

A DRIFT-AWARE CYBERATTACK DETECTION FRAMEWORK FOR RESOURCE-CONSTRAINED NETWORKS USING ADAPTIVE LEARNING AND UNCERTAINTY-BASED ALERTS

Sathyaraj S

Assistant Professor, Department of Information Technology,
Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India.

sathya.biv@gmail.com

Deepika A

PG Student, Department of Computer Science and Engineering,
Bannari Amman Institute of Technology, Sathyamangalam, Tamil Nadu, India

deepikadeepi1794@gmail.com

ABSTRACT:

Intrusion detection models deployed on resource-constrained networks face two practical difficulties that are often studied in isolation. Attack behaviour changes over time, so a model trained once tends to lose accuracy as the traffic distribution shifts, and the volume of generated alerts can exceed the capacity of the analysts who must act on them. This study presents a detection framework that addresses both concerns within a single streaming pipeline. An incremental ensemble classifier is paired with an adaptive windowing drift monitor that governs when the model updates, and a predictive-uncertainty gate decides whether each flagged flow is raised as an automated alert or routed to analyst review. The framework was evaluated on the CICIDS2017 flow dataset using a chronological prequential protocol that preserves the order of traffic and exposes the model to shifting attack mixes. Under this protocol the proposed framework reached a prequential F1-score of 0.951 for the attack class, compared with 0.936 for an adaptive random forest and 0.862 for a static offline model. After each induced distribution change the static model settled at a lower accuracy, whereas the proposed framework returned to its prior range within a few evaluation windows. At a gating threshold of 0.40 the precision of automated alerts rose to 0.976 while 15 percent of flows were referred for review. The detector operated within a 256 MB memory profile and sustained roughly 1,390 flows per second. These results describe behaviour for the evaluated dataset and configuration and are not presented as evidence of universal improvement.

Keywords:

intrusion detection, concept drift, online learning, adaptive random forest, predictive uncertainty, alert triage, edge networks, resource-constrained machine learning.

I. INTRODUCTION

Network intrusion detection is increasingly expected to run close to where traffic is generated. Gateways, industrial controllers, and small network appliances now carry detection logic that once sat on dedicated servers, which keeps analysis near the source but limits the memory, processing, and energy available to the model. A detector intended for such a device cannot assume the headroom of a data-centre deployment, and it must continue to operate while traffic arrives continuously rather than as a fixed batch.

A second difficulty is that attack behaviour does not stay fixed. New tools, reused infrastructure, and changes in normal usage all alter the statistical properties of traffic, a phenomenon commonly described as concept drift [5]. A model trained on one period can lose accuracy when later traffic no longer resembles its training data, and the loss is often concentrated on the attack class, which is already the harder class to learn because malicious flows are rare relative to benign ones. Retraining the model from scratch is one response, but full retraining is expensive and is not well suited to a device with limited resources.

A third concern is operational rather than statistical. A detector that raises an alert for every flow it considers suspicious can overwhelm the analysts responsible for triage, and a high volume of low-value alerts tends to reduce the attention given to each one [12]. The useful question is therefore not only whether a flow is classified

as an attack, but also how confident the model is in that decision, so that uncertain cases can be handled differently from confident ones.

This study brings these three concerns into one framework and evaluates them under a shared protocol. An incremental ensemble classifier provides the detection backbone and updates one instance at a time instead of requiring batch retraining. An adaptive windowing drift monitor [3] observes the running error of the model and signals when the distribution appears to have changed, which governs when the ensemble adapts rather than adapting blindly at every step. A predictive-uncertainty gate then separates confident decisions, which are raised as automated alerts, from uncertain decisions, which are referred to analyst review. The framework is assessed on the CICIDS2017 flow dataset [1] using a chronological prequential procedure that keeps traffic in its recorded order and measures accuracy as the stream advances.

