Minimax Theory and its Applications

Volume 11 (2026), No. 2s, 780 to 803

ADWIN-Guided Simulated Annealing–Optimized Online Gradient Boosting for Malware Detection under Controlled Concept Drift

Anita Venugopal[1], Dr. N. S. Kavitha[2], Mahalakshmi [3], Arul Selvan M[4], Mr.A.Raja[5], Dr.M.S.Nidhya[6], sivaganga Badipati[7]

[1]Assistant Professor, Department of IT Dhofar University ,Sultanate of Oman, Email: anita@du.edu.om, Orcid id:0000-0002-5284-8500

[2]Assistant Professor (SG) /IT Dr. N. G. P. Institute of Technology ,Coimbatore, Tamilnadu, India, Email: nskavi17@gmail.com, ORCID ID: 0000-0002-8876-1415

[3]Assistant Professor, Department of Biomedical Engineering, KIT -Kalaginarkarunanidhi Institute of Technology, Coimbatore , Email: mahakarthick238@gmail.com

[4]Assistant Professor, Department of Information Technology, Nehru Institute of Engineering and Technology, Coimbatore, Tamil Nadu 641105. India. Email: arul2591@gmail.com, ORCID ID: 0000-0003-4778-2356

[5]Assistant professor of Computer Science and Engineering (Cyber Security), United Institute of Technology, Coimbatore,Tamilnadu, India, Email: rajagct@gmail.com

[6]Associate Professor School of Computer Applications, Dayananda Sagar University, Bangalore. Email: nidhyaphd@gmail.com

[7]Assistant Professor Koneru Lakshmaiah Education Foundation, Deemed-to-be University Vaddeswaram, Guntur District, Andhra Pradesh, Email: sivagangabadipati@gmail.com, ORCID: 0000-0003-0270-6587

Abstract

Machine-learning-based malware detectors can suffer performance degradation when malicious software evolves and the relationship between Portable Executable (PE) features and class labels changes. This work proposes an ADWIN-Guided Simulated Annealing–Optimized Online Gradient Boosting framework for malware detection under controlled concept drift. The framework employs LightGBM as the gradient-boosting classifier, ADWIN to monitor the online prediction-error stream and identify statistically significant drift, and simulated annealing to optimize model hyperparameters and the malware decision threshold after a drift alarm. Experiments were conducted using a balanced 120,000-instance subset of EMBER 2018 version-2 static PE features, comprising 2,381 numeric features. The data were divided into 50,000 initial-learning instances, 50,000 controlled streaming instances, and 20,000 independent holdout instances. Gradual feature drift was reproducibly injected into 25% of malware instances in the latter half of the stream. ADWIN detected the first drift event at stream index 25,983, corresponding to a detection delay of 983 samples after the injected onset. Following drift-triggered simulated annealing optimization and model adaptation, the proposed approach achieved 96.12% post-alarm F1-score, 97.82% malware recall, and 99.44% AUROC, compared with 94.70%, 93.97%, and 98.91%, respectively, for the static LightGBM baseline. The adaptive model reduced mean false negatives from 30.24 to 10.96 per 1,000-sample batch, representing a 63.76% reduction in missed malware, while the false-positive rate increased from 4.44% to 5.75%. On the independent clean holdout set, the adapted model retained 95.67% F1-score and 99.32% AUROC. The findings demonstrate that ADWIN-triggered simulated-annealing adaptation can improve malware-detection resilience under controlled gradual feature drift.

Keywords: ADWIN, concept drift, gradient boosting,LightGBM, malware detection, online learning, Portable Executable (PE) malware, simulated annealing, streaming classification, EMBER 2018.

1. Introduction

The continually evolving realm of malware remains a security threat to endpoints, enterprise networks, and critical digital infrastructure. To evade detection from signature or machine learning-based detection mechanisms, malware authors constantly modify various aspects of their payloads, including their packing, the imported functions, their byte patterns, runtime behaviours, and evasion techniques. Static classifiers trained often on historical data can give a strong performance at the beginning. When new harmful samples arise, however, when the feature distribution of the newly reported samples is different from that of the labelled, the classifier will no longer be strong. Concept drift refers to a situation whereby the statistical properties of a

dataset change over time. In other words, there is a shift in class-conditional feature distributions, and/or relationships between explanatory features and the class label. The classifiers of malware suffer from drift since attackers modify their artifacts to avoid detection.

Modern malware analysis now utilizes machine learning algorithms for the identification of malicious software through the observation of behavioural patterns. Gradient boosted decision tree models are very useful at modelling high-dimensional structured data. Moreover, they can capture nonlinear interactions amongst features which may be cumbersome to encode in a manually created signature. Over the years, the EMBER dataset has become a de facto benchmark for this task, offering static feature representations gathered from Windows Portable Executable files. The information mentioned above was gathered from PE headers, properties of imported libraries and functions, byte and string histograms, file entropy-related characteristics, and many other static characteristics that are useful for malicious-file classification.

Despite a high accuracy rate delivered by static ensemble classifiers at the beginning of the execution of the modelling strategy of man-in-the-middle (MitM) attacks, the deployment of a fixed model in a changing threat environment suffers from a fundamental limitation. An ordinary model is trained once, is assessed on a test data set that is fixed, and is left unchanged. In the event that the fixed decision boundaries are changed by the future input stream, the benign file's malwares may not remain easily separable. In real-life malware detection, such degradation may cause two main issues. On the one hand, it causes false negatives, not detecting malware files. This also leads to false alarms, which compel security analysts to spend money on investigations." Recent work on drift-aware malware shows that the static classifier which is used to detect malware deteriorates when new malware variants drift away from the training distribution. In contrast, a drift-triggers retraining can preserve efficacy and necessitate fewer updates when compared to periodic retraining.

A simple solution is to retrain the model on each new data batch as the data arrives. However, the computational cost of retraining with fixed schedules is inefficient since the underlying data distribution remains stable. If retraining is delayed, performance degradation might persist for a longer duration. Drift detection provides a more adaptive option. It looks at online signals, like classification error, confidence, feature distribution or loss, but only triggers model adaptation when there is enough statistical evidence.

The ADWIN method or Adaptive Windowing is the drift-monitoring technique used in this work. ADWIN maintains a variable-length window of recent observations and seeks to test if the average value of some input stream has significantly changed. If the monitored data are stable, the window can be extended to store more historical evidence. When a major change is identified, the contentious window shrinks to erase old observations. The ADWIN algorithm is useful in monitoring prediction-error streams in on-line classification tasks without having to set the window size manually.

It is not enough to only detect the drift if classifier cannot be changed. When a drift alarm occurs, the model must learn to adapt its settings for the new operation.

The hyperparameters which could impact the predictive performance and operational cost are learning rate, tree complexity, feature-sampling ratio, bagging ratio, the regularization strength and malware decision threshold. When we select these values using a fixed configuration, this may reduce the benefit of retraining; in other words, the configuration that works well before drift could perform less effectively after the shift in the distribution of features.

To overcome this limitation, we use simulated annealing with drift-triggered gradient-boosting adaptation. Simulated annealing is a metaheuristic optimization method that accepts both superior and inferior candidate solutions at controlled rates. This quality of more expansion of hyperparameter search space and prevents the solution from falling into a wrong local solution. The proposed framework aims to optimize an artificial intelligence post-drift LightGBM model using simulated annealing. Thus, configuring gives effective malware-detection with high recall and least false-positives. The original learning data and a new labelled adaptation window are employed in retraining the optimized model when ADWIN detects a change in prediction-error stream.

The EMBER is a static Portable Executable feature dataset that is given to the static analyser. Of more than 2300 numerical features, balanced experimental subset (120000 instances) is split into 3 parts viz. initial-learning, controlled-stream and independent-holdout. Although the processed dataset version used here contains feature vectors and labels and not timestamps and malware-family identifiers that would establish the real temporal arrival order and allow for family-disjoint zero-day evaluation. The development of the streaming scenario is mimicked via ordered micro-batches, and gradual features drift is reproducibly injected in a subset of

malware samples in the controlled stream's later part. The design provides a transparent way to analyse the performance of the model degradation, the ADWIN detection delay and the parameter adaptation following a drift on future-stream recovery performance.

