Vol. 7, No. 4, December 2025

ISI Journal of Information Systems and Informatics

Published By Asosiasi Doktor Sistem Informasi Indonesia

Adaptive–Delta ADWIN: A Framework for Stable and Sensitive Intrusion Detection in Streaming Networks

Rodney Buang Sebopelo

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

Email: rsebopelob@gmail.com

Received: Oct 12, 2025
Revised: Nov 10, 2025
Accepted: Nov 27, 2025
Published: Dec 12, 2025

Corresponding Author
Author Name*:
Rodney Buang Sebopelo
Email*:
rsebopelob@gmail.com

DOI:
10.63158/journalisi.v7i4.1336

Abstract. Intrusion Detection Systems (IDS) must effectively adapt to dynamic network traffic, where concept drift continuously shifts the patterns of both benign and malicious behaviors. Traditional drift detection methods, which rely on a fixed sensitivity parameter (δ), often struggle to balance the need for rapid adaptation with the stability required to minimize false alarms. This study proposes the Adaptive-Delta ADWIN framework, a novel approach that dynamically adjusts the sensitivity parameter δ in response to evolving traffic patterns. The framework utilizes two lightweight online controllers: the Volatility Controller (VC) and the Alert-Rate Controller (ARC), to optimize δ in real time. Evaluated on the CICIDS2017 dataset, the Adaptive-Delta ADWIN framework integrates a multiclass ensemble of Hoeffding Adaptive Trees for robust intrusion detection. Experimental results show that the framework achieves an impressive accuracy range of 93-95%, reduces false positives by 50%, lowers false negatives by 30%, and improves ROC-AUC by up to 6.6% when compared to fixed-δ baseline methods. These findings demonstrate significant improvements in both detection performance and adaptability to concept drift. However, challenges remain in extending this approach to larger datasets and ensuring its efficiency in high-throughput, real-time deployment, which will be addressed in future work.

Keywords: Concept drift, Intrusion detection system, Streaming data, Controllers

1. INTRODUCTION

Intrusion Detection Systems (IDS) are crucial for protecting networks against malicious activities, particularly in environments with complex and dynamic network traffic. In such settings, IDS must operate in an ever-evolving landscape where legitimate usage patterns and attack strategies change over time, resulting in "concept drift" — a phenomenon where the statistical distribution of data evolves, leading to potential performance degradation if not properly addressed [1].

As modern traffic environments continue to grow in complexity, IDS need to respond to concept drift in real time. Systems that fail to adapt quickly become vulnerable to emerging threats, while overly sensitive models may generate an excessive number of false alarms [5]. For instance, in the CICIDS2017 dataset, network traffic patterns vary across days: benign traffic dominates on Monday, while Wednesday sees a spike in DoS attacks, and Thursday introduces web-based exploits and infiltration attempts [6]. A fixed sensitivity parameter in IDS may struggle to detect new attack patterns when benign traffic fluctuates over time [7, 8].

Traditional concept drift detection methods, such as ADWIN and DDM [9, 10], have been widely used to address this challenge. However, these methods rely on a fixed sensitivity parameter (delta) [11, 12]. Previous research on IDS has typically focused on binary detection tasks (benign vs. attack) [13, 14], revealing three key limitations: (1) Difficulty in balancing responsiveness and stability when traffic patterns change rapidly. (2) Inability to manage excessive drift alarms in long-running streams. (3) Lack of adaptation for multiclass intrusion detection, where multiple attack types evolve at different rates.

Despite significant advancements in concept drift detection, no existing framework has successfully integrated adaptive sensitivity control with multiclass IDS to strike an optimal balance between stability and responsiveness in real-time streaming environments. Moreover, there is a lack of research combining volatility-driven δ adaptation with long-term alarm-rate regulation in an ADWIN-based multiclass IDS [21].

To bridge this gap, we propose the Adaptive-Delta ADWIN framework, which dynamically adjusts the ADWIN sensitivity parameter (δ) using two lightweight online controllers: the

Volatility Controller (VC) and the Alert-Rate Controller (ARC). This adaptive framework is coupled with a streaming ensemble of Hoeffding Adaptive Trees for multiclass intrusion detection [23]. The proposed approach aims to improve stability, reduce false alarms, and enhance the system's responsiveness to real network changes.

Previous work on concept drift detection has contributed significantly to the field. For example, Sandeep Bharadwa et al. [12] categorized types of concept drift in data streams and evaluated various detection and mitigation strategies. Tajwar Mehmood et al. [15] focused on early drift detection, with their method outperforming others in handling sudden drift. Supriya Agrahari et al. [16] introduced DD-SCC-I and DD-KRC-I to handle multidimensional data and demonstrated early drift detection effectiveness. Similarly, Hassan Mehmood et al. [17] proposed a drift detection method for time-series data in distributed networks, achieving superior performance. Other works, such as Pingfan Wang et al.'s Noise-Tolerant Drift Detection Method (NTDDM) [18] and Mansour Zoubeirou et al.'s autoregressive-based detection [19], also offered improvements in drift detection accuracy. Additionally, Yan Zhao et al. [20] developed the STS-AEL method, which incorporates stratified and time-aware sampling to enhance detection accuracy in streaming networks.

The remainder of this paper is structured as follows: Section 2 introduces the proposed Adaptive-Delta ADWIN framework. Section 3 outlines the experimental setup and parameter settings. Section 4 presents the experimental results. Section 5 discusses the findings and limitations. Finally, Section 6 concludes the paper and suggests directions for future research.

2. METHODOLOGY

This section presents Adaptive−δ ADWIN for real−time intrusion detection. The proposed approach enhances the traditional ADWIN drift detector by dynamically adjusting the sensitivity parameter (δ) through two online controllers: the VC and ARC. The adaptive detector is integrated into a streaming ensemble of Hoeffding Adaptive Trees, enabling multi−class intrusion detection with an improved balance between stability and responsiveness. This is the first framework to combine volatility−based δ adaptation (VC) with long–term alarm–rate stabilization (ARC), enabling ADWIN to maintain both sensitivity and stability in multiclass streaming IDS.

2.1. Adaptive–Delta ADWIN

ADWIN traditionally decides whether data distribution changes by comparing two statistically different windows. However, its behaviour depends entirely on the sensitivity parameter δ. A fixed δ makes the detector either too sensitive—causing alarm flooding—or too stable, delaying detection of real attacks. The proposed Adaptive-δ ADWIN framework addresses this limitation by automatically adjusting δ using two online controllers: the Volatility Controller (VC), which reacts to sudden changes in prediction error, and the Alert-Rate Controller (ARC), which prevents excessive drift alarms.