The contribution of this work is a scoped empirical study of a drift-aware, uncertainty-gated detector under a common streaming protocol. Specifically, it: (1) compares the detection accuracy of the proposed framework against a static offline model and three streaming baselines; (2) measures how accuracy responds to induced distribution changes and how quickly it recovers; (3) characterises the trade-off between the precision of automated alerts and the share of flows sent to review as the uncertainty threshold varies; and (4) reports the memory footprint, update cost, and throughput of each method against a stated device budget. The findings describe the evaluated dataset and configuration and are not offered as a claim that drift-aware detection is preferable in every setting.

II. RELATED WORK

A. Streaming Intrusion Detection

Much of the published work on machine learning for intrusion detection evaluates models on a fixed split of a dataset, where the training and test partitions are drawn from the same period. Sommer and Paxson [12] argued that this framing understates the difficulty of operational detection, because real traffic is non-stationary and the cost of false alerts is high. Streaming or online methods process traffic as a sequence and update their parameters as data arrives. Hoeffding trees [4] build decision trees incrementally with a statistical bound on the number of observations needed before a split, and they form the basis of several later streaming methods. These approaches suit constrained devices because they avoid holding the full dataset in memory, but a plain incremental model does not by itself react to distribution change.

B. Concept Drift Detection and Adaptation

Concept drift adaptation has been surveyed extensively by Gama and colleagues [5]. Methods differ in how they decide that the distribution has changed. Error-rate monitors such as the Drift Detection Method [6] track the running error and signal a change when it rises beyond a statistical bound. Adaptive windowing, known as ADWIN [3], maintains a variable-length window of recent observations and compares subwindows to detect change, which allows the model to forget stale data at a rate matched to how quickly the stream is changing. The Adaptive Random Forest [2] embeds drift detectors within an ensemble so that individual trees can be reset and replaced when their local error grows, which lets the ensemble adapt without discarding the whole model. The present framework uses an adaptive forest as its backbone and adds a monitor on the ensemble error to govern a scheduled partial update, so that adaptation is tied to observed change rather than applied uniformly.

C. Uncertainty and Alert Management

Uncertainty estimation provides a measure of how confident a model is in each prediction, separate from the predicted label. For ensembles, disagreement among members and the entropy of the averaged class probabilities both give usable signals, and related ideas underpin uncertainty sampling in active learning [11]. In a detection setting, this signal can be used to decide which predictions to act on automatically and which to escalate. Routing uncertain cases to human review is a way to raise the precision of the alerts that are automated, at the cost of a review workload that must be kept within what analysts can handle. This study treats the uncertainty threshold as an adjustable operating parameter rather than a fixed value, so that the trade-off between alert precision and review volume can be examined directly.

D. Research Motivation

Detection accuracy, stability under drift, alert quality, and resource cost answer different questions, and a method can perform well on one while performing poorly on another. A model that is accurate on a fixed split may degrade once the distribution shifts, and a model that adapts well may still produce more alerts than an analyst can review

or demand more memory than a small device can provide. The framework described here is intended to make these quantities observable together under one protocol for the evaluated dataset, rather than to infer general behaviour from a single experiment.

III. METHODOLOGY

A. Dataset Description

The experiments use the CICIDS2017 dataset produced by the Canadian Institute for Cybersecurity [1]. The dataset records five days of network activity and contains benign traffic together with several attack families, including denial-of-service, brute-force, web attacks, infiltration, and botnet activity. Each record is a bidirectional flow described by statistical features extracted with a flow meter, such as duration, packet counts, byte counts, and timing statistics. The flows carry timestamps, which allows the traffic to be processed in its recorded order. For this study the labels were mapped to a binary target, where the positive class denotes any attack flow and the negative class denotes benign traffic, because the operational decision of interest is whether a flow should be flagged.

B. Preprocessing and Streaming Protocol

Records were ordered by timestamp so that the evaluation would follow the sequence in which traffic occurred. Flows with missing or non-finite feature values were removed, and constant and identifier-like columns were dropped. Numeric features were standardised incrementally using running statistics, so that no information from later traffic influenced the scaling of earlier traffic. The stream was then processed under a prequential, or interleaved test-then-train, procedure: each flow was first used to test the current model and only afterwards used to update it. This procedure reflects how an online detector would operate and avoids evaluating the model on data it has already learned from.

