Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

An Adaptive Support Vector Machine Approach for Explainable Anomaly Detection in Internet of Things Networks

Erenso Hundessa Muleta

erensohundessa735@gmail.com

Addis Ababa University

Research Article

Keywords: Internet of Things, anomaly detection, intrusion detection, support vector machine, explainable artificial intelligence, concept drift, edge computing

Posted Date: November 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11506403/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: Competing interest reported. non finance

An Adaptive Support Vector Machine Approach for Explainable Anomaly Detection in Internet of Things Networks

Original research article

Author: Erenso Hundessa Muleta[1],*

Affiliation: [1] Department of Computer Science, Addis Ababa University, Ethiopia

Corresponding author: *Erenso Hundessa Muleta, Department of Computer Science, Addis Ababa University, Ethiopia. E-mail: erensohundessa735@gmail.com

Target journal: Journal of Information Security and Applications (JISA)

Date of preparation: 23 October 2026

Keywords: Internet of Things; anomaly detection; intrusion detection; support vector machine; explainable artificial intelligence; concept drift; edge computing

Manuscript length: about 11,800 words, 47 references, 10 tables and 2 figures

Note to the authors. All results in this manuscript were produced by the accompanying code on a documented simulated IoT traffic benchmark, with ten independent seeds, and every table is generated from the output files of that code. The four public datasets named in earlier drafts and the Raspberry Pi 4 and Jetson Nano devices were not available in the environment where the experiments were run, so they are not used and no result is claimed for them. Before submission you should (i) repeat the experiments on at least two public datasets with the same code, (ii) measure latency, memory and energy on a real edge device, (iii) verify every reference against the publisher's page and add DOIs, (iv) complete the declarations, and (v) apply the current template of the Journal of Information Security and Applications, which is an Elsevier journal and is therefore submitted through Elsevier's system and not through Springer. The journal also asks for highlights in a separate file.

Abstract

Detecting attacks in Internet of Things (IoT) networks is difficult because four requirements must be met at once. Attack traffic is rare, benign behaviour drifts as devices and firmware change, the detector often has to run on a gateway with little memory, and analysts need to know why an alert was raised. This paper studies the Adaptive Explainable Support Vector Machine (AE-SVM), a streaming kernel detector that combines a drift-aware kernel width, class costs adapted from observed recall, a hard budget on the number of support vectors enforced by weighted prototype compression, and exact gradient-based explanations of the decision function. We evaluate it on a documented simulated IoT traffic benchmark with four device classes, five attack families, gradual and abrupt drift, a change in device mix and an attack family that appears mid-stream, using ten independent seeds, a chronological protocol with delayed labels, five ablation variants, a sweep over the memory budget and fourteen baseline configurations. The results are mixed, and we report them as they are. After an abrupt shift, AE-SVM lost 0.031 in F1 whereas static supervised baselines lost between 0.226 and 0.437, and it was significantly better than the static SVM after the shift (mean difference 0.186, ten of ten seeds). However, the periodically retrained baselines lost the same 0.03 to 0.05 and reached an F1 of 0.65 to 0.73 against 0.475 for AE-SVM, so the robustness comes from updating and not from the adaptive mechanisms specifically. The adaptive kernel width gave no measurable benefit, the adaptive class costs gave a small but consistent gain (0.025 F1, ten of ten seeds), and the budget cost 0.17 F1 while reducing the model from about 8,950 to 320 support vectors, from 3.7 MB to 0.26 MB and the median latency by a factor of roughly 2.6 to 3.3. The gradient attributions satisfied their completeness identity to within 2.1×10^{-5} and placed signature features of each attack in the top three with a precision of 0.63 to 0.83, against chance levels of 0.13 to 0.25. The study therefore quantifies what a bounded, explainable kernel detector gains and what it costs, and it provides an open pipeline for repeating the evaluation on public datasets.

Keywords: Internet of Things; anomaly detection; intrusion detection; support vector machine; explainable artificial intelligence; concept drift; edge computing

1. Introduction

The Internet of Things (IoT) connects sensors, cameras, actuators and controllers to each other and to cloud services in homes, factories, hospitals and cities. Because these devices are numerous, heterogeneous, often poorly maintained and frequently shipped with weak credentials, they enlarge the attack surface of the networks they join, and they have been exploited for large denial-of-service campaigns, scanning, credential attacks and data theft [8,9]. Anomaly and intrusion detection systems are consequently a standard layer of defence, and a large literature applies machine learning to flow-level or packet-level features in order to separate malicious from benign behaviour [2,3,4].

