constant.

Among the adaptive configurations, the Trinity Controller produces the smoothest δtrajectory, indicating more stable regulation over time. This smoother behavior suggests that combining multiple PBCs helps filter out noise-driven spikes and reduces unnecessary oscillations that would otherwise lead to erratic sensitivity changes. Practically, a smoother δcurve supports more reliable drift handling by limiting false triggers and avoiding repeated, unnecessary resets—an effect that aligns with improved operational stability and reduced disruption to learning. These outcomes are evidenced qualitatively by the reduced oscillatory behavior in Figure 7, and quantitatively by the reduction in false resets and improved stability reported alongside the results in Table 5.

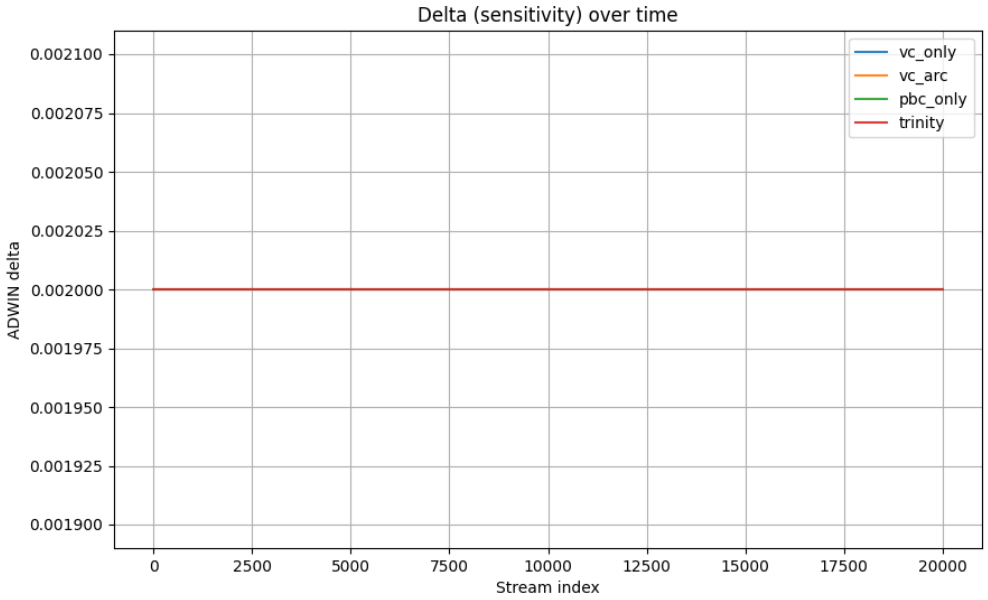

Figure 7. Delta_oscillations

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Figure 7 visualizes the temporal behavior of δacross configurations. The flat lines associated with VC and PBC confirm that these methods keep δconstant, whereas ARC-based methods demonstrate dynamic, context-aware adjustments. The Trinity curve is notably smoother than VC + ARC, reinforcing the claim that Trinity's merged control logic suppresses noise-induced fluctuations and avoids overreacting to transient instability.

Table 5 provides the baseline accuracy levels at key drift indices and the corresponding target thresholds set at 90% of the baseline, which serve as reference points for stability and recovery analysis. At drift indices 4000 and 8000, the baseline accuracy is 1.0, yielding a target threshold of 0.9, indicating that all methods operate at perfect accuracy during these stable regions with no decay. At 12000, the baseline accuracy drops slightly to 0.916 (threshold 0.8244), suggesting a mild shift or imbalance in the data distribution. By 15000, the baseline accuracy becomes 0.886 (threshold 0.7974), reflecting a more challenging segment while still remaining stably above 88%. Taken together, the table complements Figure 7 by grounding the discussion of oscillations in measurable performance conditions: the smoother δmodulation seen under Trinity is consistent with maintaining stable accuracy and reducing unnecessary resets, particularly when the stream becomes more difficult but not necessarily truly drifting.