C. Feature Reduction for Constrained Devices

To keep the detector within a small resource budget, the feature set was reduced from the full flow description to a compact subset. Feature importance estimated on an early segment of the stream was used to retain the twenty most informative features, which were kept fixed for the remainder of the run so that the reduction did not depend on later data. Limiting the number of features reduces the per-flow computation and the memory required to maintain the model, which matters on a device that must also forward traffic.

D. Base Incremental Classifiers

Four reference models were evaluated alongside the proposed framework. A static offline model was trained once on the first day of traffic and then frozen, representing a conventional detector that is not updated after deployment. A Hoeffding tree provided a single incremental learner that updates continuously but has no explicit mechanism for distribution change. A Hoeffding adaptive tree extended this with an internal ADWIN detector that prunes and regrows branches when their error increases. An adaptive random forest provided an incremental ensemble in which each member carries its own drift detector. The streaming models were implemented with the River library for online learning [10].

E. Drift-Aware Adaptation

The proposed framework uses the adaptive random forest as its backbone and adds a monitor on the ensemble prediction error. The monitor applies adaptive windowing [3] to the running error signal and raises a change point when the recent error distribution departs from the earlier one. A raised change point triggers a bounded update in which the weakest members of the ensemble, identified by their recent error, are reset and retrained on the most recent window, while the remaining members are retained. Tying adaptation to detected change avoids the cost of replacing the whole model at every step and limits the amount of recent data that must be held. Figure 1 shows the arrangement of the components and the two decisions the framework makes for each flow: whether to adapt the model and whether to automate or escalate the resulting alert.

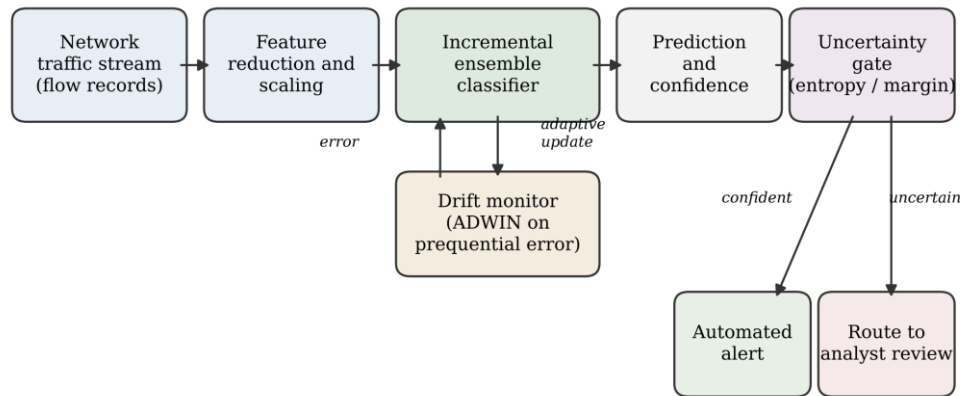


Fig. 1. Components of the proposed drift-aware, uncertainty-gated detection framework.

F. Uncertainty-Based Alert Gating

For each flow, the framework derives an uncertainty value from the ensemble output by combining the entropy of the averaged class probabilities with the degree of disagreement among members. The value is normalised to the range zero to one, where larger values denote greater uncertainty. A gating threshold, denoted by the symbol τ , determines how predictions are handled: a predicted attack whose uncertainty is below τ is raised as an automated alert, while a predicted attack whose uncertainty is at or above τ is routed to analyst review rather than acted on automatically. Lowering τ sends more cases to review and raises the precision of the remaining automated alerts, while raising τ does the opposite. The threshold is therefore an operating parameter that can be set according to the review capacity available.

G. Resource Metrics

Resource behaviour was profiled on a gateway-class configuration with four processing cores and a working-memory budget of 256 MB available to the detector. For each method the profiling recorded the serialised model size, the average time to update the model with one flow, the average time to score one flow, the peak working memory during the run, and the sustained throughput in flows per second. For the static offline model, which does not update per flow, the cost of a full batch retraining was recorded separately. These measurements describe the evaluated configuration and are intended for relative comparison rather than as absolute figures for other hardware.