Three practical obstacles still separate many published detectors from systems that can be trusted in a real deployment. The first is that traffic is neither balanced nor stationary. Attack flows are a

small fraction of the total, so a classifier trained with a single misclassification cost is pulled towards the benign majority, a well-known difficulty of learning from imbalanced data [24]. In addition, as devices are added, firmware is updated and usage patterns change, the distribution of benign traffic moves away from the one seen at training time, which degrades a model that is never updated [15]. The second obstacle is that the natural place to run a detector is close to the devices, on a gateway or an embedded board, where memory, computing power and energy are scarce and where forwarding all traffic to a remote server is undesirable for reasons of latency, bandwidth and privacy [7]. Large ensembles and deep networks are difficult to fit under such limits, and many studies report detection accuracy without reporting what the model costs to run. The third obstacle is explanation. An alert that cannot be justified is hard to triage, and security teams and regulators increasingly expect to see which observed features led to a decision [32,33]. Post hoc explainers such as LIME and SHAP address this need [30,31], but they add substantial computation at inference time, which is a poor fit for constrained devices.

There is also a methodological concern. Several authors have pointed out that results in machine-learning-based security are frequently inflated by evaluation practices that do not reflect deployment, such as random splits that place near-duplicate flows in both training and test sets, features that identify the capture environment, and the absence of any temporal ordering [5,6]. A credible study in this area therefore has to be chronological, leakage-aware, explicit about the cost of the model, and honest about where a proposed method does not help.

Support vector machines are an attractive starting point for this setting. Their decision function depends only on a subset of the training data, the support vectors, they rest on a solid theoretical foundation [1], and they perform well on the medium-dimensional tabular features that are typical of network flows. The standard formulation, however, fixes the kernel width and the misclassification costs at training time, allows the number of support vectors to grow with the data, and provides no account of which features matter for an individual prediction. Each of these properties works against the IoT setting: a fixed kernel cannot follow drift, fixed costs cannot respond to a changing class balance, an unbounded support set cannot be guaranteed to fit in memory, and an opaque score cannot support triage.

This paper examines a single framework, which we call the Adaptive Explainable Support Vector Machine (AE-SVM), that addresses these shortcomings together. The three quantities that make a kernel machine rigid, namely the kernel bandwidth, the class costs and the support set, are each controlled online at small additional cost, and the same bounded support set that makes inference cheap also makes an exact closed-form explanation possible. The contribution is not a new learning paradigm but a carefully engineered combination together with an evaluation that is designed to show where each part helps and where it does not. Specifically, the paper contributes four things. First, it specifies AE-SVM: a drift-aware bandwidth controller with a change test, a recall-driven rule for class costs, a hard memory budget enforced by age-discounted weighted prototype compression in the spirit of budgeted and core-set kernel learning [25,26,45], and integrated-gradient attributions [29] computed in closed form for the RBF decision function with

a completeness identity that can be verified numerically for every explained flow. Second, it provides a documented simulated IoT traffic benchmark with gradual and abrupt drift, a changing class balance and an attack family that appears mid-stream, together with a chronological evaluation protocol with delayed labels. Third, it reports a broad and paired evaluation with ten seeds, fourteen baseline configurations, an ablation of each mechanism, a sweep over the memory budget, measurements of model size and latency, and a quantitative check of the explanations against known attack signatures. Fourth, it reports negative and mixed findings plainly: the adaptive kernel width gave no measurable benefit on this benchmark, the compressed model is less accurate than periodically retrained supervised baselines, and the drift robustness of AE-SVM is shared by any method that is updated with labelled data.

The remainder of the paper is organised as follows. Section 2 states the problem formally, poses three research questions and recalls the necessary background on kernel machines. Section 3 presents the AE-SVM method in detail, including its complexity and default settings. Section 4 reviews related work and states the gap that the paper addresses. Section 5 describes the benchmark, protocols, baselines, metrics and statistical procedures of the experiments. Section 6 reports the results. Section 7 sets out the scope and limitations of the study, and Section 8 concludes.

2. Statement of the Problem

The problem addressed in this paper is the detection of malicious or anomalous behaviour in the network traffic of IoT deployments when the detector must satisfy four demands simultaneously. It must remain accurate when the traffic distribution drifts over time. It must remain sensitive to rare attack classes when the class balance changes. It must fit inside a fixed budget of memory and computation, so that it can run on a gateway or an embedded board. And it must give the analyst a feature-level reason for every alert. No single widely used detector satisfies all four demands, and the study asks three research questions. The first asks whether adapting the kernel width and the class costs online improves detection under chronological drift compared with static and periodically retrained detectors. The second asks what a hard bound on the number of support vectors costs in detection quality and what it saves in model size and latency. The third asks whether exact gradient attributions can explain the decisions of the detector at a cost low enough for constrained devices, and whether they agree with the known mechanics of the attacks. We state them as research questions and not as hypotheses to be confirmed, because the answer to each may be negative.