ADWIN is a drift detection algorithm designed for streaming data [11, 24]. Its primary objective is to maintain a variable–length sliding window that continuously tests for statistically significant changes between the two sub–windows. When a shift is detected, the older portion of the window is discarded, and a drift alarm is triggered [25]. Role of δ – The sensitivity of ADWIN is governed by the parameter δ, which represents the probability of false alarm. A smaller δ increases sensitivity, allowing faster adaptation to drift, whereas a larger δ enhances stability by reducing false alarms but may slow down the response to actual changes. Limitations of a Fixed δ – In dynamic network environment, long benign periods are often interspersed with sudden attack. A fixed δ cannot effectively handle this variability. When δ is set small, it becomes overly sensitive and triggering alarms. When large, it reacts slowly with delayed attack detection. To address these limitations, we propose Adaptive–δ ADWIN that dynamically adjusts the sensitivity parameter (δ) and alarm rate to achieve a trade–off between sensitivity and stability.

2.2. Volatility Controller (VC)

The VC adjusts the sensitivity parameter (δ) in response to changes in the prediction signal.

1) *Error Smoothing* – let e_t denote the prediction error at time t. A smoothed error signal $\hat{e}_t$ is computed using the Exponential Moving Average (EMA) [26], as shown in Equation 1.

$$\hat{e}_t = \beta \cdot e_t + (1 - \beta) \cdot \hat{e}_{t-1} \tag{1}$$

where β is the smoothing factor controlling the contribution of recent errors.

2) *Volatility Estimation* – The error volatility $\boldsymbol{v_t}$ is the derived as shown in Equation 2.

$$v_t = \gamma \cdot |\hat{e}_t - \hat{e}_{t-1}| + (1-\gamma) \cdot v_{t-1} \tag{2}$$

where $\boldsymbol{\gamma}$ controls the smoothing level of the volatility signal.

3) δ *Update Rule* – If v_t exceeds a target volatility v_{target}, δ is reduced to increase sensitivity. Conversely, when volatility is low, δ is increased to maintain stability. The update rule is as shown in Equation 3.

$$\delta_t = \delta_{t-1} \cdot \exp(k \cdot v_{target} - v_t) \tag{3}$$

where k represents the controller gain factor. In this way, the VC ensures that the detector reacts quickly during volatile periods while avoiding unnecessary sensitivity in stable traffic conditions.

2.3. Alert–Rate Controller (ARC)

The ARC regulates δ by monitoring the frequency of drift alarms to prevent alarm flooding [27].

1) Drift Rate Estimation – Let r_t denote the average number of drift alarms observed within a sliding window of size W.
2) Target Regulation – A target alarm rate, ρ_{target} is defined to guide the adjustment δ. If $r_t > \rho_{target}$, δ is a multiplicatively increased to reduce the sensitivity. If $r_t < \rho_{target}$, δ is decreased to improve responsiveness, as shown in Equation 4.

$$\delta_t = \begin{cases} \delta_{t-1} \cdot U, & r_t > \rho_{target} \\ \delta_{t-1} \cdot D, & r_t < \rho_{target} \end{cases} \tag{4}$$

where $U > 1$ is the upscaling factor and $D < 1$ is the downscaling factor. Through this mechanism, ARC maintains the long–term stability of the detector and suppresses excessive alarms without compromising detection capability.

2.4. Integration with Ensemble IDS

In this study, the proposed Adaptive–Delta ADWIN detector is integrated into a streaming ensemble IDS based on Hoeffding Adaptive Trees as follow.

1) Base learners—The ensemble consists of multiple Hoeffding Adaptive Trees that are trained incrementally on incoming network traffic statistics [28, 29].
2) Learning loop—For each traffic instance in the stream, the following steps are executed:
   a) Prediction: The ensemble produces a predicted label class label for the incoming instances.
   b) Update: The prediction error is computed and passed to the Adaptive—Delta ADWIN drift detector.
   c) Drift check: When a drift is detected, the affected base learner(s) are retrained or reset using the most recent data.

2.4. Framework Architecture and Operation

This section presents the Adaptive—δ ADWIN framework to enhance real—time intrusion detection in dynamic network traffic environments [30, 31].

2.4.1. Framework Architecture

Figure 1. Adaptive-δ ADWIN framework architecture. The ensemble of Hoeffding Adaptive Trees generates predictions for each incoming instance, and the prediction error is passed to two online controllers [32, 33].. The Volatility Controller (VC) adjusts δ based on fluctuations in smoothed error, ensuring rapid response to sudden behavioral shifts. The Alert-Rate Controller (ARC) regulates δ according to the frequency of drift alarms to prevent alarm flooding. The updated δ is fed back into ADWIN to detect distributional changes and detected drifts trigger reset/retraining of affected base learners.

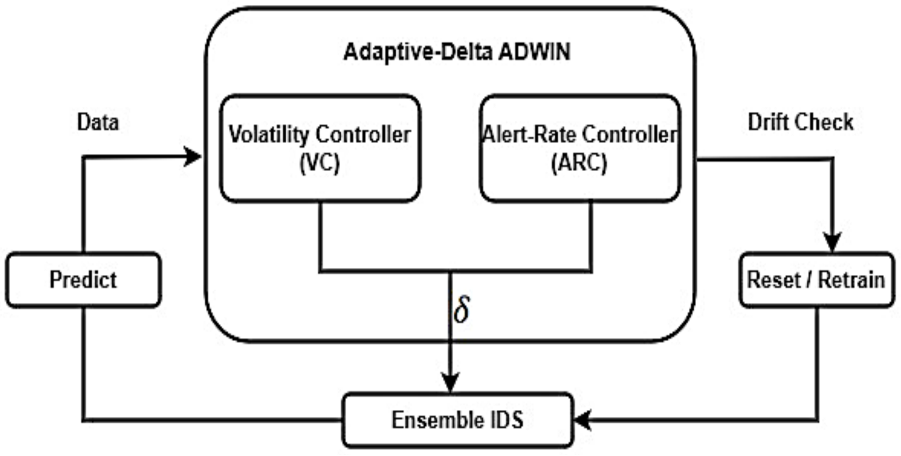


Figure 1. Adaptive-δ ADWIN framework

2.5.1. Framework Process Operation

Figure 2. End-to-end workflow of the Adaptive-δ ADWIN process. Incoming streaming data first enters the Hoeffding Adaptive Tree ensemble, which produces predictions and computes the prediction error. This error is smoothed and then processed by the VC to react to short-term volatility, while the ARC regulates long-term alarm behavior. The controllers update δ, which ADWIN uses for drift detection. When a drift is detected, the corresponding base learners are reset or retrained using the most recent data, completing the adaptive feedback loop.