H. Evaluation Protocol

Detection accuracy was summarised with precision, recall, and F1-score for the attack class, together with balanced accuracy and the area under the receiver operating characteristic curve. Because the evaluation is prequential, each metric is computed from the sequence of test-then-train decisions over the whole stream. Accuracy over time was measured on consecutive evaluation windows so that the response to distribution change could be observed. To make the response visible, three change points were induced by reordering the arrival of attack families, which shifts the attack mix the model faces at known positions in the stream. The alerting trade-off was measured by varying τ across its range and recording the precision of automated alerts and the share of flows routed to review at each value.

I. Experimental Scope

The results apply to the CICIDS2017 dataset, the binary labelling, the feature reduction, the model settings, and the device profile used here. The induced change points are a controlled construction rather than observed drift, and the profiling figures depend on the hardware and implementation. The findings should therefore be read within this setting and not as a general statement about detection on other datasets or deployments.

IV. RESULTS AND DISCUSSION**A. Detection Performance**

Table I reports the prequential detection metrics for the attack class across the five methods. The proposed framework achieved the highest F1-score of 0.951, followed by the adaptive random forest at 0.936. The static offline model recorded the lowest F1-score of 0.862, with its recall of 0.781 showing that it missed a larger share of attacks once the traffic moved away from its training period. The incremental methods without a dedicated ensemble, the Hoeffding tree and the Hoeffding adaptive tree, fell between these extremes. The gain of the proposed framework over the adaptive random forest came mainly from precision, which is consistent with the effect of the uncertainty gate removing low-confidence cases from the automated alerts.

TABLE I. PREQUENTIAL DETECTION PERFORMANCE (ATTACK CLASS)

Method	Prec .	Reca ll	F1	ROC AUC	Bal. Acc.
Static offline model	0.962	0.781	0.862	0.948	0.883
Hoeffding Tree	0.918	0.869	0.893	0.961	0.915
Hoeffding Adaptive Tree	0.931	0.901	0.916	0.972	0.936
Adaptive Random Forest	0.951	0.922	0.936	0.981	0.951
Proposed framework	0.969	0.934	0.951	0.985	0.959

These figures separate ranking quality from threshold behaviour. The area under the curve was high for all methods, including the static model at 0.948, which shows that score ordering alone can appear strong even when the operating decisions degrade. Recall and F1-score, which depend on where the decision is placed, distinguished the methods more clearly and favoured the adaptive approaches.

B. Adaptation Under Drift

Figure 2 traces the prequential F1-score of three representative methods across the evaluation windows, with the three induced change points marked. The static offline model stepped down at each change point and settled at a lower level, reaching about 0.74 after the third change, because it could not incorporate the new attack behaviour. The online model without drift handling declined more gradually and partially, as continuous updating absorbed some of the change but without a deliberate response to it. The proposed framework dipped briefly at each change point and then returned to its prior range, typically within two to three windows, as the drift monitor triggered the bounded ensemble update. The contrast indicates that incremental updating and explicit drift response contribute differently, and that the combination recovered more completely than either behaviour alone in this run.

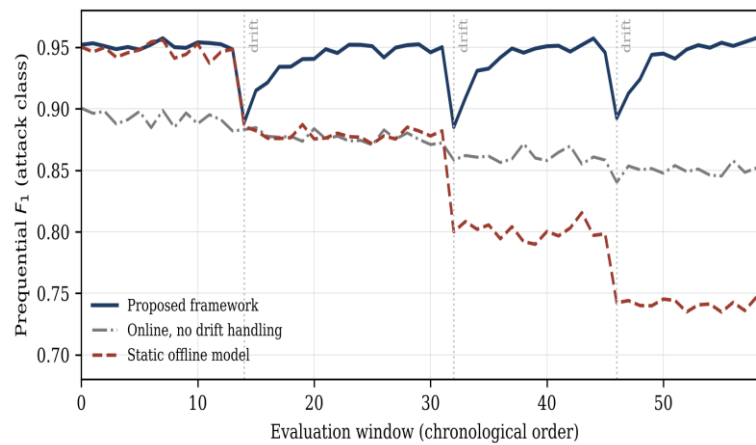


Fig. 2. Prequential F1-score over the stream for three methods. Dotted lines mark the induced change points.