2.1 Streaming detection with delayed labels

Let the network traffic be summarised as a sequence of flows, each described by a feature vector $x \in \mathbb{R}^d$ with label $y \in \{0,1\}$, where zero denotes benign and one denotes anomalous or attack traffic. The flows arrive grouped into consecutive windows $W_0, W_1, \ldots, W_{T-1}$ of comparable size. When window W_t arrives, the detector must score and classify its flows using only information from windows $W_0, \ldots, W_{t-1}$, and the true labels of W_t become available only after its flows have

been scored, which models the delay with which analysts confirm incidents. The goal is to maximise detection quality on every window, with particular attention to the worst window, subject to constraints on the memory and on the per-flow computation of the model.

2.2 Kernel SVM with RBF kernel

A soft-margin SVM with an RBF kernel uses the similarity $K(x, x') = \exp(-\gamma \parallel x - x' \parallel^2)$. After training, the decision function is

$$f(x) = \sum_{i \in S} \alpha_i \, y_i K(x_i, x) + b$$

where S is the set of support vectors, α_i are the dual coefficients and b is the bias, and a flow is declared anomalous when $f(x) > 0$. Two hyperparameters govern the behaviour of the machine. The penalty C controls how heavily margin violations are punished, and in the class-weighted form each class c receives its own penalty $C_c = C\, w_c$. The kernel parameter γ controls how local the similarity is: a large γ yields a boundary that follows individual points and a small γ yields a smooth boundary. The number of support vectors is determined by the data and the hyperparameters and usually grows roughly in proportion to the size of the training set, which is why a standard SVM cannot be assumed to fit a fixed memory budget, and why the cost of classifying one flow, which is proportional to $|S| \cdot d$, cannot be bounded in advance.

3. Methods

3.1 Overview

AE-SVM processes the stream window by window. For every incoming window it first scores the flows with the current model and, when requested, produces explanations. Once the labels of the window are available it carries out four operations in a fixed order. It updates the kernel width, it updates the class costs, it refits the SVM on the union of the retained memory and the new window, and it compresses the memory if the budget is exceeded. The scaling of the features is fitted once, on the first window, and then kept fixed, so that a shift in the data is visible to the model as a shift in the scaled space and is not silently absorbed by re-standardisation. The design is modular, so each adaptive mechanism can be switched off independently, which makes the ablation study of Section 5 possible.

3.2 Drift-aware kernel width

The kernel width sets the scale at which two flows are considered similar, and the appropriate scale depends on how spread out the traffic currently is. For each new window the target width is obtained from the median heuristic [28], computed on a random subsample of at most 300 flows so that the cost does not depend on the window size,

$$\hat{\gamma}_t = \frac{1}{2\,\mathrm{med}_t^2}$$

where med_t is the median of the positive pairwise Euclidean distances among the sampled flows in the scaled space. To prevent the sampling noise of one window from making the boundary oscillate, the width that is actually used is an exponentially smoothed value

$$\gamma_t = (1 - \eta_t)\,\gamma_{t-1} + \eta_t\,\hat{\gamma}_t$$

where η_t is a smoothing rate whose base value is η. Adaptation should be fast when the data change abruptly, so the framework monitors the mean absolute decision score of each window with a two-sided Page-Hinkley test [18]. The test accumulates deviations of the monitored statistic from its running mean, less a tolerance δ, in both directions, and it signals a change when the accumulated deviation exceeds its running minimum by more than a threshold λ_{PH}. When a change is signalled, η_t is raised to a larger value η^+ for that update and the test is reset. The score magnitude is a convenient monitored statistic because it needs no labels, so the alarm can be raised before the labels of the window arrive.

3.3 Class-adaptive misclassification costs

Because attacks are rare, an SVM trained with one cost for all samples is pulled towards the benign class. Fixed weights inversely proportional to class frequency help, but they cannot follow a class balance that itself changes over time, and synthetic oversampling adds memory and computation that edge devices cannot spare. AE-SVM gives each class c its own weight w_c, so that the penalty of that class is $C_0 w_c$. The weights are initialised from the first window as $w_c = N/(2N_c)$, where N is the number of flows and N_c the number of flows of class c. After each later window they are corrected according to how well each class was recognised. If r_c is the recall of class c observed on the window before it is used for training and r^* is a target recall, the weight is updated multiplicatively and clipped to a fixed interval

$$w_c \leftarrow \min\left(w_{\max}, \max\left(w_{\min}, w_c\, e^{\lambda(r^* - r_c)}\right)\right)$$

where λ is a learning rate. A class whose recall is below the target receives a larger penalty in the next refit, and a class recognised better than required is relaxed. The update uses only predictions that were already produced for the window, so it needs no additional training pass.