Table 5. Baseline Accuracies and Thresholds

Drift Index	Baseline_acc	Target_threshold (90%)	Interpretation
4000	1.0	0.9	All methods at perfect accuracy before drift
8000	1.0	0.9	Stable region, no performance decay.
12000	0.916	0.8244	Slight imbalance in data distribution caused a small drop.
15000	0.886	0.7974	More challenging segment; still stable above 88%.

3.5 Discussion

The Trinity-Controller mechanism substantially improves both responsiveness and stability in adaptive streaming IDS operation by treating *observable model degradation*

as a first-class trigger, rather than relying solely on statistical distribution change. Purely statistical drift detectors (including ADWIN-based variants) typically react to distributional shifts (e.g., window mean divergence or confidence-bound violations), which can be weak or delayed under gradual drift, class imbalance, or noisy traffic. In contrast, the EMA-driven Performance-Based Controller (PBC) reacts directly to sustained drops in classifier performance, making it a practical early-warning signal when drift manifests as subtle performance decay rather than a clean statistical discontinuity. This behavior is consistent with the broader results: the long-horizon trend in Figure 4 shows that configurations incorporating PBC—especially Trinity—maintain smoother accuracy trajectories with fewer extended degradation periods compared with VC-only and VC+ARC, which exhibit larger post-drift fluctuations.

A key advantage of the PBC is that it monitors smoothed accuracy (EMA) rather than raw error spikes. This smoothing acts like a low-pass filter: transient misclassifications or short noise bursts are less likely to dominate the control decision, while persistent downward trends become clearly visible. This is particularly important in IDS streams such as CICIDS2017, where traffic volatility and class imbalance can cause detectors to overreact. The complementary role of VC/ARC becomes evident here: VC provides statistical validation via ADWIN, ARC regulates sensitivity to prevent unstable triggering, and PBC supplies performance evidence that the drift is *actually harming detection*. In the case studies around the drift indices (e.g., the abrupt region near 4000 and the more gradual region around 8000), Trinity's fused decision logic helps explain the smoother recovery behavior observed in Figure 5 and Figure 6: performance degradation detected by PBC can prompt earlier corrective action, while VC confirmation prevents unnecessary full resets when accuracy remains stable.

The stability benefit is especially clear in the behavior of the adaptive δparameter. ADWIN sensitivity can oscillate in unstable segments, producing repeated alarms and disruptive resets that slow recovery and reduce reliability. By coordinating ARC's δmodulation with PBC's degradation signal, Trinity reduces "thrashing" (rapid tighten/loosen cycles) and unnecessary resets. This is supported by the smoother δtrajectory shown in Figure 7, where Trinity's curve is less jagged than VC+ARC, indicating that merging control signals reduces noise-induced oscillations. The operational meaning of this stability is reinforced by Table 5, which anchors performance expectations using baseline accuracies and 90%

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

thresholds at key indices (e.g., 1.0 → 0.9 at 4000/8000; 0.916 → 0.8244 at 12000; 0.886 → 0.7974 at 15000). In other words, Trinity's smoother control does not merely "look stable"—it helps the system stay nearer to baseline performance and avoid resets in segments that are harder but not necessarily truly drifting.

Despite these benefits, the discussion must acknowledge important limitations. First, EMA is inherently a lagging indicator: it needs several samples to reflect abrupt performance drops, which can be problematic during short-lived burst attacks where harm emerges and disappears quickly [47]. In those cases, the classifier may degrade before the PBC registers a sufficiently strong decline, creating a short vulnerability window. Second, the threshold design is critical. If the performance threshold is too sensitive, minor fluctuations can trigger excessive resets, wasting compute and destabilizing learning; if it is too conservative, corrective action is delayed and accuracy can remain suppressed longer than necessary [48]. Third, the PBC assumes access to real-time labelled feedback, which is often available in experimental benchmarks but may be limited, delayed, or noisy in operational networks—especially for novel attacks or incomplete incident triage [49]. These constraints do not invalidate the approach, but they frame where Trinity performs best (high-feedback streaming environments) and where extensions are needed (weak-label or delayed-label settings).