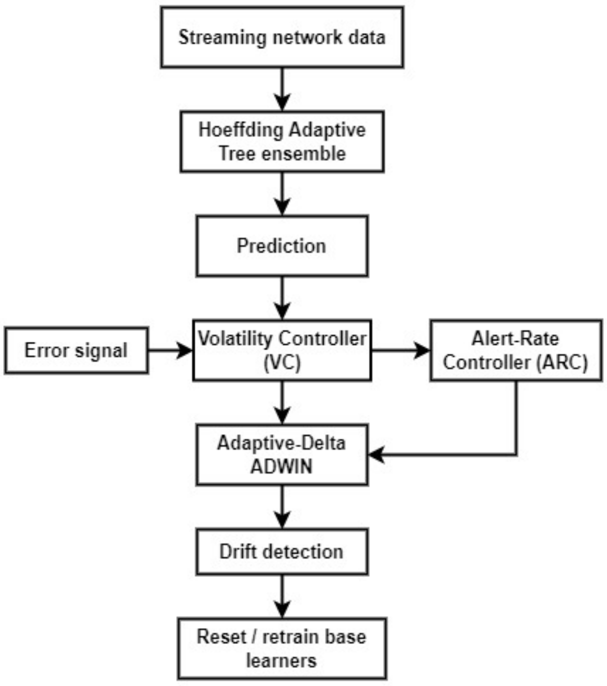

Streaming network data
Hoeffding Adaptive Tree ensemble
Prediction
Error signal
Volatility Controller (VC)
Alert-Rate Controller (ARC)
Adaptive-Delta ADWIN
Drift detection
Reset / retrain base learners

Figure 2. Adaptive−δ ADWIN Workflow

```
Algorithm 1: Adaptive-Delta ADWIN for Multi−Class Intrusion Detection
Input:  D $(X, y)$, initial $\delta_\theta$
          $v_{target}$
          $\rho_{target}$
Output: Adaptive multi−class IDS predictions with controlled stability and Sensitivity
   1.  Initialize Hoeffding Adaptive Tree Ensemble E
   2.  Initialize ADWIN $\delta = \delta_\theta$
   3.  Initialize VC and ARC
   4.  Each sample $(x_t, y_t)$ do
   5.  $y_{pred} \leftarrow E \cdot predict\ (x_t)$
                          $error_t \leftarrow compute_{error}(y_{pred}, y_t)$
```

Algorithm 1: Adaptive-Delta ADWIN for Multi–Class Intrusion Detection

```
6.  Update smoothed error via EMA
7.  VC update– adjust $\delta$
8.  ARC update– adjust $\delta$
9.  Feed $error_t$ into ADWIN($\delta$)
10. If ADWIN signals drift, then
11. Reset or retrain base learner (s)
12. End if
13. Update Ensemble E with $(x_t, y_t)$
```

2.5.2. Design Rationale

The Adaptive-δ ADWIN framework is geared to tackle both sensitivity and stability in streaming drift detectors. Since the traditional ADWIN uses fixed δ parameter which delays detection of sudden drifts. The framework introduces two online controllers: VC which adjusts δ in response to prediction error, and ARC which manages frequency of alarms. Both controllers provide optimal balance between the responsiveness and stability in a closed–loop δ parameter. The framework ensures a robust and responsive solution by achieving both adaptability and resilience in a multi–class intrusion detection within the dynamic network environments.

3. RESULTS AND DISCUSSION

3.1. Experimental Setup

This section presents the experimental settings used to evaluate the framework performance for multi–class intrusion detection [35, 36].

3.1.1. Dataset

1) CICIDS2017: A widely recognized benchmark dataset for multiclass intrusion detection, CICIDS2017 contains both normal traffic and a variety of attack categories [37]. It is well-suited for evaluating intrusion detection systems due to its diversity and real-world relevance.
2) Chronological Streaming Order: To simulate real-world network traffic patterns and the natural progression of concept drift, the CICIDS2017 dataset was processed in temporal order [38, 39]. This ensures that the data reflects the evolution of both

benign and malicious activities over time, mimicking the dynamic nature of actual network traffic.

3) Preprocessing

The preprocessing steps applied to the CICIDS2017 dataset are summarized in Table 1, which outlines how the data was prepared for continuous stream processing in the IDS system.

Table 1. Preprocessing Steps For CICIDS2017 Stream

Step	Description	Purpose in IDS
File Merging	The CICIDS2017 dataset was merged into a single CSV file for continuous stream of network traffic.	Ensures the chronological flow of traffic, simulating realistic streaming evolution.
Label Encoding	Categorical attack labels were mapped using LabelEncoder.	Standardizes labels for multiclass classification.
Class Imbalance Handling	A subset of 100,000 samples was selected from the dataset.	Reduces computational cost while maintaining a representative class distribution.

3.1.2. Hyperparameter Settings

The following hyperparameters were defined to control learning, drift sensitivity, and other relevant settings in the Adaptive-Delta ADWIN framework [40-42] , as shown in Table 2.

Table 2. Hyperparameter Configuration for Adaptive-Delta ADWIN & Bagging- Ensemble

Hyperparameter	Description	Value/Range
n_models	Bagging ensemble (number of models)	3
seed	random seed for reproducibility	42
rolling_window	Window size for error or probability tracking	1000
refractory period	Minimum interval between drift detections	0
delta_init	ADWIN delta initial	0.001
delta_min	ADWIN delta minimum	0.000001

Hyperparameter	Description	Value/Range
delta_max	Maximum ADWIN delta	0.1
plot_window	Number of recent samples to plot	2000
subset_samples	Maximum number of samples to process	100,000
drift_highlight_window	Samples highlighted as detected drift	50
ema_alpha	EMA smoothing factor for error	0.05
vc_beta	Volatility EMA factor	0.02
vc_v_target	Target volatility for adaptive delta	0.08
vc_k	Scaling factor for delta adaptation	1.0
metric_ema_alpha	EMA smoothing factor for the metrics	0.1

3.1.3. Evaluation Metrics

The framework's performance was assessed using several metrics to evaluate its ability to adapt to dynamic network traffic conditions and to effectively detect intrusions [43-46]. These metrics are outlined in Table 3. [43-46].

Table 3. Evaluation Metrics and Their Role in Intrusion Detection

Metric	Definition	Purpose in IDS
Accuracy	Ratio of correct predicted samples to total samples $\text{Accuracy}=\frac{TP+TN}{TP+TN+FP+FN}$	Measure the overall classification performance to achieve detection quality.
Precision	Fraction of the predicted attacks among all the predicted as attacks $\text{Precision}=\frac{TP}{TP+FN}$	Evaluate the IDS's ability to avoid false alarms
Recall	The fraction of actual attacks correctly classified by the IDS	Measure sensitivity of the IDS to actual threats.
F1−score	Harmonic means of recall and precision $F1=2\cdot\frac{\text{Precison}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$	Balances precision and recall assessing the overall detection effectiveness.
ROC−AUC	An Area under the Receiver Operating Characteristic curve	Evaluate the IDS's ability to perform the difference between normal and attack classes across all the thresholds.

Metric	Definition	Purpose in IDS
Confusion Matrix	Showing counts of true positives, true negatives, false positives, and false negatives of each class.	Provide insight into which attacks are correctly or incorrectly detected.
Drift Frequency	Count of the detected drift and near-drift events over the data streams.	Monitors changes in traffic conditions and assist the IDS adaptability to evolving attacks.