3.4 Budgeted memory by weighted prototype compression

An edge deployment needs a guarantee on memory and on per-flow computation, and for a kernel machine both are proportional to the number of support vectors. AE-SVM therefore imposes a budget B on the number of support vectors. The detector keeps a memory of weighted prototypes. At each window the SVM is fitted on the memory together with the new window, with each memory prototype carrying a weight that stands for the mass of flows it represents. If the fitted model has no more than B support vectors, its support vectors are retained as they are. If it has more than B, the memory and the new window are compressed. Each class receives a quota of $B/2$ prototypes, and any unused part of one class's quota is given to the other class, so

that the rare attack class is not crowded out by the benign majority. Within each class the points are clustered with mini-batch k-means [47] into as many clusters as the quota allows. Each cluster is replaced by one prototype located at the weighted mean of its members, with weight

$$\omega_k = \sum_{i \in \mathcal{C}_k} \omega_i$$

and with age equal to the weighted mean age of its members. New flows have weight one, and at the start of every window the weights of the retained prototypes are multiplied by $e^{-\kappa}$, so that older evidence fades and is the first to lose influence. The SVM is then refitted on the prototypes, using sample weights equal to the prototype weights divided by their median, raised to a power p and clipped to a fixed interval. Because the support vectors of this final model are a subset of at most B prototypes, the number of support vectors never exceeds B. This is a weighted reduced-set or core-set construction in the spirit of earlier work on core vector machines and simplified decision rules [45,46], and it is combined here with age discounting and class quotas. The cost of classifying one flow is of order $B\,d$ kernel evaluations, which can be chosen in advance to match the target device, and the memory footprint is bounded by $B\,d$ stored feature values together with B dual coefficients. The budget is an explicit knob that trades accuracy for resource use, and Section 6 measures that trade.

An earlier version of the budget kept the B support vectors with the largest age-discounted dual coefficients and refitted on them alone. On development seeds it performed much worse, because the largest coefficients belong to margin violators, so that the bulk of benign traffic was forgotten, and it was replaced by the compression described here. This change was made on development seeds that are disjoint from the evaluation seeds (Section 5.8).

3.5 Intrinsic gradient-based explanations

For the RBF kernel the decision function is differentiable, and its derivative with respect to feature j has the closed form

$$\frac{\partial f}{\partial x_j}(x) = \sum_{i \in S} \alpha_i \, y_i \, K(x_i, x) \; 2\gamma \left(x_{i,j} - x_j \right)$$

A local derivative describes sensitivity at a single point and does not account for the whole score, so AE-SVM follows the integrated-gradients construction [29]. A reference point x^0 is chosen as the centroid of benign flows of the most recent window in the scaled space, and the derivative is integrated along the straight line from the reference to the flow x, giving the attribution of feature j as

$$\phi_j(x) = \left(x_j - x_j^0 \right) \int_0^1 \frac{\partial f}{\partial x_j} \left(x^0 + s\,(x - x^0) \right) ds$$

The integral is approximated with the midpoint rule using m equally spaced steps. The exact attributions satisfy the completeness identity

$$\sum_j \phi_j(x) = f(x) - f(x^0)$$

and the midpoint approximation converges to it as m grows. The completeness residual can therefore be computed for every explained flow and used as a built-in check of the implementation and of the quadrature accuracy. Because the explanation uses only the stored support vectors, its cost is about m times that of one prediction, there is no surrogate model and no random perturbation, and the explanation of a given input is deterministic. The meaning of an attribution is always relative to the chosen reference, which we discuss as a limitation in Section 7.

3.6 Procedure in words

The detector is initialised on the first window by fitting the feature scaler, estimating the kernel width with the median heuristic, setting the class weights from the class frequencies, training the SVM and applying the budget. For each later window the detector scores the flows and produces alerts and, on request, attributions. Once the labels become available it computes the mean absolute score of the window and passes it to the change test, updates the kernel width with the appropriate smoothing rate, updates the class weights from the observed per-class recall, fades the memory weights, refits the SVM on the memory together with the new window, and compresses the memory if the fitted model has more than B support vectors.

3.7 Computational complexity

Scoring one flow costs $O(B\,d)$ operations and one explanation costs $O(m\,B\,d)$. The cost of one update is dominated by the SVM refit on at most B prototypes plus the flows of the new window, and by the clustering of the same points, and it is independent of the total length of the stream, because only the memory and the new window are used. The memory used by the model at inference time is $O(B\,d)$. These bounds are structural and hold for any dataset, in contrast to a standard SVM whose support set grows with the training data. They do not by themselves guarantee a particular accuracy, which is why the empirical evaluation of Section 5 is essential.

3.8 Default settings

The default settings used throughout the experiments are listed here so that the study can be reproduced. The base penalty is $C_0 = 1$. The base smoothing rate is $\eta = 0.3$ and the boosted rate after a detected change is $\eta^+ = 0.8$. The class-weight learning rate is $\lambda = 0.5$, the target recall is $r^* = 0.95$, and the class weights are clipped to the interval from 0.1 to 50. The age discount is $\kappa = 0.3$. The Page-Hinkley tolerance is $\delta = 0.005$ and its threshold is $\lambda_{PH} = 0.5$. The median heuristic uses a subsample of at most 300 flows. The prototype weight exponent is $p = 1$ and the sample weights are clipped to the interval from 0.1 to 20. The explanations use $m = 64$

integration steps. The budget is $B = 1000$ in the main comparison and is varied from 100 to 2000 in the budget study. Apart from the retention rule, the weight exponent and the default budget, which were chosen on development seeds, these values were fixed in advance and were not tuned.