The performance of static LightGBM classifier starts to drop after injecting drift according to result. ADWIN finds the first drift event at a stream index of 25,983, 983 samples or seconds after controlled drift. After alarm, ADWIN-triggered optimization and model retraining lead to enhanced output in adapted model. In particular, the suggested method increases post-alarm F1-score 94.70% to 96.12%. This also enhances malware recall from 93.97% to 97.82% and reduces mean false negatives. The false positive rate grows from 4.44% to 5.75% along with a drop in missed malware of 63.76%.

The study offers the following main contributions.

- A malware-detection framework for controlled streaming based on static features of Windows PE from EMBER 2018 v2.
- A method that utilizes ADWIN to keep track of the online classification error and instigates adaptation if a large concept drift is detected.
- A strategy for selecting post-drift gradient-boosting settings and a malware decision threshold is simulated-annealing-based hyperparameter optimization.
- A protocol for gradual-drift injection that is reproducible and allows controlled evolution of malware features when temporal or malware family metadata is missing.
- A pre-drift, post-drift, independent-holdout evaluation of a static and adaptive LightGBM empirical comparison.
- Controlled gradual drift provides quantitative evidence that drift-triggered simulated annealing adaptation improves malware recall and significantly decreases false negatives.

The rest of this paper is organized as follows. Section 2 discusses work involving malware detection using machine learning, adaptation to concept drift, online learning and metaheuristic hyperparameter optimization. The proposed framework is an online gradient-boosting framework using simulated-annealing optimized by ADWIN. The dataset, controlled streaming setup, drift-injection methodology, test conditions, and performance metrics are described in Section 4. Section 5 reveals and talks about experimental results. Section 6 ends the work and indicates future work.

2. Literature Survey

The demand for machine-learning based security systems to operate in non-stationary settings and adapt to temporal changes has increased significantly over the past few years. In Intrusion Detection Systems (IDSs) and malware classification, changing user behavior, newly emerged communication patterns, attacker evasion techniques, novel code packing approaches, code obfuscation, updated API sets or new strains of malware can lead to classifier effectiveness degradation. Hence, a classifier trained on past data is likely to perform with low accuracy when tested on data with slightly shifted statistical properties; such changes are known as 'concept drift'.

Concept drift is an utmost concern in security as such performance decrease often leads to false negatives and thus successful intrusions remain unnoticed. Numerous techniques, like adaptive windows, online classifiers, retraining, drift-aware learning, etc. Have therefore been proposed in existing literature to continue offering adequate detection capabilities after such drift.

ADWIN-based network intrusion detection system to handle concept-drifting network traffic streams by Rajeswari et al. Has made use of data cleaning, normalization and finally assigned a timestamp attribute to create network data into an online stream. ADWIN is used to detect sudden drift occurrence, then classify whether there should be a split on an observation window and how an observer stream split with regard to time or content. A support vector machine is then trained on network traffic and classified as normal or intrusive.

Data from KDD Cup 1999, CICIDS2017 and classified using various metrics, including accuracy, precision, recall, F1-score and false-positive rate show that ADWIN-SVM detects with superior accuracy to commonly used algorithms, namely, Random Forest, Decision Tree, Naïve Bayes, Logistic Regression, AdaBoost, KNN, and SVM-RBF classifiers. This work establishes ADWIN-based concept detection for security streams; however, the focus is primarily on using an SVM with data in network streams, and does not mention a post-drift meta-heuristic operation to tune the classifier parameters and the final decision threshold.

Adaptive-Delta ADWIN to enhance the sensitivity-stability performance trade-off in stream intrusion detection systems by Sebopelo et al., aimed at the challenge of fixed sensitivity (δ) values to detect malware samples in data streams too cautiously leading to delayed alerts, or too cautiously leading to numerous false alerts. A Volatility Controller and Alert-Rate Controller were designed which automatically adjusted δ according to trends and rates in the data stream to mitigate these limitations. This novel detector was deployed in a framework in conjunction with a multi-class Hoeffding Adaptive Tree ensemble which was evaluated on the CICIDS2017 dataset achieving between 93-95% accuracy, 50% reduction in false-positives and 30% in false-negatives compared to a benchmark at fixed$-\delta$ levels. This article introduces an adaptation at the detector level; however, there is no explicit optimization of a downstream classification model after detecting drift event.

Drift detection based online learning approach for Android malware detection using API call graphs by Hussain and Muzaffar recognized that due to dynamic changes, models trained for former apps perform with lower detection efficiency for updated and upcoming versions. They developed two frameworks employing varying detecting/ classifying algorithms and combinations to achieve a significant classification efficiency. Their study compared ADWIN, DDM and EDDM with machine-learning models in order to find ideal configurations for online detection. They achieve a maximal detection rate of 96.01% accuracy.

While the work emphasizes online learning combined with drift detection for Android malware, there is no particular study of any meta-heuristic method tuning any post-drift classifier parameter including the classification threshold that is done in the proposed framework.

A framework analysing concept drift for network attack detection, entitled Secured adaptive windowing using website data authentication was proposed by Alsuwat et al. Which dealt with network traffic but with primarily attack (DDoS) focused scenarios in addition to user verification. The system achieved impressive prediction rates using a multilayer perceptron gradient decision-tree classifier, balanced against its overhead in resources and performance in real-time by means of multiple metrics namely; accuracy, precision, F1 score, throughput, end-to-end delay, resource utilization, network security, and drift-associated indicators.

Unlike our research which centers on classifying the nature of Windows PE malware samples, this research mainly focusses on network-based intrusion detection. In addition, it does not perform any metaheuristic configuration changes after detected drift occurrence.

Huynh et al. Most appropriately relates to this work's framework, through a large-scale longitudinal analysis using different datasets to study wear and degradation using a LightGBM and MLP classifier models. The researchers simulated an actual-time environment by running experiments across 2017-2020 with a fixed feature set in addition to modelling monthly drift, re-training and, malware-family based false-negatives analysis.

It was demonstrated that for a 2017 trained LightGBM and evaluated on a 2018 dataset, F1-scores degraded by as much as 8.47 per-centage points.

Their study also produced false-negative rates of up to 23.92 per-centage points from newly emerged malware families. When triggered by an ADWIN or DDM alarm for concept drift and then re-training an entire classifier resulted in approximately a 33-35% reduction in re-training frequency compared to fixed periodic re-training and comparable cumulative performance; offering strong support that learning can be made significantly more data-efficient by drift-based adaptation compared to fixed periodic re-training methods. Unlike Huynh et al. Where no re-training parameter tuning was carried out nor the concept of meta-heuristic-based re-training of existing components was explored in their study, the objective herein aims to address the suitability of meta-heuristic-based re-tuning on ADWIN triggered concept drift through a single dataset and compare to ADWIN with fixed learner parameters.

Table 1. Comparative Analysis of Related Work

Study	Security task	Drift-handling	Learning	Main	Limitation

	and data	method	model	contribution	relative to proposed work
Rajeswari *et al.* (2022)	Network intrusion detection using KDD Cup 1999 and CICIDS2017	ADWIN detects sudden drift in timestamp-organized data stream	SVM	Shows ADWIN-SVM as a novel adaptive multi-attribute IDS	The paper focuses on network traffic and SVM without drift-triggered.
Sebopelo (2025)	Multiclass streaming intrusion detection using CICIDS2017	Adaptive-Delta ADWIN with Volatility Controller and Alert-Rate Controller	Hoeffding Adaptive Tree ensemble	Adapts ADWIN sensitivity balancing stability and responsiveness dynamically.	Instead of tweaking the hyperparameters and thresholds of post-drift classifier, it optimizes the sensitivity of detector.
Hussain and Muzaffar (2025)	Android malware detection using API call graphs	Comparative use of ADWIN, DDM, and EDDM	Multiple online ML models	Shows the value of detector–classifier combinations best accuracy is 96.01%	There is no apparent simulated-annealing driven reconfiguration triggered after an initial drift is detected.
Alsuwat*et al.* (2023)	DDoS and network malware-attack detection	Secure Adaptive Windowing with Website Data Authentication	MLPGDT	It merges security authentication, adaptive windowing, and attack classification.	There is no explicit online hyperparameter search for meta-heuristics. The focus is on network/DDoS.
Huynh *et al.* (2026)	Temporal Windows PE malware detection using EMBER 2017/2018 and BODMAS	ADWIN/DDM-triggered retraining; incremental retraining and active learning	LightGBM, Random Forest, MLP	Measures how time can cause damage: Usingdetector-basedretrainingcan mean a 33%–35% reduction in update events.	It is not concerned with optimizing simulated-annealing settings for post-drift boosting and the decision threshold.
ADWIN-guided SA-optimized online gradient boosting	Controlled EMBER 2018 v2 PE-feature stream with injected gradual malware-feature drift	ADWIN monitors prediction errors; alarm triggers SA optimization and retraining	LightGBM gradient boosting	This method optimizes the parameters of the post-drift model and the malware threshold to enhance recovery under drift.	It is not concerned with optimizing simulated-annealing settings for post-drift boosting and the decision threshold.