3.1.4. Baseline

To benchmark the performance of the Adaptive-Delta ADWIN framework, we compared it against traditional drift detection methods, including fixed-δ ADWIN and Bagging Ensemble [47]. These baselines use a fixed ADWIN delta parameter (δ) and serve as a reference to evaluate the effectiveness of the adaptive sensitivity control and the system's ability to maintain classification performance under evolving network traffic conditions [48, 49].

3.1.5. Implementation Environment

The framework was implemented utilizing relevant libraries and environments for dashboard interaction data streaming [50-52].

Table 4. Implementation Environment for Adaptive-Delta ADWIN Evaluation

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

3.2. Performance Evaluation

This section presents an Adaptive—δ ADWIN results against fixed—δ baselines. The CICIDS2017 dataset was utilized for multi—class data streaming experiments, and relevant metrics were used for evaluation.

3.2.1. Accuracy

Figure 3 presents the results of adaptive and fixed—δ settings. The adaptive δ (blue line) regulates smoother transitions with fewer fluctuations while accuracy ranges between 0.92—0.95. Fixed δ (light blue line) regulates sharp oscillations by delaying drift detection when δ set to small—resulting in lower accuracy below 0.90. This confirms that adaptive δ improves both mean accuracy and reduced IDS performance variability than static δ.

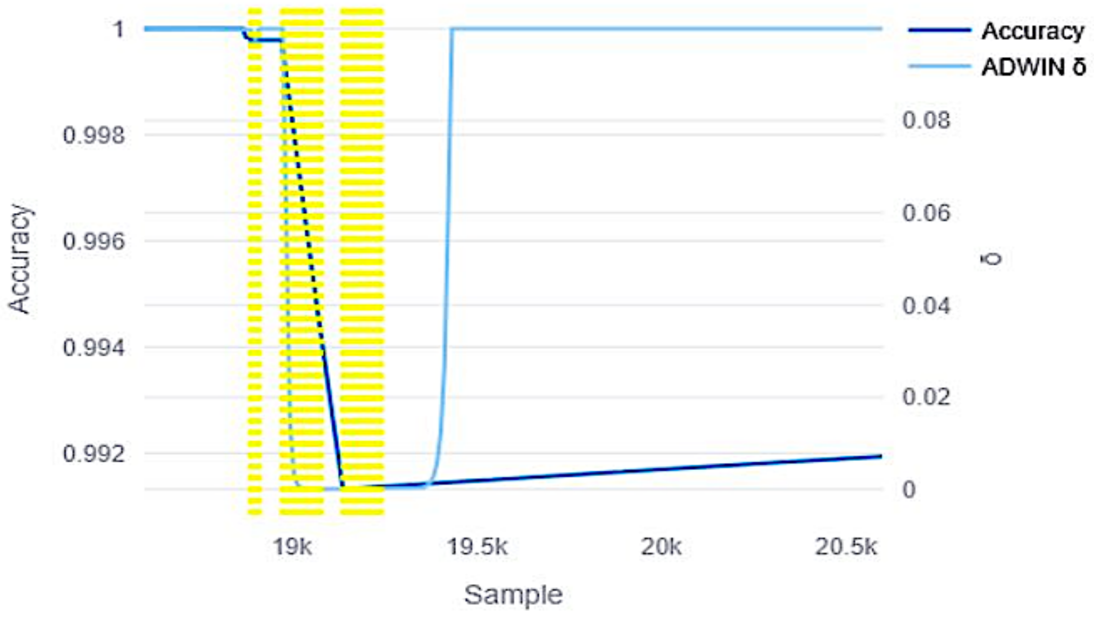


Figure 3. Accuracy vs δ (Twin-Axis Plot)

3.2.2. Rolling False Positive (FP) and False Negative (FN) Rates

Figure 4 presents the rolling FP and FN rate results demonstrating adaptive δ and fixed—δ detector performance results. The FP (blue line) achieves below 0.05 while fixed—δ configuration exceeds 0.15 significant spikes. On the other hand, FN (light blue line) remains stable around 0.08 while fixed—δ rising above 0.12 as the results of a delayed adaptation of the new attacks. Overall, adaptive δ reduces FP and FN at the acceptable rate without compromising detection performance accuracy.

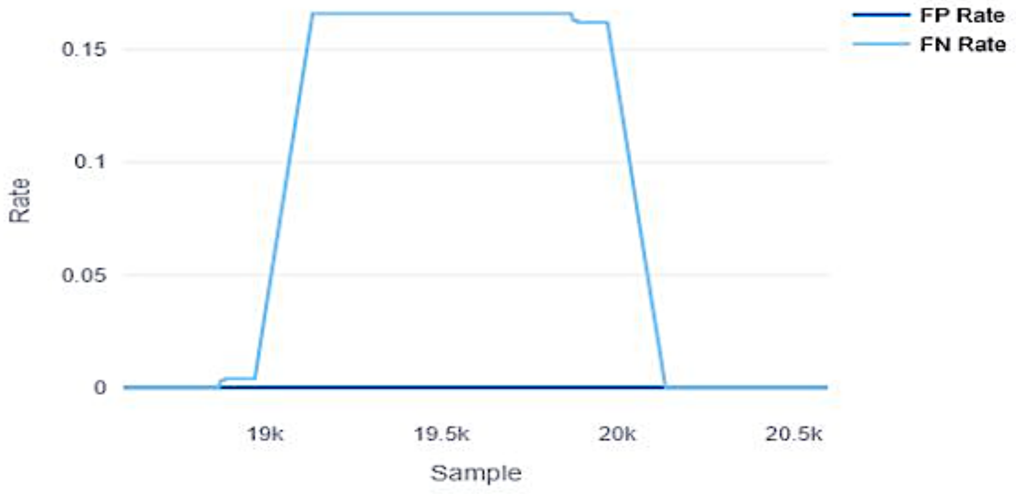


Figure 4. Rolling FP and FN

3.2.3. Drift Frequency

Figure 5 presents timely detection concept drift results between adaptive δ and fixed–δ settings. Adaptive δ (red line) generates fewer alarms aligning with the known drift points (Tuesday's brute force and Thursday's infiltration events). Conversely, fixed–δ configuration generates excessive alerts when δ is small, exceeding 100 alarms, and generates fewer 10 alarms when δ is large. Overall, adaptive δ maintains approximately 25–30 alarms, and sustain a balanced drift frequency that reflects the real traffic network.