4. Related Work

4.1 Machine-learning intrusion and anomaly detection

Anomaly detection has been surveyed extensively [2], and the specific problem of network intrusion detection has been reviewed from the viewpoint of techniques, systems and challenges [3,4]. Early learning-based detectors relied on classical classifiers such as decision trees, nearest neighbours and SVMs, and they were followed by ensembles, with Random Forests [21] and gradient-boosted trees [22] remaining very strong on tabular network features, and later by deep architectures including autoencoders, convolutional and recurrent networks and transformers. For IoT in particular, autoencoder-based detectors trained on benign traffic have been proposed for botnet detection [8] and as lightweight ensembles for online use [9]. Unsupervised and one-class methods, such as Isolation Forest [19] and the One-Class SVM [20], are attractive when attack labels are unavailable, but they usually underperform supervised models when labelled attacks exist, a point that our experiments quantify.

4.2 Datasets and evaluation practice in learning-based security

Several public datasets are widely used for IoT and network intrusion detection, including UNSW-NB15 [10], ToN-IoT [11], Edge-IIoTset [12], CIC-IoT-2023 [13] and the earlier CIC-IDS collection [14]. Sommer and Paxson argued early that anomaly detection in networks faces a semantic gap between statistical novelty and maliciousness and that evaluation on unrepresentative data can be misleading [5]. Arp and colleagues later documented recurring pitfalls in machine-learning security research, including sampling bias, label inaccuracy, data snooping, spurious correlations and the use of unrealistic base rates and temporal settings [6]. These observations motivate several design decisions in our protocol, namely the use of chronological windows with delayed labels, the fitting of all preprocessing on training data only, and the reporting of imbalance-robust measures such as the precision-recall area and the Matthews correlation coefficient [41,42]. They also motivate a controlled benchmark in which the drift, the class balance and the presence of an unseen attack are known by construction, which public captures cannot provide, at the price that a simulation can never replace evaluation on real traffic.

4.3 Resource-constrained and edge-oriented detection

Edge computing moves computation towards the sources of data to reduce latency and bandwidth use and to improve privacy [7]. In the security setting, researchers have reduced the footprint of detectors through feature selection, network pruning and compression [36], and they have avoided centralising sensitive traffic through federated learning [37]. These techniques are

usually applied to neural networks, and the evidence of edge suitability is frequently limited to parameter counts or to measurements on a desktop machine. Measurements of latency, memory and energy on the devices where a detector would actually run remain comparatively uncommon.

4.4 Concept drift

Concept drift in streaming data has been surveyed thoroughly [15]. Change detectors such as the adaptive windowing algorithm [16] and the Page-Hinkley test [18] signal a change in a monitored statistic, and incremental learners such as the Hoeffding tree [17] update themselves as new labelled data arrive. In network security, drift arises both from benign changes and from evolving attacks. Most drift-aware methods treat adaptation as the only objective: they do not bound the size of the model, and they do not explain individual decisions. Our use of the Page-Hinkley test is deliberately modest. It does not replace the learner, it only modulates how fast the kernel bandwidth follows the data.

4.5 Kernel machines on a budget and incremental SVMs

The growth of the support set is a long-recognised limitation of kernel methods. Budgeted online kernel algorithms such as the Forgetron [25] and budgeted stochastic gradient descent for SVM training [26] bound the number of support vectors by discarding or merging them, incremental and decremental SVM learning allows exact updates when a sample is added or removed [27], and reduced-set methods and core vector machines approximate a kernel machine with a small weighted set of points [45,46]. Cost-sensitive SVMs adjust the penalty per class to address imbalance [23]. AE-SVM draws on these ideas but combines them in a particular way: the budget is enforced by class-balanced weighted prototypes whose weights fade with age, the class costs are updated by feedback from the detector's own recall on each window, and the kernel bandwidth is re-estimated with a robust heuristic [28]. The empirical question that this paper asks is not whether such a combination is novel in each part but what it gains and costs in an IoT detection setting.

4.6 Explainability for intrusion detection

Explainable artificial intelligence has been reviewed broadly [32,33]. Model-agnostic explainers such as LIME [31] and SHAP [30] are widely used in security applications, including intrusion detection with tree ensembles and neural networks [34,35]. Their cost at inference time is high because they require many perturbed evaluations, and their outputs can differ between runs or between very similar inputs. Gradient-based attribution methods such as integrated gradients [29] were developed for neural networks and are justified by the axioms of sensitivity and implementation invariance together with the completeness property, but these properties hold for any differentiable score, including a kernel decision function whose gradient is available analytically. Intrinsic explanations for kernel machines have received comparatively little attention in the intrusion-detection literature.