The papers under review further attest to the growing importance of adaptive learning in data streaming from cybersecurity sources. However, they provide two broad avenues of research; one is to refine existing concept drift detection methods as, illustrated by ADWIN-based windows, or the adaptive-delta concept. The other is to enhance model up-dating methods like incremental re-training, on-line learning algorithms, adaptive ensembles or active-learning. A main drawback is that many methods either deploy a static predictor-model after the

concept drift or only address selection of a detector-classifier pair without systematic adaptation of the predictor model itself.

In view of that, this study proposes an ADWIN driven simulated-annealing optimized on-line gradient boosting process for the detection of malware with controlled concept drift: the ADWIN monitors continuously the prediction-error flow; on detecting a drift event, an estimation data chunk containing labelled samples recently observed after the event is formed, and an optimized LightGBM model is found using simulated annealing against its hyper-parameters & thresholds, and then deployed to subsequent chunks until the occurrence of the next drift. Thus, unlike fixed-modelDeployment strategies or fixed-configuration re-training methods, this study jointly exploits both concept drift detection and adaptive optimization of the predictor's hyper-parameters and classifier's optimal decision threshold.

The experimentation is performed on the basis of a synthetic controlled data streaming setup generated from static PE features of the EMBER 2018 dataset v2. While the utilized post-processed dataset is comprised of about 2,381 numeric features and labels it doesn't possess information regarding the underlying (non-existing) time-order, family-disjoint nature or malware family labels. Nevertheless, gradual feature drift (rather than natural seasonal temporal change or family segregation), is predictably inserted at a late phase of the stream against the samples. Hence, transparency and direct measurement of the cost of drift detection (delay and drift detection recall) the cost-of-correction (recovery time, accuracy, False Negative Rate (FNR) reduction), and trade-offs concerning Fals Positive Rate (FPR), are measurable under controlled condition.

3. Proposed System Model

The ADWIN-Guided Simulated Annealing–Optimized Online Gradient Boosting for Malware Detection Under Controlled Concept Drift that we propose is designed to enable reliable detection of Windows Portable Executable malware when the statistical behavior of incoming samples varies. The framework consists of four functions: (i) feature-based malware classification, (ii) online error monitoring, (iii) drift detection using ADWIN, and (iv) post-drift model adaptation using simulated annealing.

The system receives a sequence of static PE feature vectors, classifies each instance as benign or malware, and observes the prediction error after the actual class label is known. ADWIN issues a drift alarm in response to major changes in the observed error stream. The alarm calls upon simulated annealing to select a suitable configuration for LightGBM and a malware-decision threshold. The classifier is retrained and deployed again for prediction on next streaming batches.

The high-level operational sequence is expressed as:

PE Feature Stream → LightGBM Classification → Prediction Error Stream → ADWIN Drift Monitoring → SA Hyperparameter Optimization → Adapted LightGBM Model

We evaluate the proposed system design with a simulated streaming setup constructed from the EMBER 2018 v2 features. Because the available processed features lack temporal information, hash identities and malware-family labels, we make no assertions about real-world temporal streaming and family-disjoint zero-day detection capabilities. We instead simulate streaming with time-ordered micro-batchesand the reproducibility injected gradual feature drift for observing learning degradation and adaptive recovery capabilities.

3.1 System Architecture

The proposed framework contains five major layers:

1. Input and feature-stream layer
2. Initial gradient-boosting learning layer
3. Online prediction and error-monitoring layer
4. ADWIN-guided drift-detection layer
5. Simulated-annealing adaptation and model-update layer

The system processes each arriving instance x_t, represented as a static PE feature vector, where:

$$x_t = [f_1, f_2, f_3, \dots, f_{2381}]$$

Every feature vector has a binary label associated with.

$$y_t \in \{0,1\}$$

where:

$$y_t = \begin{cases} 0, & \text{if the PE file is benign} \\ 1, & \text{if the PE file is malware} \end{cases}$$

The classifier that was deployed gives malware probability:

$$p_t = P(y_t = 1 \mid x_t)$$

A predicted probability is compared to malware threshold τ to obtain a binary decision.

$$\hat{y_t} = \begin{cases} 1, & p_t \geq \tau \\ 0, & p_t < \tau \end{cases}$$

Initially, the threshold is set to:

$$\tau = 0.50$$

Following the identification of drift by ADWIN, simulated annealing can choose a new threshold to increase malware recall whilst controlling the false-positive rate.

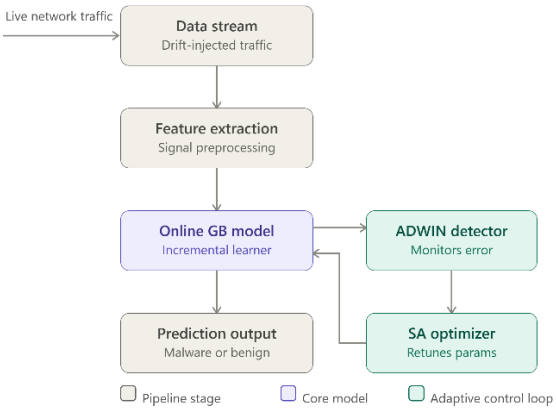


Figure 1. Proposed ADWIN-Guided Simulated Annealing–Optimized Online Gradient Boosting Framework

3.2 Initial Gradient-Boosting Malware Classifier

The first malware classifier uses LightGBM, a gradient-boosted decision-tree method that works well on high-dimensional tabular feature data. Gradient boosting creates an ensemble by adding decision trees that help reduce errors of already existing trees.

Let $F_{m-1}(x)$ represent the ensemble model after m-1 boosting iterations. At iteration m, a new decision tree $h_m(x)$ is added to reduce the loss:

$$F_m(x) = F_{m-1}(x) + \eta h_m(x)$$

where:

- $F_m(x)$ is the updated ensemble score after iteration m;
- $h_m(x)$ is the newly learned decision tree;
- η is the learning rate controlling the contribution of each tree.

The sigmoid function is utilized to convert the ensemble score into a malware probability for binary malware classification.

$$p_t = \sigma(F(x_t)) = \frac{1}{1 + e^{-F(x_t)}}$$

The initial-learning subset is used for training the initial classifier denoted.

$$\mathcal{D}_{init} = \{(x_i, y_i)\}_{i=1}^{Ninit}$$

In the implemented experiment:

$$N_{init} = 50{,}000$$

The baseline LightGBM configuration is defined as:

$$\Theta_{base} = \{\eta, L, d, ff, bf, \lambda_1, \lambda_2, \tau\}$$

where:

- η: learning rate
- L: number of boosting iterations
- d: maximum tree depth
- ff: feature fraction
- bf: bagging fraction
- λ_1: L1 regularization coefficient
- λ_2: L2 regularization coefficient
- τ: malware classification threshold

With feature fraction LightGBM is sampling a fraction of features from the total available when building each tree and with bagging fraction lightGBM is sampling a fraction of training observations. Controls that could help with reducing overfitting and making the training process faster when set appropriately.

3.3 Controlled Streaming and Drift Injection

The EMBER 2018 v2 dataset analysed in the present work consists of 2,381 numerical static PE features and a class label. The labelled data are divided into three subsets.

$$\mathcal{D} = \mathcal{D}_{init} \cup \mathcal{D}_{stream} \cup \mathcal{D}_{holdout}$$

where:

- $\mathcal{D}_{init}$ contains 50,000 initial-learning samples;
- $\mathcal{D}_{stream}$ contains 50,000 streaming samples;
- $\mathcal{D}_{holdout}$ contains 20,000 independent samples for final evaluation.