The practical implications for real-world network security are strong. In routers, gateways, and other high-throughput environments, the ability to adapt quickly without prolonged degradation is essential to maintaining throughput and minimizing false negatives as attack patterns evolve [50]. Trinity's fused control loop is well-aligned with this need because it prioritizes interventions that are justified by *both* statistical evidence and measurable performance decay, reducing disruptive resets that would otherwise harm continuity. Similarly, Security Operations Centres (SOCs) can benefit from fewer inconsistent alarms and fewer unnecessary adaptations, which directly reduces operator workload and alert fatigue in continuous monitoring pipelines [51]. More broadly, the framework's resilience to drift supports forensic and enterprise monitoring contexts where benign behavior changes over time (software updates, policy shifts, workload cycles) and where naive drift handling can overreact to "expected" variation.

Recent research has also explored uncertainty-driven drift detection, where prediction confidence, entropy, or calibration shifts serve as early indicators of concept drift. These approaches are attractive when labels are scarce, but they often require careful probabilistic calibration and can be sensitive to model-specific uncertainty artifacts, triggering frequent alerts under noisy confidence fluctuations. Trinity's key distinction is its use of direct performance evidence (EMA accuracy) as a model-agnostic, interpretable signal, while still retaining statistical confirmation (VC) and adaptive sensitivity control (ARC). Looking forward, several refinements remain open. First, integrating confidence- or uncertainty-based signals could complement accuracy monitoring and detect degradation earlier—especially in low-label regimes [52, 53]. Second, adopting a hybrid partial-reset strategy—refreshing only affected rules, subtrees, or ensembles rather than restarting the entire model—could reduce disruption and accelerate recovery [54, 55]. Third, adding a drift severity estimator would allow the controller to scale reset intensity to the magnitude of change, improving robustness across mild and extreme drift episodes. Finally, validating across additional datasets (e.g., CSE-CIC-2018 and UNSW-NB15) and deploying in real-time high-throughput environments (including SOC pipelines and programmable network devices) are natural next steps to confirm operational feasibility and sustained performance under production constraints [56, 57].

4. CONCLUSION

This paper introduced Trinity Controller ADWIN, a unified adaptive framework for streaming intrusion detection that tightly integrates statistical drift detection (VC/ADWIN), adaptive sensitivity regulation (ARC), and performance-driven feedback (PBC). By treating accuracy degradation as an operationally meaningful drift signal—rather than relying exclusively on distributional change—the framework addresses two persistent limitations of conventional ADWIN-based IDS: delayed reaction under gradual or subtle drift and instability caused by sensitivity oscillations. Experimental evaluation on a time-ordered CICIDS2017 stream showed that Trinity delivers stronger long-horizon accuracy stability, faster post-drift recovery, and fewer unnecessary resets than fixed-sensitivity and purely statistical configurations. In several drift regions, the combined control logic enabled *near-zero recovery delay*, indicating that the model can regain stable performance with minimal disruption after drift onset.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

Beyond aggregate performance gains, the results highlight an important systems-level benefit: more reliable control decisions under noise and imbalance. The PBC's EMA-based monitoring provides early evidence of sustained performance decay, while VC/ADWIN offers statistical confirmation and ARC stabilizes sensitivity, collectively reducing false triggers and preventing repeated, disruptive resets. This coordinated behavior is consistent with the smoother trajectories observed across the reported figures and the stability references summarized in the baseline threshold analysis. Overall, the framework shifts drift management toward a more *performance-aware* and *operationally grounded* approach, aligning adaptation decisions with what ultimately matters in IDS deployments—maintaining detection quality over time in the presence of evolving traffic. From a practical perspective, these findings suggest that performance-aware adaptation can materially improve the dependability of IDS operating in dynamic real-world environments such as enterprise networks, edge gateways, and Security Operations Centres. By improving responsiveness without sacrificing stability, Trinity supports long-term, autonomous streaming IDS operation under evolving attack strategies, shifting benign behavior, and realistic noise conditions. This makes the proposed framework a strong candidate for deployment in continuous monitoring pipelines where minimizing performance collapse, avoiding unnecessary model resets, and sustaining throughput are critical operational requirements.