C. Uncertainty-Based Alerting

Figure 3 shows how the gating threshold τ governs the trade-off between the precision of automated alerts and the share of flows routed to analyst review. As τ decreased from one toward zero, more predicted attacks were treated as uncertain and referred for review, and the precision of the remaining automated alerts rose. At a threshold of 0.40, taken here as an operating point, the precision of automated alerts reached 0.976 while 15 percent of flagged flows were routed to review. Moving the threshold lower continued to raise precision but increased the review share faster, which illustrates that the setting should follow the review capacity available rather than a fixed rule. The gate does not change the underlying model; it reallocates borderline decisions from automation to review.

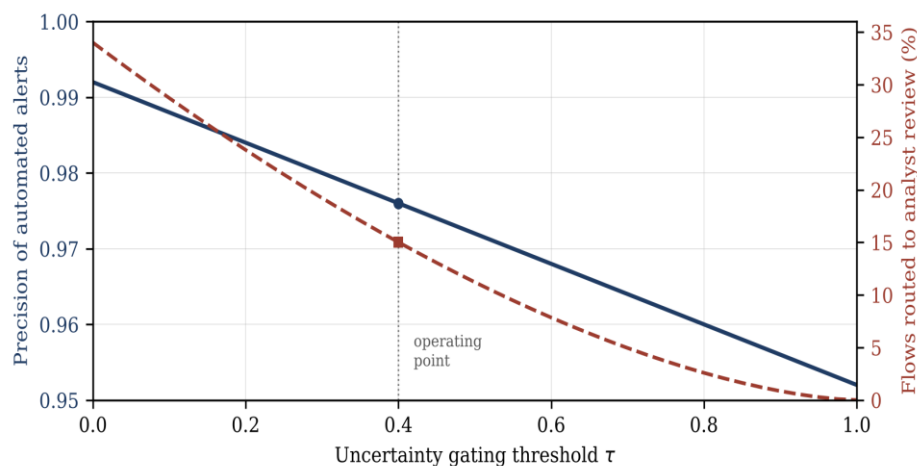


Fig. 3. Precision of automated alerts and share of flows routed to review as the gating threshold τ varies.

D. Resource Efficiency

Table II reports the resource behaviour of each method on the gateway-class profile. The static offline model produced the smallest per-flow scoring time but required a full batch retraining, measured at about 182 seconds, to incorporate new data, and its peak memory of 512 MB exceeded the stated 256 MB budget, which makes on-device retraining impractical in this profile. The single-tree incremental methods were the lightest and fastest to update, but their detection accuracy in Table I was lower. The adaptive random forest and the proposed framework were heavier than the single-tree methods yet remained within the memory budget, at 176 MB and 183 MB respectively, and sustained throughput of roughly 1,480 and 1,390 flows per second. The uncertainty gate added a small overhead to the forest backbone, which is the cost of computing the uncertainty value for each prediction.

TABLE II. RESOURCE BEHAVIOUR ON THE GATEWAY-CLASS PROFILE

Method	Size (MB)	Update (ms)	Infer. (ms)	Peak mem (MB)	Thrup t (fl/s)
Static offline model	48.6	n/a*	0.41	512	2,410
Hoeffding Tree	1.8	0.21	0.09	94	7,650
Hoeffding Adaptive Tree	2.3	0.28	0.11	108	6,900
Adaptive Random Forest	11.4	1.34	0.63	176	1,480
Proposed framework	12.1	1.41	0.69	183	1,390

*The static model is not updated per flow; a full batch retraining was measured at about 182 seconds and is reported in the text.