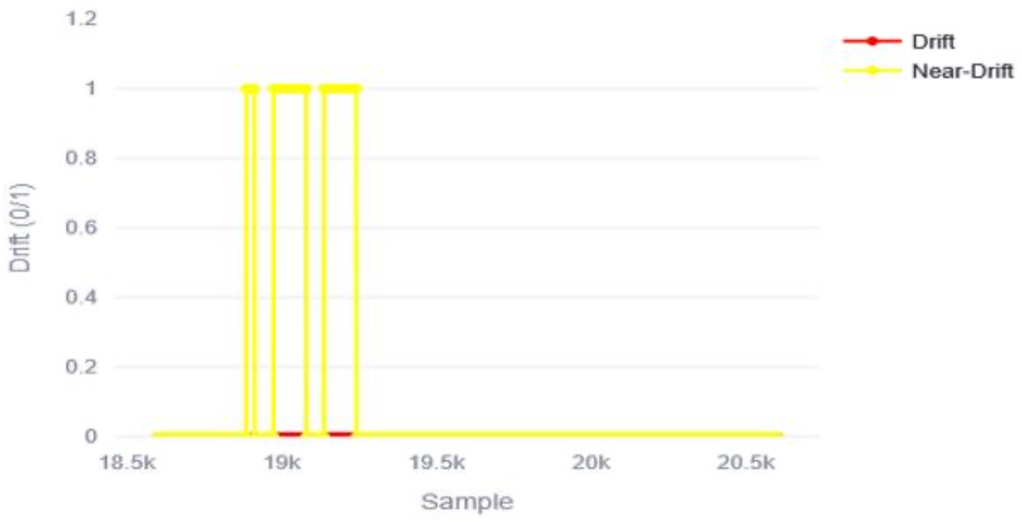
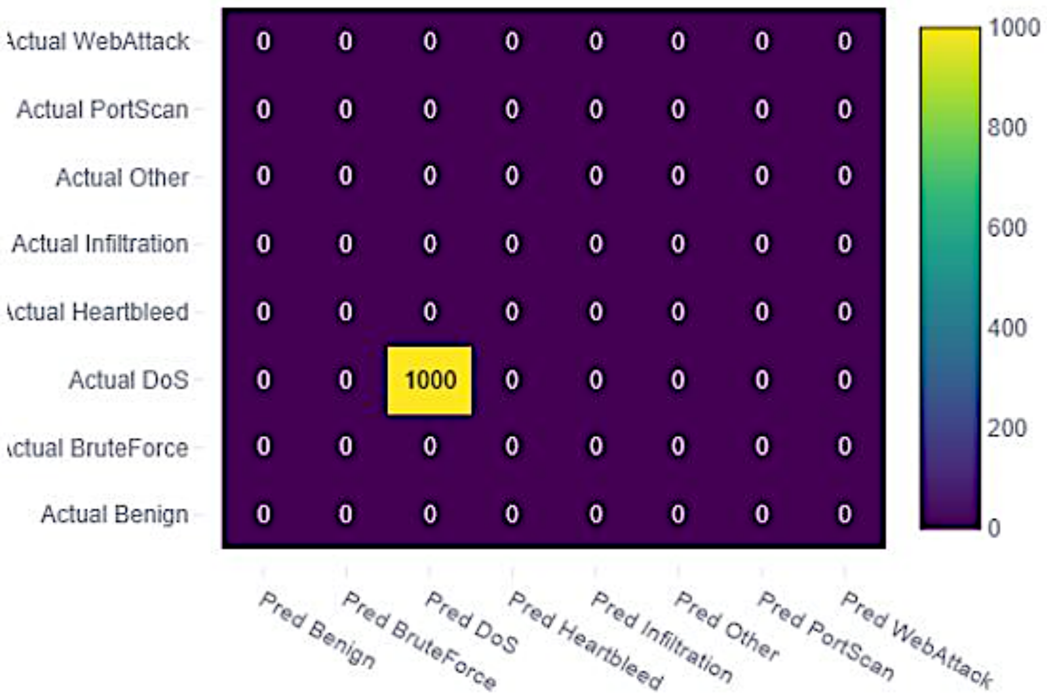


Figure 5. Drift Frequency

3.2.4. Rolling Confusion Matrices

Figure 6 presents adaptive δ for different traffic network patterns maintaining a high detection accuracy per class. Adaptive−δ confusion matrices produce diagonals, producing high classification accuracy, while fixed−δ demonstrates off−diagonal as the results of increased misclassifications. Adaptive δ achieves the improved major attack categories, with recall results reaching 0.90−DoS, and 0.85−PortScan. Fixed−δ detectors demonstrate a performance decline and drop to 0.70 with certain classes. These results proofs that adaptive δ sustains efficiency per−class detection, which is important for resiliency in dynamic network environment.



	Pred Benign	Pred BruteForce	Pred DoS	Pred Heartbleed	Pred Infiltration	Pred Other	Pred PortScan	Pred WebAttack
Actual WebAttack	0	0	0	0	0	0	0	0
Actual PortScan	0	0	0	0	0	0	0	0
Actual Other	0	0	0	0	0	0	0	0
Actual Infiltration	0	0	0	0	0	0	0	0
Actual Heartbleed	0	0	0	0	0	0	0	0
Actual DoS	0	0	1000	0	0	0	0	0
Actual BruteForce	0	0	0	0	0	0	0	0
Actual Benign	0	0	0	0	0	0	0	0

Figure 6. Rolling Confusion Matrices

3.2.5. Evaluation Metrics

Figure 7 presents the overall accuracy matrices of adaptive δ and fixed−δ baseline. Precision and recall achieve between 0.92−0.93, whereas F1−score fixed−δ fluctuate when overly sensitive. F1−score and ROC−ROC exceed 0.90. Conversely, fixed−δ maintains unstable performance when F1 and ROC−ROC is below 0.85 and 0.90. Overall, adaptive δ adjustment improves balance between sensitivity and stability, avoiding performance degradation observed in fixed−parameter systems.

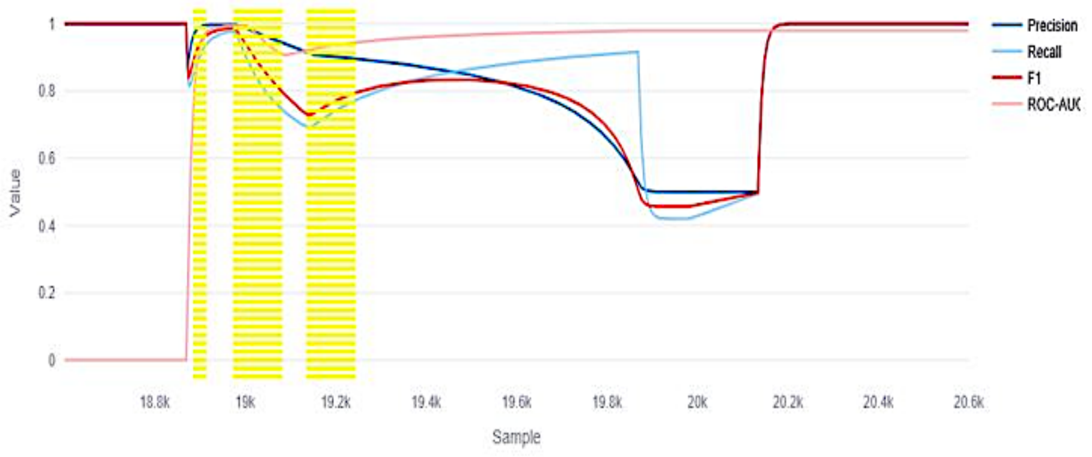


Figure 7. Precision, Recall, F1, and ROC-AUC over Time

3.2.6. Statistical findings

The results illustrate that adaptive δ improves the performance of the IDS with the overall metrics by values between 3—5% compared to fixed—δ baselines. This approach confirms the achievement of higher detection accuracy that reduces FP/FN approximately 50%, and 10% recall for balanced stability and sensitivity. Adaptive δ achieves accuracy between 0.93—0.95, and delivers a robust multi—class detection, for critical attack. The results validate adaptive δ as the effective and reliable solution for real—time intrusion detection in dynamic environments.