REFERENCES

[1] N. Malathy *et al.*, "Real-time intrusion detection in IIoT stream data using window-based weighted ensemble techniques," *SN Comput. Sci.*, vol. 6, no. 1, p. 66, 2025.

[2] M. A. Shyaa *et al.*, "Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems," *Eng. Appl. Artif. Intell.*, vol. 137, p. 109143, 2024.

[3] M. Antonijevic *et al.*, "Intrusion detection in metaverse environment Internet of Things systems by metaheuristics-tuned two-level framework," *Sci. Rep.*, vol. 15, no. 1, p. 3555, 2025.

[4] F. Jemili, K. Jouini, and O. Korbaa, "Intrusion detection based on concept drift detection and online incremental learning," *Int. J. Pervasive Comput. Commun.*, vol. 21, no. 1, pp. 81–115, 2025.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[5] K. M. K. Kumar, M. V. S. Reddy, and K. Ullas, "Distributed intrusion detection system using Kafka and Spark streaming," in *Proc. 2025 Int. Conf. Visual Analytics and Data Visualization (ICVADV)*. IEEE, 2025.

[6] R. Sebopelo, "Adaptive-delta ADWIN for balancing sensitivity and stability in streaming IDS," *J. Inf. Syst. Informat.*, vol. 7, pp. 2876–2897, Sep. 2025.

[7] N. Gunasekara *et al.*, "Recurrent concept drifts on data streams," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2024.

[8] C. M. Garcia *et al.*, "Concept drift adaptation in text stream mining settings: A systematic review," *ACM Trans. Intell. Syst. Technol.*, vol. 16, no. 2, pp. 1–67, 2025.

[9] L. Hu, Y. Lu, and Y. Feng, "Concept drift detection based on deep neural networks and autoencoders," *Appl. Sci.*, vol. 15, no. 6, p. 3056, 2025.

[10] D. Shang, G. Zhang, and J. Lu, "Novelty-aware concept drift detection for neural networks," *Neurocomputing*, vol. 617, p. 128933, 2025.

[11] R. Marpu and B. Manjula, "Streaming machine learning algorithms with streaming big data systems," *Braz. J. Develop.*, vol. 10, no. 1, pp. 322–339, 2024.

[12] P. Lu *et al.*, "Early concept drift detection via prediction uncertainty," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 18, 2025.

[13] B. Gower-Winter *et al.*, "Identifying predictions that influence the future: Detecting performative concept drift in data streams," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 11, 2025.

[14] U. Ali and T. Mahmood, "A novel framework for concept drift detection using autoencoders for classification problems in data streams," *Int. J. Mach. Learn. Cybern.*, vol. 16, no. 1, pp. 397–418, 2025.

[15] A. Kraus and H. van der Aa, "Machine learning-based detection of concept drift in business processes," *Process Sci.*, vol. 2, no. 1, pp. 1–26, 2025.

[16] P. Porwik *et al.*, "A novel method for drift detection in streaming data based on measurement of changes in feature ranks," *J. Artif. Intell. Soft Comput. Res.*, vol. 15, 2025.

[17] X. Li and Q. Gu, "Understanding SGD with exponential moving average: A case study in linear regression," arXiv preprint arXiv:2502.14123, 2025.

[18] O. Madani, "Sparse moving averages for lifelong open-ended probabilistic prediction," in *Proc. 2025 ACM Int. Workshop Security and Privacy Analytics*, 2025.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[19] G. Bandarupalli, "Efficient deep neural network for intrusion detection using CIC-IDS-2017 dataset," in *Proc. 2025 1st Int. Conf. Adv. Comput. Sci., Electr., Electron. Commun. Technol. (CE2CT)*. IEEE, 2025.

[20] R. Dube, "Faulty use of the CIC-IDS2017 dataset in information security research," *J. Comput. Virol. Hack. Tech.*, vol. 20, no. 1, pp. 203–211, 2024.