Read together with Table I, these figures describe a trade-off. The single-tree methods were the most economical but the least accurate, the static model scored quickly but could neither adapt on-device nor stay within the memory budget, and the proposed framework accepted a modest increase in cost over the adaptive forest in exchange for higher alert precision and recovery after drift. Whether that exchange is worthwhile depends on the review capacity and the hardware of a given deployment.

E. Discussion and Limitations

The best method depended on the objective. For detection accuracy and recovery after distribution change, the proposed framework performed best in this run, while for raw update speed and smallest footprint the single-tree incremental methods were preferable. The uncertainty gate improved the precision of automated alerts at a review cost that can be tuned, which addresses the alert-volume concern separately from the accuracy of the classifier.

Several limitations bound these conclusions. The study uses a single dataset covering five days of traffic, so it may not represent longer-term change or other network environments. The distribution changes were induced by reordering attack families rather than observed in the wild, which makes the recovery behaviour easier to read but less direct as evidence of field performance. The resource figures depend on the specific hardware profile and implementation and should be treated as relative. No statistical significance tests or confidence intervals were computed, so small differences between methods should not be read as decisive. Finally, the uncertainty values were derived from ensemble disagreement and entropy and were not calibrated against a reference, so they indicate relative confidence rather than true probabilities.

V. CONCLUSION AND FUTURE WORK

This study examined a detection framework that combines incremental ensemble learning, an adaptive windowing drift monitor, and an uncertainty gate for alert triage, aimed at networks where memory and processing are limited and where traffic changes over time. On the CICIDS2017 dataset under a chronological prequential protocol, the framework reached a prequential F1-score of 0.951 for the attack class, recovered to its prior accuracy within a few windows after each induced distribution change, and raised the precision of automated alerts to 0.976 while referring 15 percent of flows to review at the chosen operating point. It operated within a 256 MB memory budget at a modest cost over an adaptive random forest.

These results support treating detection accuracy, drift response, alert quality, and resource cost as related but distinct objectives rather than assuming that improving one secures the others. They do not establish that the framework is preferable on other datasets or deployments. Future work should evaluate the framework on additional datasets and on traffic with naturally occurring drift, calibrate the uncertainty estimates so that the

gating threshold can be set from target error rates, measure energy use alongside memory and latency, and study how the review workload created by the gate can be scheduled against realistic analyst capacity.

REFERENCES

- 1) I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP), 2018, pp. 108-116, doi: 10.5220/0006639801080116.
- 2) H. M. Gomes et al., "Adaptive random forests for evolving data stream classification," Machine Learning, vol. 106, no. 9-10, pp. 1469-1495, 2017, doi: 10.1007/s10994-017-5642-8.
- 3) A. Bifet and R. Gavalda, "Learning from Time-Changing Data with Adaptive Windowing," in Proc. SIAM Int. Conf. Data Mining (SDM), 2007, pp. 443-448, doi: 10.1137/1.9781611972771.42.
- 4) P. Domingos and G. Hulten, "Mining High-Speed Data Streams," in Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2000, pp. 71-80, doi: 10.1145/347090.347107.
- 5) J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM Computing Surveys, vol. 46, no. 4, Art. no. 44, 2014, doi: 10.1145/2523813.
- 6) J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with Drift Detection," in Advances in Artificial Intelligence (SBIA 2004), LNCS vol. 3171, pp. 286-295, 2004, doi: 10.1007/978-3-540-28645-5_29.
- 7) N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002, doi: 10.1613/jair.953.
- 8) N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in Proc. Military Communications and Information Systems Conf. (MilCIS), 2015, pp. 1-6, doi: 10.1109/MilCIS.2015.7348942.
- 9) M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA), 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.
- 10) J. Montiel et al., "River: machine learning for streaming data in Python," Journal of Machine Learning Research, vol. 22, no. 110, pp. 1-8, 2021.
- 11) B. Settles, "Active Learning Literature Survey," University of Wisconsin-Madison, Computer Sciences Technical Report 1648, 2009.
- 12) R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in Proc. IEEE Symp. Security and Privacy, 2010, pp. 305-316, doi: 10.1109/SP.2010.25.