The streaming subset is arranged as a sequence of micro-batches:

$$\mathcal{D}_{stream} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_T\}$$

where each batch contains 1,000 instances:

$$|\mathcal{B}_t| = 1000$$

Thus, the experimental stream contains:

$$T = \frac{50{,}000}{1000} = 50$$

streaming batches.

Controlled gradual drift is introduced from stream index 25,000 onward. Let t_d denote the drift-onset index:

$$t_d = 25{,}000$$

Drift applies only to malware record in later part of the stream. 8% features is chosen randomly:

$$N_{drift_features} = \lfloor 0.08 \times 2381 \rfloor = 190$$

Among malware samples arriving after t_d, 25% are selected for perturbation. In the implemented controlled stream, this resulted in 3,136 modified malware samples.

For a selected feature f_j of malware instance x_t, the altered feature value is computed as:

$$x'_{t,j} = x_{t,j} + \epsilon_{t,j}\sigma_j$$

where:

- $x'_{t,j}$ is the drifted feature value;
- $x_{t,j}$ is the original feature value;
- $\epsilon_{t,j}$ is Gaussian noise whose intensity increases gradually after drift onset;
- σ_j is the pre-drift standard deviation of feature j.

The gradual perturbation intensity is controlled using:

$$\epsilon_{t,j} \sim \mathcal{N}\left(0, \alpha\left(1 + \frac{t - t_d}{T - t_d}\right)\right)$$

The base disturbance multiplier is α. This controlled plan allows the degradation of the model to be studied under a consistent and transparent drift.

3.4 ADWIN-Based Drift Detection

After prediction, the true label becomes available and the system computes a binary error indicator:

$$e_t = \begin{cases} 1, & \hat{y}_t \neq y_t \\ 0, & \hat{y}_t = y_t \end{cases}$$

The error stream is defined as:

$$\mathscr{E} = \{e_1, e_2, \ldots, e_t\}$$

ADWIN observes this sequence through an adaptive window W containing recent data points. The window is divided into two non-overlapping sub windows.

$$W = W_0 \cup W_1$$

where:

- W_0 represents older observations;
- W_1 represents more recent observations.

Let the mean errors within the two windows be:

$$\mu_{W_0} = \frac{1}{|W_0|} \sum_{e_i \in W_0} e_i$$

and

$$\mu_{W_1} = \frac{1}{|W_1|} \sum_{e_i \in W_1} e_i$$

A drift event is triggered when:

$$|\mu_{W_0} - \mu_{W_1}| > \epsilon_{cut}$$

where ϵ_{cut} is a confidence-bound threshold computed from the window lengths and ADWIN sensitivity parameter δ. In the implemented experiment:

δ = 0.002

ADWIN utilizes an observation window of variable length instead of a fixed-length sliding window. It compares the statistics of old and new sub-windows, signalling drift once they are deemed no longer likely to come from the same distribution. Once detected, old observations are discarded to make the monitoring process responsive to the new operating regime.

In the controlled experiment, the first ADWIN alarm occurred at:

$$t_{alarm} = 25{,}983$$

The detection delay was therefore:

$$D_{delay} = t_{alarm} - t_d = 25{,}983 - 25{,}000 = 983 \text{ samples}$$

This drift alert activates the simulated-annealing adaptation stage.

3.5 Simulated Annealing Hyperparameter Optimization

Immediately after an ADWIN trigger, the framework captures a new window of recently labelled changes:

$$\mathcal{D}_{adapt} = \{(x_t, y_t)\}_{t=t_{alarm}-W_a+1}^{talarm}$$

where W_a is the adaptation-window size. In this work:

$$W_a = 8{,}000$$

The adaptation window divides 75% for optimization-training samples and 25% for validation samples.

$$\mathcal{D}_{adapt} = \mathcal{D}_{SA-train} \cup \mathcal{D}_{SA-valid}$$

where:

$$|\mathcal{D}_{SA-train}| = 6000$$

and

$$|\mathcal{D}_{SA-valid}| = 2000$$

Simulated annealing method traverses the LightGBM parameter space that occurs post-drift.:

$$\Theta = \{\eta, L, d, ff, bf, \tau\}$$

where:

- η: learning rate
- L: number of boosting leaves
- d: maximum depth
- ff: feature fraction
- bf: bagging fraction
- τ: malware decision threshold

The search bounds are defined as:

$$0.02 \leq \eta \leq 0.1215 \leq L \leq 633 \leq d \leq 100.60 \leq ff \leq 1.000.60 \leq bf \leq 1.000.35 \leq \tau \leq 0.65$$

For each candidate configuration Θ_k, LightGBM is trained using $\mathcal{D}_{SA-train}$ and validated using $\mathcal{D}_{SA-valid}$. The candidate objective function balances F1-score, malware recall, and false-positive rate:

$$J(\Theta_k) = 0.50(1 - \mathrm{F1}_k) + 0.35(1 - \mathrm{Recall}_k) + 0.15(\mathrm{FPR}_k)$$

The goal is to minimize:

$$\Theta^* = \arg\min_{\Theta_k} J(\Theta_k)$$

More emphasis on recall is justified because false negative malware predictions are very costly at security. But the false positive term keeps the optimizer away from choosing too permissive a threshold.

At annealing iteration k, let J_c denote the objective cost of the current solution and J_n denote the objective cost of a neighboring candidate. The candidate is accepted if it improves the solution:

$$J_n \leq J_c$$

Otherwise, it may still be accepted with probability:

$$P_{accept} = \exp\left(-\frac{J_n - J_c}{T_k}\right)$$

where T_k is the temperature at iteration k. The temperature decreases according to:

$$T_{k+1} = \gamma T_k$$

where γ is the cooling rate. In this work:

$$T_0 = 1.0$$

$$\gamma = 0.85$$

$$K = 15$$

where K is the maximum number of simulated-annealing iterations.

With stimulated annealing, hyper parameter tuning is viewed as an energy minimization problem, where neighbouring parameter spaces are explored probabilistically (accepting the worst solutions so the program doesn't prematurely get caught in a local minimum).

3.6 Drift-Triggered Model Adaptation

Once simulated annealing identifies the best parameter configuration Θ^*, the model is retrained using the union of initial learning data and the recent adaptation window:

$$\mathcal{D}_{retrain} = \mathcal{D}_{init} \cup \mathcal{D}_{adapt}$$

The total retraining set is:

$$|\mathcal{D}_{retrain}| = 50{,}000 + 8{,}000 = 58{,}000$$

The adapted model is expressed as:

$$M_{adapted} = \mathrm{LightGBM}(\mathcal{D}_{retrain}, \Theta^*)$$

The updated model is subsequently used to classify future stream observations:

$$\{x_t\}_{t=t_{alarm}+1}^{T}$$

In the implemented experiment, simulated annealing selected the following post-drift configuration:

Table 2. Simulated Annealing–Selected Post-Drift LightGBM Configuration

Hyperparameter	**Symbol**	**Selected value**
Learning rate	η	0.0974
Number of leaves	L	47
Maximum tree depth	d	6
Feature fraction	ff	0.9434
Bagging fraction	bf	0.8789
Malware decision threshold	τ	0.3783
SA objective cost	$J(\Theta^*)$	0.067216
SA validation F1-score	F1	92.99%
SA validation recall	Recall	94.74%
SA validation false-positive rate	FPR	9.17%

The optimized threshold is lower than the conventional 0.50 threshold. To increase recall and reduce the number of false negatives, this configuration deliberately classifies additional borderline instances as malware, although it will increase false-positive alerts. As a result, the experimental evaluation measures both detection gains and the false-positive trade-off.