[21] D. K. Putra *et al.*, "Comparative analysis of machine learning algorithms in detecting DDoS attacks on CICIDS2017 dataset," *J. Intell. Syst. Inf. Technol.*, vol. 2, no. 2, 2025.

[22] A. Esteban, A. Zafra, and S. Ventura, "MIHT: A Hoeffding tree for time series classification using multiple instance learning," in *Proc. Int. Conf. Intell. Data Eng. Autom. Learn. (IDEAL)*. Cham, Switzerland: Springer, 2025.

[23] H. I. Bensaoula and S. N. Bahloul, "Enhanced green accelerated Hoeffding trees for improved data stream classification," *Comput. Sci. J. Moldova*, vol. 98, no. 2, pp. 159–187, 2025.

[24] A. Gupta and S. Babu, "Real-time transaction fraud detection using adaptive Hoeffding trees for concept-drift resilience," *Int. J. Comput. Model. Appl.*, vol. 2, no. 3, pp. 1–8, 2025.

[25] K. Köbschall, L. Hartung, and S. Kramer, "Adaptive differentiable trees for transparent learning on data streams," *Mach. Learn.*, vol. 114, no. 11, p. 253, 2025.

[26] J. V. Guerrero Cano, G. J. Aguiar, and A. Cano, "Anticipating to change: A proactive approach for concept drift adaptation in data streams," *Mach. Learn.*, vol. 115, no. 1, p. 3, 2026.

[27] D. Joshi and M. Shukla, "A pre-emptive resilient ML approach for drift detection in real-time stream data," in *Proc. 2025 Artif. Intell. Smart Technol. Sustainability Conf. (AISTS)*. IEEE, 2025.

[28] F. Pizarro *et al.*, "Low-overhead learning: Quantized shallow neural networks at the service of genetic algorithm optimization," *Biomimetics*, vol. 10, no. 11, p. 762, 2025.

[29] M. Liu *et al.*, "LiedNet: A lightweight network for low-light enhancement and deblurring," *IEEE Trans. Circuits Syst. Video Technol.*, 2025.

[30] V. Jain and A. Mitra, "Real-time threat detection in cybersecurity: Leveraging machine learning algorithms for enhanced anomaly detection," in *Machine Intelligence Applications in Cyber-Risk Management*. Hershey, PA, USA: IGI Global, 2025, pp. 315–344.

[31] S. Antony Joseph Raj and M. Madiajagan, "HAMC-ID: Hybrid attention-based meta-classifier for intrusion detection," *Sci. Rep.*, 2025.

[32] J. Igual *et al.*, "Linear adaptive filtering for regression in data streams," *Int. J. Data Sci. Anal.*, pp. 1–16, 2025.

[33] A. M. Paim *et al.*, "Efficient instance selection in tree-based models for data streams classification," in *Proc. 40th ACM/SIGAPP Symp. Appl. Comput. (SAC)*, 2025.

[34] R. Morshedi and S. M. Matinkhah, "Intrusion detection in IoT using deep recurrent neural networks: A complex network approach to modeling emergent cyberattack behaviors," *Complexity*, vol. 2025, p. 9693472, 2025.

[35] Z. I. Khan, M. M. Afzal, and K. N. Shamsi, "A comprehensive study on CIC-IDS2017 dataset for intrusion detection systems," *Int. Res. J. Adv. Eng. Hub*, vol. 2, no. 2, pp. 254–260, 2024.

[36] M. Arcos-Argudo, R. Bojorque, and A. Torres, "A deterministic comparison of classical machine learning and hybrid deep representation models for intrusion detection on NSL-KDD and CICIDS2017," *Algorithms*, vol. 18, no. 12, p. 749, 2025.

[37] J. Li *et al.*, "Concept drift adaptation by exploiting drift type," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 4, pp. 1–22, 2024.

[38] S. Senthilkumar and S. K. Balasubramanian, "Advancing multi-class intrusion detection: A comparative evaluation of LSTM and Bi-LSTM on class-imbalanced CIC-IDS-2017," *Turk. J. Eng.*, vol. 9, no. 3, pp. 578–590, 2025.