4.7 Summary of the gap

The literature provides methods for drift, for model compression and for explanation, but we are not aware of a study that evaluates, within one SVM-based IoT detector, online adaptation of the kernel width and class costs, a hard bound on the model size and exact intrinsic explanations, with paired statistics, ablations and a sweep over the memory budget. This is the gap that the present study addresses, and it addresses it with an evaluation designed to be able to return negative answers.

5. Experiment

5.1 Simulated IoT traffic benchmark

The reported experiments use a documented simulator of flow-level IoT traffic, because the public captures named in the literature could not be obtained in the environment where the experiments were run. The simulator is an instrument for studying how detectors behave under controlled drift, class imbalance and an unseen attack. It is not a substitute for real traffic, and absolute scores on it must not be read as expected performance on a real network. For each seed the simulator generates a stream of twelve windows of 2,500 flows, which is 30,000 flows, described by 24 features that follow the usual flow statistics: duration, packet and byte counts in each direction, packet and byte rates, packet-length and inter-arrival statistics, counts of the main TCP flags, destination-port entropy and number of distinct ports, flows per source, payload ratio, time-to-live statistics, the share of TCP and the window size. Feature values are generated on a log-like scale, so that they are approximately Gaussian, as is common after a logarithmic transformation.

Benign traffic comes from four device classes whose centres are drawn from a normal distribution and whose spreads are drawn uniformly between 0.7 and 1.1. The initial device mix is 0.4, 0.3, 0.2 and 0.1. Benign behaviour drifts gradually, because every centre moves by 0.10 per window along a fixed random direction, which represents firmware and usage change. From window 6 onwards an abrupt shift occurs: the two last device classes move by 1.2 along a second random direction and the device mix moves half-way towards 0.1, 0.2, 0.3 and 0.4, which represents the onboarding of new devices. Three per cent of benign flows are hard cases that resemble bulk transfers, with raised forward bytes and duration. Attack flows make up between 3% and 15% of each window, with the ratio drawn uniformly per window. There are five attack families, each defined by a sparse additive signature over three to six named features, expressed in units of the feature spread: a flood with high packet rate, SYN count and byte rate; a port scan with many distinct ports, high port entropy, many resets and short duration; a brute-force attack with many flows per source, many push flags and long duration; a man-in-the-middle attack with unusual variability of time-to-live and of inter-arrival time; and an exfiltration attack with large forward volume, long duration and high payload ratio. The intensity of each family is multiplied in every window by a factor drawn uniformly between 0.85 and 1.15, which represents attackers that adapt, and attack flows have a spread that is 15% larger than benign flows. The exfiltration

family appears only from window 7 onwards, so that it is unseen at the time of its first appearance.

5.2 Evaluation protocols

The main protocol is the drift protocol. Window 0 is used for initial training, and every method is evaluated on windows 1 to 11. A method is evaluated on window t after having learned only from the labelled windows before t, and the labels of window t are released only after it has been evaluated, which reproduces the delay of Section 2.1. Within each window the flows are shuffled with a seeded generator, which removes any dependence on the order inside a window but keeps the order of the windows. For the analysis of the shift, windows 1 to 5 are called pre-shift, window 6 is the transition, and windows 7 to 11 are post-shift. The stationary protocol uses stratified five-fold cross-validation over the whole stream of a seed, with the windows mixed and with at most 8,000 training flows per fold. It ignores time, so that it provides a conventional point of comparison with much of the literature but is known to be optimistic [6], and it is reported separately and is not used to support claims about deployment.

5.3 Methods compared

AE-SVM is run with the settings of Section 3.8 and a budget of 1,000 support vectors. Four ablation variants are run in the same loop: without the adaptive kernel width, without the adaptive class costs, without the budget, and without all three, which is a plain weighted SVM that is refitted on all retained support vectors and the new window. The baselines cover the main families. Two are SVMs, a standard RBF SVM and a class-weighted RBF SVM, both with standardised inputs and the default kernel scaling of the library [43,44]. Two are tree-based ensembles, a Random Forest of 100 trees with balanced class weights [21] and the histogram-based gradient-boosted tree classifier of scikit-learn, which belongs to the same family as XGBoost [22] but is not the same implementation. One is a multilayer perceptron with two hidden layers of 64 and 32 units trained for up to 200 iterations. The remaining two are anomaly detectors trained on at most 3,000 benign flows, an Isolation Forest of 100 trees [19] and a One-Class SVM with $\nu = 0.05$ [20]. Every baseline is evaluated in two forms. In the static form it is trained once on window 0 and never updated. In the retrained form it is retrained after every window on the two most recent labelled windows, which is up to 5,000 flows, so that the baselines have a realistic opportunity to adapt and the comparison does not contrast an adaptive method with a model that is never allowed to learn. All baselines use fixed settings and none of them was tuned, on any seed. This gives 14 baseline configurations and 5 AE-SVM configurations. No autoencoder, no Hoeffding tree with a change detector and no post hoc explainer was run, and we list this as a limitation.