3.7 Algorithmic Workflow

Algorithm 1. ADWIN-Guided Simulated Annealing–Optimized Online Gradient Boosting

Input:

Initial-learning dataset Dinit
Controlled data stream Dstream = {B1, B2, ..., BT}
Initial LightGBM parameters Θbase
ADWIN sensitivity δ
Adaptation window size Wa
SA iteration limit K
Initial temperature T0
Cooling rate γ

Output:

Adapted malware classifier Madapted
Drift-event log Edrift
Batch-level performance measures

Train initial LightGBM model Mbase using Dinit and Θbase
Initialize ADWIN detector using sensitivity parameter δ
Initialize drift-event log Edrift ← Ø
for each incoming batch Bt in Dstream do
    Generate probability pt = Mbase.predict(Bt)
    Assign predicted labels ŷt using malware threshold τ
    Obtain actual labels yt
    Compute error sequence et = I(ŷt ≠ yt)
    Update ADWIN using each error observation in et
    if ADWIN reports drift then
        Record drift position and detection information
        Extract recent labelled adaptation window Dadapt
        Split Dadapt into DSA-train and DSA-valid
        Initialize simulated annealing state Θ0 and temperature T0
        for k = 1 to K do
            Generate neighbouring configuration Θk
            Train LightGBM using DSA-train and Θk
            Evaluate F1-score, recall, and FPR using DSA-valid
            Compute J(Θk)
            Accept or reject Θk using SA acceptance probability
            Update the best configuration Θ*
            Reduce temperature using Tk+1 = γTk

23: end for
24: Retrain Madapted using DinitUDadapt and Θ*
25: Replace Mbase with Madapted
26: end if
27: Record accuracy, precision, recall, F1-score, FPR, AUROC, FN, and FP
28: end for
29: Evaluate final model on independent holdout data
30: Return Madapted, Edrift, and performance results

3.8 Performance Measures

Accuracy Precision Recall F1-score false positive rate area under the receiver operating characteristic curve false negatives drift-detection delay. The proposed methodology is validated by measuring the metrics such as Accuracy, Precision, Recall, F1-score, false positive rate, Area Under the ROC Curve, false negatives, and drift-detection delay.

Accuracy is calculated as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision indicates the proportion of predicted malware files that are actually malicious:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall measures the proportion of actual malware samples correctly detected:

$$\text{Recall} = \frac{TP}{TP + FN}$$

The F1-score balances precision and recall:

$$\text{F1-score} = 2\left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\right)$$

The false-positive rate is defined as:

$$\text{FPR} = \frac{FP}{FP + TN}$$

The false-negative reduction achieved by the adaptive model relative to the static baseline is computed as:

$$\text{FN Reduction} = \frac{FN_{static} - FN_{adapted}}{FN_{static}} \times 100$$

Finally, the detection delay is calculated as:

$$D_{delay} = t_{alarm} - t_d$$

where $t_d$is the controlled drift-onset index and t_{alarm} is the first ADWIN alarm location.

The system model, therefore, links drift detection with parameter optimization and model adaptation. Instead of starting over with a classifier retraining whenever drift happens, the framework explicitly searches for a post-drift configuration that favours malware recall with acceptable false-positive behavior.

4. Implementation And Dataset

The implementation presented hereafter is a setup designed to test and experiment the framework proposed: "ADWIN-Guided Simulated Annealing-Optimized Online Gradient Boosting" using Python with the objective of Windows Portable Executable malware detection within a concept-drift testbed. Specifically, the framework utilizes LightGBM as an efficient classifier model, ADWIN for changes detection of prediction errors and Simulated Annealing for improved model's configuration choice following drift detection.

4.1 Dataset

The experiments were conducted using a dataset known as EMBER 2018. EMBER is a benchmark data set for static malware detection from Windows PE files. In this study, pre-processed Kaggle version with 2,381 numerical features (F1 to F2381) and one target attribute, Label, is used.

The label values are defined as:

$$\text{Label} = \begin{cases} 0, & \text{Benign file} \\ 1, & \text{Malware file} \\ -1, & \text{Unlabelled file} \end{cases}$$

Only legitimate & Malware records were considered for supervise training. The unlabelled records was discarded as we cannot use it for computing classification, accuracy,malware, recall or prediction error for the ADWIN detection in the case of concept drift.

Table 3. EMBER 2018 v2 Dataset Details

Item	Value
Dataset	EMBER 2018 version 2
Data type	Static Windows PE features
Number of features	2,381
Training-file records	799,912
Benign records	299,991
Malware records	299,929
Unlabelled records	199,992
Selected experimental records	120,000
Benign samples in experiment	60,017
Malware samples in experiment	59,983

The experimental subset was almost balanced, had 50.01% benign and 49.99% malware records. The dataset was examined to see if there were any features with missing and infinite values, they were not found.

4.2 Data Partitioning

The selected records of 120,000 were categorised into 3 groups. Stratified sampling was utilized to maintain a similar benign to malware ratio for each group.

Table 4. Experimental Dataset Partitioning

Dataset group	Number of samples	Purpose
Initial training set	50,000	Train the initial LightGBM malware classifier
Controlled stream set	50,000	Simulate incoming malware-detection data
Independent holdout set	20,000	Final model evaluation
Total	120,000	Complete experimental dataset

The stream set was divided into 50 batches, with 1,000 samples per batch:

$$50{,}000/1{,}000 = 50 \text{ batches}$$

Each batch is predicted by the model first. The prediction errors are then used to monitor the drift of the classifier through ADWIN.

4.3 Controlled Concept Drift

The timestamps and malware-family labels are not included in the EMBER dataset after processing. As a result, natural temporal drift could not be evaluated. Controlled gradual drift was instead injected into the later part of the stream in the malware samples.

The drift has begun from 25,000 stream sample data, i.e. With batch 26. Number of randomly selected features is 190 features and represents 8% of 2,381 features. 25% of the selected features are added with Gaussian noise to 25% of the malware samples after the drift point.

Table 5. Controlled Drift Configuration

Parameter	**Value**
Stream size	50,000 samples
Batch size	1,000 samples
Total batches	50
Drift onset	Stream index 25,000
Drift onset batch	Batch 26
Drift type	Gradual feature drift
Modified class	Malware only
Modified features	190
Feature proportion modified	8%
Affected later-stream malware samples	3,136
Noise type	Gaussian noise
Random seed	42

The drifted feature value is represented as:

$$x'_{t,j} = x_{t,j} + \epsilon_{t,j}\sigma_j$$

where $x_{t,j}$ is the original feature value, $x'_{t,j}$ is the changed feature value, σ_j is the standard deviation of feature j, and $\epsilon_{t,j}$ is the generated Gaussian noise.

4.4 Initial LightGBM Model

LightGBM served as the first malware classifier, a gradient boost. The 50,000 initial-training samples were used to train it. The first model's task is predict whether a PE incoming feature vector is benign or malware.

The predicted malware probability is:

$$p_t = P(y_t = 1 \mid x_t)$$

The file is classified as malware when:

$$p_t \geq \tau$$

where τ is the malware decision threshold. The initial static model uses:

$$\tau = 0.50$$

Table 6. Static LightGBM Configuration

Parameter	**Value**
Number of boosting rounds	250
Learning rate	0.05
Number of leaves	31
Feature fraction	0.80
Bagging fraction	0.80
L1 regularization	0.10
L2 regularization	0.10
Malware threshold	0.50
Random seed	42

4.5 ADWIN Drift Detection

The model error for each incoming sample is calculated as:

$$e_t = \begin{cases} 1, & \hat{y}_t \neq y_t \\ 0, & \hat{y}_t = y_t \end{cases}$$

ADWIN received stream of errors, comparing the more recent ones with the more "past" ones through an adaptive windowing approach. If significant change in the error level occurs, ADWIN will output "concept drift".

The ADWIN sensitivity parameter was set as:

$$\delta = 0.002$$

According to first drift alarm, stream index was 25,983 in this experiment. The detection delay occurred as the controlled drift began at the 25,000 index.

$$25{,}983 - 25{,}000 = 983 \text{ samples}$$

Table 7. ADWIN Drift Detection Results

Drift event	Stream index	Batch	Detection delay
First ADWIN alarm	25,983	26	983 samples
Second ADWIN alarm	32,831	33	—

4.6 Simulated Annealing Adaptation

After the first ADWIN alarm, the most recent 8,000 labelled samples were selected as the adaptation window. These samples were divided into:

- 6,000 samples for simulated-annealing model training
- 2,000 samples for validation

Simulated annealing searched for improved LightGBM parameters and a better malware threshold. The optimization considered F1-score, malware recall, and false-positive rate.