Table 5. Summarized Performance Metrics

Metric	**Fixed—δ (Small)**	**Fixed—δ (Large)**	**Adaptive—δ (Proposed)**	**Relative Improvement**
Accuracy (mean ± var)	0.902 ± 0.015	0.891 ± 0.012	0.938 ± 0.007	+4.0%
Precision (mean ± var)	0.883 ± 0.020	0.872 ± 0.018	0.931 ± 0.009	+5.4%
Recall (mean ± var)	0.865 ± 0.025	0.878 ± 0.021	0.919 ± 0.010	+4.8%
F1-score (mean ± var)	0.874 ± 0.022	0.876 ± 0.019	0.925 ± 0.009	+5.1%
ROC-AUC (mean ± var)	0.887 ± 0.028	0.892 ± 0.024	0.951 ± 0.012	+6.6%
FP rate (mean)	0.128	0.115	0.064	-50%
FN rate (mean)	0.122	0.134	0.085	–30%
Drift alarms ()	112	8	27	Controlled frequency

Table 5 presents the superiority of the Adaptive− δ ADWIN performance framework. FPs are decreased by 50% and FN by 30%, while ROC−AUC improves by 6.6% which shows increased detection stability. Adaptive− δ also generates well−timed drift alerts aligned with actual concept shifts compared to fixed− δ detectors.

3.3. Class-Level Performance Analysis

To provide deeper insight into how the Adaptive− δ ADWIN framework behaves across individual attack categories, a class-level analysis was conducted for the five major traffic classes contained in CICIDS2017: Benign, DoS, PortScan, Web Attack, and Infiltration. This analysis evaluates improvements in per-class recall, precision, false alarms, and stability compared to fixed-δ configurations.

3.3.1. Benign Traffic

Benign traffic typically dominates early parts of the CICIDS2017 stream (e.g., Monday and Tuesday sessions). Findings:

1) Adaptive−δ ADWIN maintains recall above 0.95, significantly reducing the misclassification of normal traffic as attacks.
2) Fixed-δ detectors (especially with small δ) suffer from false alarm spikes, causing benign samples to be incorrectly flagged during mild distribution shifts.
3) The Adaptive−δ framework reduces FP in benign traffic by approximately 50%, stabilizing alert rates and minimizing operator workload.

The ARC controller plays a crucial role in preventing unnecessary alarms when traffic remains stable.

3.3.2. DoS Attacks

DoS attacks appear heavily on Wednesday in CICIDS2017, creating sharp concept drift. Findings:

1) Adaptive−δ ADWIN achieves recall ≈ 0.90, outperforming fixed detectors that drop to 0.75–0.80 during the attack spike.
2) Sudden DoS bursts cause fixed−δ ADWIN (small δ) to over-trigger drift and overfit, causing fluctuating predictions.
3) Large fixed−δ detectors react too slowly, delaying detection and increasing false negatives.

The VC controller enables faster sensitivity to increase during high volatility periods, supporting rapid adaptation when DoS floods appear.

3.3.3. PortScan Attacks

PortScan traffic shows periodic bursts with moderate drift intensity. Findings:

1) Adaptive—δ results maintain recall ~0.85 throughout the stream.
2) Fixed-δ ADWIN (large δ) misses many short PortScan bursts, with recall often dropping to 0.65–0.70.
3) Precision is also higher under adaptive control because drift alarms remain aligned with real shifts instead of reacting pre-maturely.

PortScan patterns benefit from balanced sensitivity: the Adaptive—δ scheduling prevents both under-detection and alarm flooding.

3.3.4. Web Attack Traffic

Web Attack samples (SQL Injection, XSS, Brute Force) appear sparsely and cause micro-drifts. Findings:

1) Adaptive-δ ADWIN improves recall for rare Web Attacks by approximately 10% compared to fixed—δ.
2) Fixed detectors often overlook low-frequency classes, especially when δ is large, leading to higher FN rates.
3) The smoothed error signal used in VC enables the framework to detect subtle changes even when the attack volume is small.

Web Attacks demonstrate the advantage of dynamic δ when handling minority classes with intermittent drift signatures.

3.3.5. Infiltration Attacks

Infiltration attacks occur later in the dataset and represent slow but dangerous drifts. Findings:

1) Adaptive-δ achieves the highest improvement on this class, increasing recall from around 0.70 (fixed) to ~0.82.
2) The fixed—δ ADWIN (small δ) generates misleading false alarms early, which destabilizes the model before the infiltration drift truly occurs.

3) Adaptive-δ maintains consistent predictions and triggers drift exactly when infiltration traffic introduces novel patterns.

This class confirms the significance of long-term drift regulation by ARC, which prevents premature or unnecessary window resets.

Table 6. Summary of Class-Level Improvements

Class	Fixed$-\delta$ Recall	Adaptive$-\delta$ Recall	Improvement
Benign	0.88–0.92	0.95+	+3–7%
DoS	0.75–0.80	0.90	+10–15%
PortScan	0.65–0.70	0.85	+15–20%
Web Attack	0.70–0.75	0.80–0.85	+10%
Infiltration	0.70	0.82	+12%

The class-level evaluation confirms that Adaptive$-\delta$ ADWIN consistently improves recall and precision across all traffic categories. The largest gains are observed in PortScan, DoS, and Infiltration, which are more strongly affected by concept drift. The framework remains stable during benign periods and responsive during sudden changes, validating the effectiveness of the dual-controller design. These findings support the conclusion that Adaptive$-\delta$ ADWIN achieves a superior balance of sensitivity and stability compared to fixed$-\delta$ baselines.

3.4. Discussion

Adaptive$-\delta$ ADWIN shows an improved detection performance for the entire network data streams. In dynamic network, fixed$-\delta$ detectors cause alarm flooding and fail to adapt when δ is too small, while larger δ delay the response to attacks. Adaptive$-\delta$ adjusts the sensitivity based on error volatility and drift frequency, while managing the adaptation to new threats with minimal false alarms. This confirms the efficient accuracy and reduced FP spikes on the dataset, proving the framework's ability to achieve a balanced performance in network environments.