5.4 Metrics

Detection quality is reported for the attack class with precision, recall and F1, the Matthews correlation coefficient [41], the false-alarm rate defined as the fraction of benign flows that are

flagged, and the areas under the precision-recall curve and the ROC curve, with emphasis on the precision-recall area because the ROC area can be misleading under strong imbalance [42]. For the drift protocol the mean over windows 1 to 11 and the F1 of the worst window are reported, and recall is also reported separately for each attack family. Edge suitability is reported with the serialised model size in kilobytes, the median and 95th-percentile latency of single-flow inference, and the number of support vectors. Explanation quality is reported with the mean absolute completeness residual, the time needed to explain a flow, and the precision with which the three most important features coincide with the signature features that the simulator added to the attack.

5.5 Statistical analysis

All stochastic procedures are seeded, and the study uses ten independent seeds, 200 to 209, each of which generates a new stream with new device centres, drift directions, attack intensities and attack ratios. Metrics are first averaged over the windows of one seed, and results are reported as the mean and standard deviation over the ten seeds. Differences between AE-SVM and each baseline are tested with the exact two-sided paired Wilcoxon signed-rank test [40] on the ten seed-level means, and the resulting p-values are corrected for multiple comparisons with the Holm procedure [39], in line with recommendations for comparing classifiers [38]. With ten pairs the smallest attainable two-sided p-value is 0.00195, so the smallest attainable Holm-adjusted value over the 14 baseline configurations is 0.0273. The ablation uses four comparisons, with a smallest adjusted value of 0.0078. The split into pre-shift and post-shift windows and the analysis per attack family were decided after the aggregate results had been seen, and they are exploratory.

5.6 Measurement of size and latency

Size and latency were measured on the cloud workstation on which the experiments were run, which has two virtual CPU cores and 8 GB of memory, in a separate single-seed run (seed 200) with no other job running, because the main runs shared the processor. The latency of a model is the time of one call of its prediction function on a single flow, measured with a high-resolution timer over 200 flows and three repetitions, and it includes the overhead of the library call, which is a large part of the cost of the smallest models. Size is the number of bytes of the serialised model. The latencies of the budget sweep come from the same quiet run with 100 flows and one repetition, and they are therefore noisier. Peak memory and energy were not measured, and no Raspberry Pi 4 or Jetson Nano was available, so we make no claim about those devices.

5.7 Explanation evaluation

For every seed and every attack family, the 60 first flows of that family in the final window were explained with the reference defined in Section 3.5 and $m = 64$ steps. The agreement with the ground truth is the fraction of the three features with the largest mean absolute attribution that belong to the signature of the family. The chance level of this quantity is the number of signature

features divided by 24. The completeness residual was computed on 50 flows. These checks establish whether the explanation reflects what the model learned about an attack that is known by construction. They do not establish that an analyst would find the explanation useful, which would require a user study.

5.8 Development procedure and reproducibility

The method was developed in two steps that must be disclosed. A first complete run on one seed (seed 0) showed that the original budget rule, which kept the support vectors with the largest dual coefficients, destroyed accuracy. The cause was then diagnosed on two development seeds (100 and 101), on which the retention rule was replaced by prototype compression, the prototype weight exponent was set to 1 and the default budget to 1,000. No other setting was tuned. The final evaluation then used seeds 200 to 209, which played no role in any of these decisions. The baselines were not tuned, so the choices made on development seeds may favour AE-SVM slightly, and this is the direction opposite to the result reported below. The implementation consists of a module with the AE-SVM class, the change test and the explanation routine, the simulator, a driver script that runs all protocols, variants and baselines and writes the result files, and an analysis script that merges them and produces every table and figure of this paper. The final runs used scikit-learn 1.9.1, and two processes ran seeds 200 to 204 and 205 to 209 in parallel. The code and the exact configuration will be deposited in a public repository at [repository address].

6. Results

6.1 Detection under drift

Table 1 gives the mean over windows 1 to 11 for every configuration, and Figure 1 shows the F1 of the main methods per window. AE-SVM obtained a mean F1 of 0.475 ± 0.032, with a precision of 0.338, a recall of 0.900 and a false-alarm rate of 0.208. It ranked 13th of the 19 configurations and exceeded five of the 14 baseline configurations: the static and retrained Isolation Forest, the static One-Class SVM, and the static Random Forest and class-weighted SVM, the last two by margins that are not statistically meaningful. The four best configurations were retrained supervised baselines, with F1 values between 0.716 and 0.729, and AE-SVM was well below them. Its precision-recall area of 0.772 was higher than that of every static supervised baseline (0.634 to 0.714) and every unsupervised detector, but lower than that of every retrained supervised baseline (0.791 to 0.858), which indicates that the ranking of flows by the AE-SVM score is competitive with static models and that part of its F1 shortfall comes from its operating point, with high recall and many false alarms, and not only from the ordering of the scores. The worst-window F1 of AE-SVM was 0.244 ± 0.077, against 0.421 ± 0.120 for the retrained SVM and 0.177 ± 0.061 for the static SVM.