The objective function is:

$$J(\Theta) = 0.50(1 - \text{F1}) + 0.35(1 - \text{Recall}) + 0.15(\text{FPR})$$

The objective is minimized to obtain the best parameter configuration.

Table 8. Simulated Annealing Configuration

Parameter	Value
Adaptation window	8,000 samples
SA training samples	6,000
SA validation samples	2,000
SA iterations	15
Initial temperature	1.0
Cooling rate	0.85
Number of boosting rounds per candidate	150
SA runtime	308.97 s

The best parameter combination is shown below.

Table 9. SA-Selected LightGBM Parameters

Parameter	Selected value
Learning rate	0.0974
Number of leaves	47
Maximum depth	6
Feature fraction	0.9434
Bagging fraction	0.8789
Malware threshold	0.3783
Validation F1-score	92.99%
Validation recall	94.74%
Validation FPR	9.17%
Objective cost	0.067216

The chosen threshold is less than 0.50. While it may lead to fewer false negatives, it can support a rise in false-positive alerts.

4.7 Adapted Model Evaluation

The optimized model was retrained using the 50,000 initial-training samples and the 8,000 adaptation-window samples:

$$50{,}000 + 8{,}000 = 58{,}000 \text{ samples}$$

The time to retrain was 45.43 s. Evaluation of the adapted model was performed on subsequent stream batches, following the initial ADWIN alert, and on the separate holdout of 20000 samples. The primary metrics to examine were accuracy, precision, recall, F1-score, false positive rate, AUROC, false negatives and false positives.

The proposed implementation therefore follows a simple sequence:

Initial LightGBM Model → Stream Prediction → ADWIN Drift Detection → Simulated Annealing → Retrained LightGBM Model

This design can have the model adapt to the controlled modification of malware-features, instead of running the model with the original classifier upon drift.

5. Results And Discussion

This section shows the performance of the proposed framework for online gradient boosting using simulated annealing. The discussion of the results takes four steps: baseline performance on the clean holdout set, performance degradation for controlled drift, ADWIN drift detection and post alarm performance of the SA adapted model.

Usage of balanced 120,000-sample subset of EMBER 2018 v2 was done for experiments. The dataset consisted of 50,000 original-training samples, 50,000 controlled-stream samples, 20,000 independent holdout samples. Beginning with Batch 26, perturbations were introduced to selected feature values of later-stream malware records to induce controlled gradual drift. The lack of timestamps or malware-family labels means that the results reflect a drift experiment (controlled) as opposed to an evaluation for family-disjoint zero-day.

5.1 Baseline Performance

The first LightGBM classifier was trained on a 50K sample and evaluated on the independent clean holdout set before drift adaptation. The model exhibited impressive metrics, achieving an accuracy of 95.42%, a precision of 95.43%, a recall of 95.41%, an F1-score of 95.42%, and an AUROC of 99.20%.

Table 10. Static LightGBM Performance on the Independent Clean Holdout Set

Metric	Static LightGBM
Accuracy	95.42%
Precision	95.43%
Recall	95.41%
F1-score	95.42%
False-positive rate	4.57%
AUROC	99.20%
True negatives	9,546
False positives	457
False negatives	459
True positives	9,538

According to the results, the original model has the ability of malware-detection to the same extend for clean data in this condition. The SA-adapted model yields an AUROC of 99.20% indicating the model is well performed in differentiating between benign and malignant PE files. Nevertheless, according to the confusion

matrix, in the clean data set, there are 459 malware samples were misclassified as benign which means they would bypass the detection of security system in the real mal-detection scenario.

From Fig.4, two ROC curves also demonstrate the excellent discriminative ability to the independent clean holdout data set, AUROC of static model, SA-adapted model is 0.9917 and 0.9932, respectively.

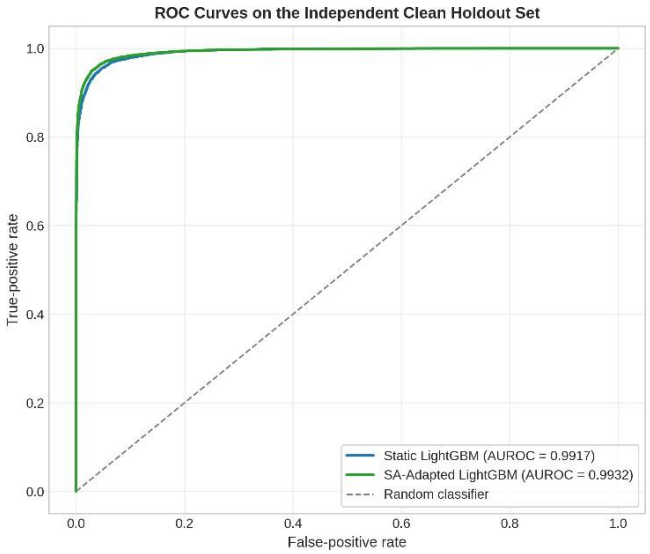


Figure2. ROC curves of static and SA-adapted LightGBM models on the independent clean holdout set.

The SA-model attained a performance score of 0.9932 versus the static light GBM model with the value 0.9917 for the Area Under ROC curve (AUROC).

5.2 Static Model Under Drift

Static LightGBM model performance was compared for 50 controlled streams of batches. A controlled gradual drift was implemented at stream index 25,000 at stream index 26, where there were 1,000 samples per batch.

Prior to drift, the static model achieved a mean F1 score of 95.57%. Post drift injection, the mean F1 score dropped to 94.59%. This decline may seem small in value, but in reality malware recall dropped from 95.51% to 93.71%, and the average number of false positives increase from 22.32 to 31.68 in each batch, demonstrating the loss of ability of the static model at detecting malware with altered feature distribution.

Table 11. Static LightGBM Performance Before and After Controlled Drift

Metric	Pre-drift	Injected-drift	Change
Accuracy	95.60%	94.64%	−0.96 percentage points
Precision	95.64%	95.54%	−0.10 percentage points
Recall	95.51%	93.71%	−1.80 percentage points
F1-score	95.57%	94.59%	−0.98 percentage points
False-positive rate	4.33%	4.41%	+0.08 percentage points
AUROC	99.22%	98.90%	−0.32 percentage points
Mean false negatives per batch	22.32	31.68	+9.36
Mean false positives per batch	21.72	21.96	+0.24

Figure 3. The static classifier's F1-score was computed across all 50 streaming batches (Figure 1). Batches conducted prior to drift maintained F1-score values above 94% while after the start of the drift region, there's an apparent fall. The static model hit nearly 90.6% around Batch 30 at which the lowest F1 score was observed. This shows that a classifier trained on previous versions of the malware may lose its stability when the features change gradually.

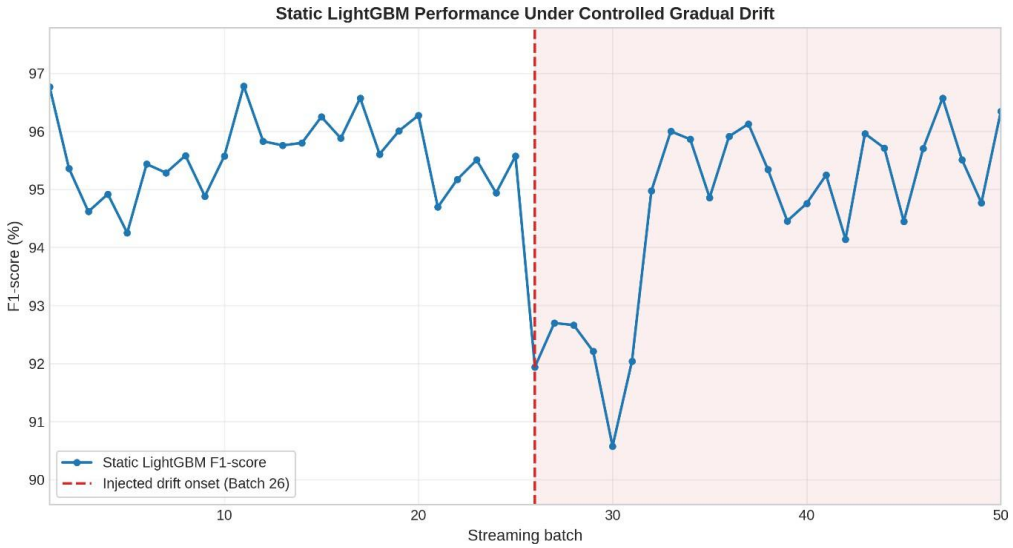


Figure 3. Static LightGBM F1-score across 50 controlled streaming batches.