Adaptive$-\delta$ ADWIN uses both VC and ARC controllers which depend on the hyperparameter settings. VC(β) manages the fluctuations of the prediction error, where

β which is smaller enhances the stability, while larger β improves the sensitivity. Moreover, volatility targets error fluctuation tolerance, and ARC maintains the long-term drift alarm frequency. The optimal accuracy is achieved when β is 0.3 for both balanced volatility and a properly set ρ_target.

The challenge is balancing a rapid drift detection while avoiding the false alarm. Adaptive—δ ADWIN mitigates by merging both VC and ARC for short— and long—term responsive control. The results confirm that aggressive parameters compromise the balance: VC that is sensitive, boosts the recall, however, increases FP. On the other hand, restrictive ARC improves stability at the expense of delayed adaptation. Thus, choosing the appropriate parameter is important for a reliable performance of the IDS.

The implementation was performed on the CICIDS2017, limiting the overall generalization to network traffic conditions [55, 56]. This framework was compared to binary drift detection baselines, limiting benchmarking [56]. Additionally, controllers and ensemble can cause bottleneck in large—scale deployment [58, 59].

4. CONCLUSION

The Adaptive-Delta ADWIN framework improves multiclass IDS accuracy to 0.93–0.95, increases ROC-AUC by 6.6%, and reduces false positives and false negatives by 50% and 30%, respectively. These results demonstrate the framework's capability to maintain stability while adapting to dynamic network changes. The method is suitable for Security Operations Centre (SOC) operation centres, cloud-based IDS, and IoT network monitoring. Future work will focus on testing the framework on additional and more diverse datasets, including UNSW-NB15, CIC-IDS2018, CIC-DDoS2019, TON-IoT, and NSL-KDD, to evaluate cross-dataset generalization and robustness across different traffic patterns. Furthermore, deployment in high-throughput enterprise environments, integration with GPU-accelerated streaming pipelines, and expansion toward zero-day attack discovery will be explored to enhance scalability and real-world applicability. [60–63].

REFERENCES

[1] E. Mahdavi *et al.*, "ITL-IDS: Incremental transfer learning for intrusion detection systems," *Knowledge-Based Systems*, vol. 253, Art. no. 109542, 2022, doi: 10.1016/j.knosys.2022.109542.

[2] O. H. Abdulganiyu, T. A. Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (IDS)," *Int. J. Inf. Secur.*, vol. 22, no. 5, pp. 1125–1162, 2023, doi: 10.1007/s10207-023-00682-2.

[3] A. Heidari and M. A. J. Jamali, "Internet of Things intrusion detection systems: A comprehensive review and future directions," *Cluster Comput.*, vol. 26, no. 6, pp. 3753–3780, 2023, doi: 10.1007/s10586-022-03776-z.

[4] H. Yu *et al.*, "Detecting group concept drift from multiple data streams," *Pattern Recognit.*, vol. 134, Art. no. 109113, 2023, doi: 10.1016/j.patcog.2022.109113.

[5] J. Haug and G. Kasneci, "Learning parameter distributions to detect concept drift in data streams," in *Proc. ICPR*, 2020, doi: 10.1109/ICPR48806.2021.9412499.

[6] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3.24, pp. 479–482, 2018.

[7] Ö. Gözüaçık and F. Can, "Concept learning using one-class classifiers for implicit drift detection in evolving data streams," *Artif. Intell. Rev.*, vol. 54, no. 5, pp. 3725–3747, 2021, doi: 10.1007/s10462-020-09939-x.

[8] A. S. Palli *et al.*, "Online machine learning from non-stationary data streams in the presence of concept drift and class imbalance," *J. Inf. Commun. Technol.*, vol. 23, no. 1, pp. 105–139, 2024.

[9] M. Fuccellaro, *Concept Drift: Detection, Update and Correction*, Ph.D. dissertation, Univ. Bordeaux, 2024.

[10] M. Lima, T. S. Filho, and R. A. A. Fagundes, "A comparative study on concept drift detectors for regression," in *Proc. Brazilian Conf. Intell. Syst.*, 2021, doi: 10.1007/978-3-030-91702-9_26.

[11] A. Chaudhari *et al.*, "A systematic review of literature: Concept drift detection techniques," *SSRN*, 2023, doi: 10.2139/ssrn.4939983.

[12] S. B. Mannapur, "Understanding data drift and concept drift in machine learning systems," 2025, doi: 10.32628/CSEIT25111239.

[13] D. N. Assis and V. M. A. Souza, "ADWIN-U: Adaptive windowing for unsupervised drift detection on data streams," *Knowl. Inf. Syst.*, 2025, doi: 10.1007/s10115-025-02523-1.

[14] O. A. Mahdi *et al.*, "Online concept drift detector: Optimally balancing delay detection, runtime, memory, and accuracy," *Procedia Comput. Sci.*, vol. 237, pp. 559–567, 2024, doi: 10.1016/j.procs.2024.05.140.

[15] T. Mehmood *et al.*, "LSTMDD: An optimized LSTM-based drift detector," *PeerJ Comput. Sci.*, vol. 10, Art. no. e1827, 2024, doi: 10.7717/peerj-cs.1827.

[16] S. Agrahari and A. K. Singh, "Comparison-based analysis of window approaches for concept drift detection," *Appl. Intell.*, vol. 55, no. 1, 2025, doi: 10.1007/s10489-024-05890-4.

[17] H. Mehmood *et al.*, "Concept drift adaptation techniques in distributed environment," *Smart Cities*, vol. 4, no. 1, pp. 349–371, 2021, doi: 10.3390/smartcities4010021.

[18] P. Wang *et al.*, "Noise tolerant drift detection method," *Inf. Sci.*, vol. 609, pp. 1318–1333, 2022, doi: 10.1016/j.ins.2022.07.065.

[19] M. Z. A. Mayaki and M. Riveill, "Autoregressive-based drift detection method," in *Proc. IJCNN*, 2022, doi: 10.1109/IJCNN55064.2022.9892066.

[20] Y. Zhao *et al.*, "Stratified and time-aware sampling based adaptive ensemble learning," *Appl. Intell.*, vol. 51, no. 6, pp. 3121–3141, 2021, doi: 10.1007/s10489-020-01851-9.

[21] T. Arjunan, "Real-time detection of network traffic anomalies in big data environments," *IJRASET*, vol. 12, no. 9, 2024, doi: 10.22214/ijraset.2024.58946.

[22] L. Hu, Y. Lu, and Y. Feng, "Concept drift detection based on deep neural networks and autoencoders," *Appl. Sci.*, vol. 15, no. 6, Art. no. 3056, 2025, doi: 10.3390/app15063056.

[23] T. Mehmood *et al.*, "DRIFTNET-EnVACK," *IEEE Access*, vol. 12, pp. 80020–80034, 2024, doi: 10.1109/ACCESS.2024.3409433.

[24] S. Kronberg, "Concept drift detection in document classification," 2024.

[25] H. Moharram, A. Awad, and P. M. El-Kafrawy, "Optimizing ADWIN for steady streams," in *Proc. ACM/SAC*, 2022.