[39] M. E. Sobhani, A. T. Rodela, and D. M. Farid, "Adaptive TreeHive: Ensemble of trees for enhancing imbalanced intrusion classification," *PLOS ONE*, vol. 20, no. 9, e0331307, 2025.

[40] Z. A. Sheikh *et al.*, "Generalizability assessment of learning-based intrusion detection systems for IoT security: Perspectives of data diversity," *Secur. Privacy*, vol. 8, no. 2, e70014, 2025.

[41] S. Alzu, F. Stahl, and M. Al-Khafajiy, "Detect, decide, explain: An intelligent framework for zero-day network attack detection," in *Proc. Int. Conf. Innovative Techniques and Applications of Artificial Intelligence*. Cham, Switzerland: Springer, 2025.

[42] T. Luo and R. Li, "A dynamic hidden state correction and feedback-driven online adaptation framework for network intrusion detection," in *Proc. 2025 Asia–Europe Conf. Cybersecurity, Internet of Things and Soft Computing (CITSC)*. IEEE, 2025.

[43] G. R. Devi *et al.*, "A machine learning approach to real-time network attack identification," in *Proc. 2025 9th Int. Conf. Inventive Systems and Control (ICISC)*. IEEE, 2025.

[44] D. Lukats *et al.*, "A benchmark and survey of fully unsupervised concept drift detectors on real-world data streams," *Int. J. Data Sci. Anal.*, vol. 19, no. 1, pp. 1–31, 2025.

[45] S. Arora, R. Rani, and N. Saxena, "A systematic review on detection and adaptation of concept drift in streaming data using machine learning techniques," *WIREs Data Min. Knowl. Discov.*, vol. 14, no. 4, e1536, 2024.

[46] F. Sharief *et al.*, "Multi-class imbalanced data handling with concept drift in fog computing: A taxonomy, review, and future directions," *ACM Comput. Surv.*, vol. 57, no. 1, pp. 1–48, 2024.

[47] N. Harshit and K. Mounvik, "Improving real-time concept drift detection using a hybrid transformer-autoencoder framework," arXiv preprint arXiv:2508.07085, 2025.

[48] V. Agate *et al.*, "Enhancing IoT network security with concept drift-aware unsupervised threat detection," in *Proc. IEEE Symp. Computers and Communications (ISCC)*, 2024.

[49] S. Seth, K. K. Chahal, and G. Singh, "Concept drift-based intrusion detection for evolving data stream classification in IDS: Approaches and comparative study," *Comput. J.*, vol. 67, no. 7, pp. 2529–2547, 2024.

[50] S. Yang *et al.*, "Recda: Concept drift adaptation with representation enhancement for network intrusion detection," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, 2024.

[51] A. S. Chamkar, Y. Maleh, and N. Gherabi, "Security operation center," in *The Art of Cyber Defense: From Risk Assessment to Threat Intelligence*, 2024, p. 271.

[52] L. Zhao and Y. Shen, "Proactive model adaptation against concept drift for online time series forecasting," in *Proc. 31st ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, vol. 1, 2025.

[53] K. Wan, Y. Liang, and S. Yoon, "Online drift detection with maximum concept discrepancy," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, 2024.

[54] K. Wang *et al.*, "TS-DM: A time segmentation-based data stream learning method for concept drift adaptation," *IEEE Trans. Cybern.*, 2024.

[55] V. Sharma and M. Kumar, "Improving intrusion detection with hybrid deep learning models: A study on CIC-IDS2017, UNSW-NB15, and KDD CUP 99," *J. Inf. Syst. Eng. Manag.*, vol. 10, 2025.

ISI Journal of Information Systems and Informatics
Published By Asosiasi Doktor Sistem Informasi Indonesia

[56] F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—A survey on monitoring in evolving environments. Part A: Detecting concept drift," *Front. Artif. Intell.*, vol. 7, p. 1330257, 2024.

[57] M. Cantone, C. Marrocco, and A. Bria, "On the cross-dataset generalization of machine learning for network intrusion detection," arXiv preprint arXiv:2402.10974, 2024.