Starting from Batch 26, controlled gradual drift occurs. The static classifier shows a marked decrease in F1-score in early drifted batches.

Fixed LightGBM shows powerful clean-data performance, but not necessarily the same performance if the incoming malware characteristics change. The decline in recall is significant because the drifted malware samples are more likely misclassified as benign.

5.3 ADWIN Drift Detection

The static classifier's prediction error stream was subject to ADWIN. A value of 1 in the error was assigned to a sample classified incorrectly and a value of 0 for a sample classified correctly. PARAPARSE THIS: ADWIN monitors the errors continuously in order to detect changes which are statistically significant.

Drifting was controlled from stream index 25,000. ADWIN generated its first alarm at index 25,983 or Batch 26. Thus, the first drift detection delay is of 983 samples. On batch 33 at index 32,831, the second alarm was generated. This indicates that the classifier error pattern is still influenced by the slow drift.

Table 12. ADWIN Drift Detection Results

Drift event	Stream index	Batch	Estimated error	Adaptive window width	Interpretation
Controlled drift onset	25,000	26	—	—	Start of gradual feature perturbation
ADWIN alarm 1	25,983	26	4.83%	9,600	First detected distributional change
ADWIN alarm 2	32,831	33	6.19%	3,264	Continued drift effect

The detection delay is calculated as:

$$\text{Detection Delay} = 25{,}983 - 25{,}000 = 983 \text{ samples}$$

The third figure shows the batch-wise classification error rate and the position of the ADWIN alarm. Prior to the controlled drift period, the error values were more or less between 3% and 5.6%. At Batch 26, a spike in the error rate to near 8.1% was observed. The first ADWIN alarm was triggered by this spike. The deviation peaked at approximately 9.2% at Batch 30, which indicated the strongest effect of the gradual change of features on static-model predictions.



Figure 4. Mean classification error across streaming batches with ADWIN drift alarms.

The monitored drift starts from Batch 26. ADWIN senses a drift event at Batch 26 and a second one at Batch 33. This result shows that ADWIN can detect a change in the model error stream right after a drift event takes place, making it applicable to situations when model adaptations should not be performed while there are still no noticeable performance issues but only after a sufficient performance drop is manifested.

5.4 SA-Adapted Model Performance

Once the first ADWIN alarm was issued, we collected the most recent 8,000 labelled samples. We performed simulated annealing on these samples to optimize the LightGBM parameters, along with the malware decision threshold. The re-trained model was evaluated on an updated dataset of 58,000 samples: 50,000 training samples, plus the 8,000 samples from the adaptation window.

The parameters we optimized using Simulated annealing, for a total of 1,000 iterations, are learningrate = 0.0974, num_leaves = 47, max_depth = 6, feature_fraction = 0.9434, bagging_fraction = 0.8789, and malware_threshold = 0.3783.

Due to enhancing recall, we chose the lower malware decision threshold to decrease the number of false negative alerts, though it may increase some false positive alerts in the process.

Table 13. Post-Alarm Performance: Static vs. SA-Adapted LightGBM

Metric	**Static LightGBM**	**SA-Adapted LightGBM**	**Change**
Accuracy	94.76%	96.04%	+1.28 percentage points
Precision	95.50%	94.49%	−1.01 percentage points
Recall	93.97%	97.82%	+3.85 percentage points
F1-score	94.70%	96.12%	+1.42 percentage points
False-positive rate	4.44%	5.75%	+1.31 percentage points
AUROC	98.91%	99.44%	+0.53 percentage points
Mean false negatives per batch	30.24	10.96	−19.28
Mean false positives per batch	22.16	28.68	+6.52

The most notable improvement is in malware recall. The static model had a recall of 93.97% while the SA-adapted model had 97.82%. As a result, more malware samples were detected after drift.

The reduction in mean false negatives is:

$$\text{False Negative Reduction} = \frac{30.24 - 10.96}{30.24} \times 100 = 63.76\%$$

This indicates that the adaptive model was able to reduced missing malware at 63.76% in the post-alarm streams and at the same time improve the F1-score at 1.42%. This explains that the recall gain was not just created by classifying all the files as malware. Fig.5 illustrates the comparison of the F1-score between the static classifier and the SA-adapted models on per-batch basis after the ADWIN alarms occur at time 0. The F1-score with static classifier is low during Batches 26–30 which is roughly from 90% to 91.7% with average of 90.87on the contrary,SA-adaptedmodel obtains high F1-score roughly 95.5%–98% during those batches with average of 96.55.

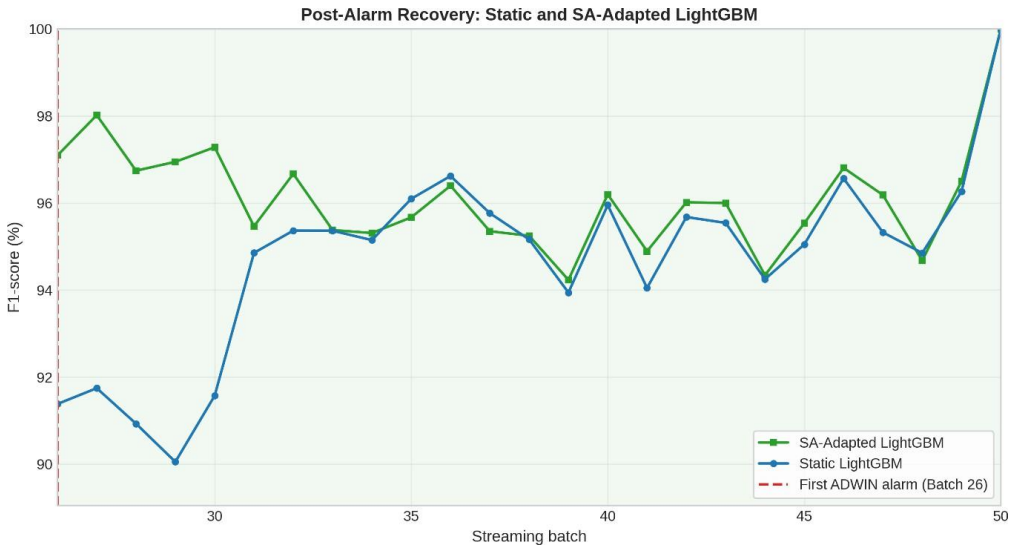


Fig. 5. Post-alarm F1-score comparison between static and SA-adapted LightGBM models.

The F1-score of SA-adapted model preserves its value after the first ADWIN alarm, and especially during early drifted batches.

The main trade-off is the increased rate of false positives. The SA model raised the FPR from 4.44% to 5.75% The increment can be explained by the optimized threshold of 0.3783 being less than the static threshold of 0.50. fewer borderline samples are labelled as malware. While this will generate additional benign-file alerts, it greatly decreases the chance that malware will be misclassified as benign.

This trade-off is justifiable for malware detection with security sensitivity as a false negative is typically more detrimental than moderate uncomfortable increases in false positives. The choice of final threshold must be made as per operational requirement of targeted environment. For instance, an enterprise endpoint-protection system that places a greater emphasis on recall may be desired; whereas an environment limited by analyst capacity may rely on a stricter threshold to limit alerts.

5.5 Independent Holdout Evaluation

The clean holdout set of 20,000 samples was also used to evaluate the adapted model. This assessment confirms that the model's enhancement was not solely due to the adaptation window and that it maintained robust generalization on independent data.