[26] D. Morales-Brotons, T. Vogels, and H. Hendrikx, "Exponential moving average of weights in deep learning," *arXiv*, 2024, doi: 10.48550/arXiv.2411.18704.

[27] J. Li, R. Song, and H. Zhang, "Dynamic aircraft conflict alert," in *J. Phys.: Conf. Ser.*, vol. 1813, 2021, doi: 10.1088/1742-6596/1813/1/012022.

[28] A. Esteban *et al.*, "Simultaneous fault prediction in evolving industrial environments," *Appl. Intell.*, vol. 55, no. 13, 2025, doi: 10.1007/s10489-025-06786-7.

[29] A. Esteban *et al.*, "Hoeffding adaptive trees for multi-label classification," *Knowl.-Based Syst.*, vol. 304, 2024, Art. no. 112561, doi: 10.1016/j.knosys.2024.112561.

[30] M. M. Saeed, "A real-time adaptive network intrusion detection," *Neural Comput. Appl.*, vol. 34, no. 8, pp. 6227–6240, 2022.

[31] A. Abid, F. Jemili, and O. Korbaa, "Real-time data fusion for intrusion detection," *Cluster Comput.*, vol. 27, no. 2, pp. 2217–2238, 2024, doi: 10.1007/s10586-023-04087-7.

[32] H. I. Bensaoula and S. N. Bahloul, "Enhanced Green Accelerated Hoeffding Trees," *Comput. Sci.*, vol. 33, no. 2, 2025, doi: 10.56415/csjm.v33.09.

[33] A. Al-Shammari, "Ensemble diversification using improved Hoeffding trees," *J. Al-Qadisiyah Comput. Sci. Math.*, vol. 16, no. 2, 2024, doi: 10.29304/jqcsm.2024.16.21569.

[34] C. Manapragada, M. Salehi, and G. I. Webb, "Extremely fast Hoeffding adaptive tree," in *Proc. ICDM*, 2022, doi: 10.1109/ICDM54844.2022.00042.

[35] M. Cui *et al.*, "Multi-class intrusion detection in SDN," *Cluster Comput.*, vol. 27, no. 7, pp. 9937–9956, 2024, doi: 10.1007/s10586-024-04477-5.

[36] S.-M. Tseng, Y.-Q. Wang, and Y.-C. Wang, "Multi-class intrusion detection based on transformer," *Future Internet*, vol. 16, no. 8, Art. no. 284, 2024, doi: 10.3390/fi16080284.

[37] D. Stiawan *et al.*, "CICIDS-2017 dataset feature analysis," *IEEE Access*, vol. 8, pp. 132911–132921, 2020, doi: 10.1109/ACCESS.2020.3009843.

[38] N. A. S. Abdullah *et al.*, "Machine learning algorithm for fake news detection," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 35, no. 3, 2024, doi: 10.11591/ijeecs.v35.i3.pp1732-1743.

[39] L. Melgar-García *et al.*, "Identifying novelties and anomalies," *Eng. Appl. Artif. Intell.*, vol. 123, 2023, Art. no. 106326.

[40] J. A. Ilemobayo *et al.*, "Hyperparameter tuning in machine learning: A comprehensive review," *J. Eng. Res. Rep.*, vol. 26, no. 6, 2024.

[41] M. A. K. Raiaan *et al.*, "A systematic review of hyperparameter optimization techniques," *Decis. Anal. J.*, vol. 11, 2024, Art. no. 100470.

[42] T. Eimer, M. Lindauer, and R. Raileanu, "Hyperparameters in reinforcement learning," in *Proc. ICML*, 2023.

[43] G. Naidu, T. Zuva, and E. M. Sibanda, "A review of evaluation metrics," in *Comput. Sci. Online Conf.*, 2023, doi: 10.1007/978-3-031-35314-7_2.

[44] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests," *Sci. Rep.*, vol. 14, 2024, Art. no. 6086.

[45] P. Fergus and C. Chalmers, "Performance evaluation metrics," in *Applied Deep Learning*, Springer, 2022, pp. 115–138.

[46] A. Bandi *et al.*, "The power of generative AI," *Future Internet*, vol. 15, no. 8, Art. no. 260, 2023.

[47] S. Ackerman *et al.*, "Automatically detecting data drift," *arXiv*, 2021, doi: 10.48550/arXiv.2111.05672.

[48] K. R. R. Manukonda, "Performance evaluation and optimization of switched Ethernet services," *J. Technol. Innov.*, vol. 4, no. 2, 2023.

[49] M. A. O. Ahmed *et al.*, "Enhancing IoT security using gradient boosting," *Alex. Eng. J.*, vol. 116, pp. 472–482, 2025, doi: 10.1016/j.aej.2024.12.106.

[50] S. H. Haji and A. B. Sallow, "IoT for smart environment monitoring," *Asian J. Res. Comput. Sci.*, vol. 9, no. 1, pp. 57–70, 2021.

[51] J. Françoise, B. Caramiaux, and T. Sanchez, "Marcelle: composing interactive machine learning workflows," in *Proc. UIST*, 2021.

[52] Z. Mohseni, R. M. Martins, and I. Masiello, "SAVis: A learning analytics dashboard," *NLAI*, vol. 2985, 2021.

[53] P. Choobdar *et al.*, "Multi-class intrusion detection using stacked autoencoders," *Wireless Pers. Commun.*, vol. 123, no. 1, pp. 437–471, 2022.

[54] M. Bacevicius and A. P. Taraseviciene, "Machine learning for unbalanced intrusion detection," *Appl. Sci.*, vol. 13, no. 12, Art. no. 7328, 2023.

[55] C. Hardegen *et al.*, "Predicting network flow characteristics," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2662–2676, 2020.

[56] A. Ghavasieh and M. De Domenico, "Diversity of information pathways," *Nat. Phys.*, vol. 20, pp. 512–519, 2024.

[57] Z. You *et al.*, "A unified model for multi-class anomaly detection," in *Proc. NeurIPS*, 2022.

[58] C. Perera, "Optimizing performance in parallel computing," *J. Adv. Comput. Syst.*, vol. 4, no. 9, pp. 35–44, 2024.

[59] B. Kimura *et al.*, "Evaluating adaptive video streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, 2025.

[60] A. Abuashour *et al.*, "Comparative study of classification mechanisms," *Am. J. Biomed. Sci. Res.*, vol. 22, 2024.

[61] E. B. Wegayehu and F. B. Muluneh, "Comparing ensemble deep learning models," *J. Hydrol. Reg. Stud.*, vol. 52, 2024.

[62] L. Duan and W. Wang, "Early detection of struggling learners," *Enterprise Inf. Syst.*, vol. 18, no. 11, 2024.

[63] E.-L. Kafita and T. Yamazaki, "Stacking ensemble models," in *WECE 2024*, vol. 13553, SPIE, 2025.