Table 14. Independent Clean Holdout Comparison

Metric	Static LightGBM	SA-Adapted LightGBM	Change
Accuracy	95.42%	95.60%	+0.18 percentage points
Precision	95.43%	94.10%	−1.33 percentage points
Recall	95.41%	97.29%	+1.88 percentage points
F1-score	95.42%	95.67%	+0.25 percentage points
False-positive rate	4.57%	6.10%	+1.53 percentage points
AUROC	99.20%	99.32%	+0.12 percentage points
False negatives	459	271	−188
False positives	457	610	+153

On the clean holdout data, the SA-adapted model reduced false negatives from 459 to 271. The missed-malware reduction is:

$$\frac{459 - 271}{459} \times 100 = 40.96\%$$

The independent holdout set's evaluation returned a recall of 97.29% and a F1-score of 95.67% for the adapted model. This verifies that the adaptation not only performed well overall but also improved malware detection sensitivity.

The false positive rate was observed to be increasing from 4.57 to 6.10% due to recall oriented threshold selected through simulated annealing. Nevertheless, there was an increase in AUROC from 99.20% to 99.32% that indicates that the adapted model also achieved a slightly stronger discrimination between benign and malware files.

5.6 Discussion

The results confirm that the integration of ADWIN with simulated-annealing based parameter optimisation is effective. The performance of the static LightGBM model was high initially, under pure (clean) operation, but dropped significantly under a controlled malware-feature-drift. The F1 score decreased by 0.98pps, malware recall dropped by 1.80pp, and the mean number of false negatives increases by 9.36 per batch, after drift has been introduced, the ADWIN scheme detects the change within short time frame to the drift begin with first alarm occurring 983 samples after the controlled drift begins, illustrating that a continuous monitored learning approach can detect that a previous trained classifier no longer behaves as desired, After the ADWIN has raised alarm simulated annealing is run with respect to re- optimise and select better configuration, using below malware threshold the adapted model increased the F1 score by 1.42pps, malware recall increases by 3.85pps and mean false negatives is reduced by 63.76%, this forms part of the central result that the framework was able to reduce the missed malware.

As SA enables both a tree model parameter adjustment alongside selection of decision thresholds SA is particularly interesting to be compared against existing drift-based methods, Traditional retraining might occur with use of previous parameter settings that may no longer be applicable, SA ensures optimisation based on the data in the labelled window available after drift has been detected.

The results demonstrate that the reduction in recall against false positive does come with trade off of, SA-model reduced the malware but creates an increase number of alerts for benign file detection, this is an inevitable drawback, but should be considered within context of a security environment where the number of benign files to investigate does not, generally, pose as much a threat as unseen malware, and furthermore security systems are in a position to employ different response methodologies when dealing with borderline predictions, which the SA-modified model creates from benign files and malware, sending such cases down to the other investigation procedures such as Sandbox or further analysis. To summarise, the presented method represents a pragmatic strategy for drift based adaptation in a controlled scenario of a static PE malware detection environment when previously established models start showing decline due to differing malware feature landscape, thereby improving post-drift resilient and malicious behaviour identification capabilities while, with only a slight trade-off, making successful reduction in false negatives without compromising on the initial performance of models or the AuROC.

Conclusion and Future Work

In this paper, we introduce an ADWIN-Guided Simulated Annealing–Optimized Online Gradient Boosting for online malware detection under a controlled concept drift. The presented system utilizes LightGBM classifier for predicting maliciousness of PE files, ADWIN for monitoring errors to predict the stream evolution and identify concept drift, and simulated annealing for tuning the model parameters and the malware decision threshold following a concept drift detection. The tests are conducted on a balanced 120,000 samples dataset which consists of 50,000 training initial samples, 50,000 controlled-stream samples, and 20,000 independent validation samples derived from EMBER 2018 v2 static PE features. Clean samples have achieved high performance (Accuracy: 95.42%, Recall: 95.41%, F1: 95.42%, AUROC: 99.20%), while controlled drift negatively affected stream performance especially malware recall. ADWIN detects the initial drift event on index 25,983 which means that the system has a lag of 983 samples till the first drift is detected. As a result of Simulated Annealing selection, a malware recall optimizing threshold 0.3783 and new LightGBM model parameters are chosen, leading to post-alarm F1: 96.12%, Recall: 97.82%, compared with the performance of

the stationary LightGBM model that achieved a F1: 94.70% and Recall: 93.97%. The proposed method significantly reduced the number of missed malware files from 30.24 to 10.96 (63.76%) per batch while increased false positives slightly from 4.44% to 5.75%. In future research directions, we intend to apply the proposed method on a real time chronological malware streams, use malware family level labels instead of just classifying between malicious or clean files, as well as include threat intelligence feeds in the online framework. We plan to further improve the operational utility through adaptive ADWIN sensitivity and/or a multi-objective learning and explainable AI features as well as involving the analyst's perception and expertise to enable humans-in-the-loop in the decision-making process.

References

1. Rajeswari, P. V. N., Shashi, M., Rao, T. K., Rajya Lakshmi, M., & Kiran, L. V. (2022). Effective intrusion detection system using concept drifting data stream and support vector machine. *Concurrency and computation: practice and experience*, *34*(21), e7118.

2. Hussain, M. D., & Muzaffar, A. (2025, August). Online learning-based android malware detection using API call graphs and drift detection: a comparative study. In *Proceedings of the AAAI Symposium Series* (Vol. 6, No. 1, pp. 87-89).

3. Alsuwat, E., Solaiman, S., &Alsuwat, H. (2023). Concept drift analysis and malware attack detection system using secure adaptive windowing. *Computers, Materials and Continua*, *75*(2), 3743-3759.

4. Huynh, T. T., Huynh, D. T., Nguyen, M. T., & Trinh, V. Q. (2026). Temporal Degradation in Machine Learning-Based Malware Detection: A Multi-Dataset, Multi-Year Empirical Study. *IEEE Access*.

5. Wang, K., Lu, J., Liu, A., Song, Y., Xiong, L., & Zhang, G. (2022). Elastic gradient boosting decision tree with adaptive iterations for concept drift adaptation. *Neurocomputing*, *491*, 288-304.

6. Gunasekara, N., Pfahringer, B., Gomes, H., & Bifet, A. (2024). Gradient boosted trees for evolving data streams. *Machine Learning*, *113*(5), 3325-3352.

7. Maillet, W., & Marais, B. (2023). Optimized deep learning models for malware detection under concept drift. *arXiv preprint arXiv:2308.10821*.

8. Augello, A., De Paola, A., & Lo Re, G. (2025). Hybrid multilevel detection of mobile devices malware under concept drift. *Journal of Network and Systems Management*, *33*(2), 36.

9. Molina-Coronado, B., Mori, U., Mendiburu, A., & Miguel-Alonso, J. (2023). Efficient concept drift handling for batch android malware detection models. *Pervasive and Mobile Computing*, *96*, 101849.

10. Alam, M. T., Piplai, A., & Rastogi, N. (2025, October). ADAPT: a pseudo-labeling approach to combat concept drift in malware detection. In *2025 28th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)* (pp. 693-712). IEEE.

11. Alam, M. T., Fieblinger, R., Mahara, A., & Rastogi, N. (2024). Morph: Towards automated concept drift adaptation for malware detection. *arXiv preprint arXiv:2401.12790*.

12. Escudero García, D., & DeCastro-García, N. (2023, August). Application of anomaly detection models to malware detection in the presence of concept drift. In *International Conference on Hybrid Artificial Intelligence Systems* (pp. 15-26). Cham: Springer Nature Switzerland.

13. Korycki, Ł., & Krawczyk, B. (2023). Adversarial concept drift detection under poisoning attacks for robust data stream mining. *Machine learning*, *112*(10), 4013-4048.

14. Darem, A. A., Ghaleb, F. A., Al-Hashmi, A. A., Abawajy, J. H., Alanazi, S. M., & Al-Rezami, A. Y. (2021). An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning. *IEEE Access*, *9*, 97180-97196.

15. Li, A. S., Iyengar, A., Kundu, A., & Bertino, E. (2024). Revisiting concept drift in windows malware detection: Adaptation to real drifted malware with minimal samples. *arXiv preprint arXiv:2407.13918*.

16. Wang, Q., Wang, L., & Zhao, W. (2026). A Classification Framework and Research Progress on Adaptation Methods for Concept Drift in Malicious Code Detection Models. *Future Internet*, *18*(5), 231.

17. García, D. E., DeCastro-García, N., & Castañeda, A. L. M. (2023). An effectiveness analysis of transfer learning for the concept drift problem in malware detection. *Expert systems with Applications*, *212*, 118724.