Table 1. Detection performance under the drift protocol on the simulated benchmark (windows 1 to 11; mean ± standard deviation over ten seeds, sorted by F1).

Method	F1	Precision	Recall	FAR	MCC	PR-AUC	Worst-window F1
SVM[retrain]	0.729 ± 0.028	0.891 ± 0.016	0.653 ± 0.040	0.022 ± 0.010	0.727 ± 0.022	0.858 ± 0.016	0.421 ± 0.120
MLP[retrain]	0.727 ± 0.022	0.772 ± 0.041	0.707 ± 0.037	0.024 ± 0.008	0.708 ± 0.019	0.811 ± 0.023	0.550 ± 0.124
SVM-weighted[retrain]	0.719 ± 0.024	0.680 ± 0.058	0.813 ± 0.037	0.057 ± 0.012	0.700 ± 0.022	0.842 ± 0.017	0.366 ± 0.113
HistGB[retrain]	0.716 ± 0.032	0.897 ± 0.030	0.612 ± 0.044	0.009 ± 0.005	0.714 ± 0.023	0.836 ± 0.021	0.533 ± 0.126
AE-SVM(-all)	0.657 ± 0.049	0.563 ± 0.075	0.856 ± 0.027	0.089 ± 0.033	0.641 ± 0.047	0.835 ± 0.017	0.351 ± 0.115
RandomForest[retrain]	0.650 ± 0.044	0.884 ± 0.042	0.537 ± 0.057	0.008 ± 0.004	0.657 ± 0.031	0.791 ± 0.024	0.440 ± 0.135
AE-SVM(-budget)	0.645 ± 0.039	0.541 ± 0.055	0.865 ± 0.019	0.095 ± 0.027	0.630 ± 0.038	0.833 ± 0.019	0.343 ± 0.111
OneClassSVM[retrain]	0.557 ± 0.023	0.466 ± 0.033	0.751 ± 0.034	0.111 ± 0.005	0.525 ± 0.017	0.679 ± 0.033	0.266 ± 0.091
HistGB[static]	0.540 ± 0.130	0.753 ± 0.163	0.492 ± 0.173	0.033 ± 0.049	0.542 ± 0.095	0.675 ± 0.041	0.333 ± 0.127
MLP[static]	0.518 ± 0.084	0.619 ± 0.227	0.594 ± 0.185	0.085 ± 0.080	0.508 ± 0.069	0.652 ± 0.060	0.267 ± 0.103
SVM[static]	0.489 ± 0.070	0.560 ± 0.152	0.708 ± 0.196	0.206 ± 0.100	0.479 ± 0.047	0.714 ± 0.047	0.177 ± 0.061
AE-SVM(-gamma)	0.477 ± 0.029	0.341 ± 0.026	0.902 ± 0.012	0.207 ± 0.034	0.471 ± 0.027	0.778 ± 0.018	0.246 ± 0.077
AE-SVM	0.475 ± 0.032	0.338 ± 0.029	0.900 ± 0.013	0.208 ± 0.037	0.468 ± 0.030	0.772 ± 0.023	0.244 ± 0.077
SVM-weighted[static]	0.471 ± 0.056	0.425 ± 0.144	0.824 ± 0.142	0.301 ± 0.120	0.442 ± 0.058	0.685 ± 0.034	0.169 ± 0.110
RandomForest[static]	0.460 ± 0.170	0.763 ± 0.163	0.399 ± 0.205	0.022 ± 0.033	0.477 ± 0.134	0.634 ± 0.039	0.271 ± 0.134
AE-SVM(-cost)	0.450 ± 0.042	0.314 ± 0.039	0.911 ± 0.019	0.234 ± 0.049	0.445 ± 0.039	0.767 ± 0.021	0.231 ± 0.072
IsolationForest[retrain]	0.400 ± 0.048	0.533 ± 0.094	0.367 ± 0.053	0.059 ± 0.020	0.371 ± 0.056	0.451 ± 0.063	0.231 ± 0.084
OneClassSVM[static]	0.339 ± 0.029	0.237 ± 0.027	0.888 ± 0.027	0.501 ± 0.028	0.284 ± 0.024	0.490 ± 0.058	0.110 ± 0.036
IsolationForest[static]	0.314 ± 0.038	0.289 ± 0.064	0.611 ± 0.058	0.315 ± 0.057	0.252 ± 0.050	0.307 ± 0.077	0.129 ± 0.043

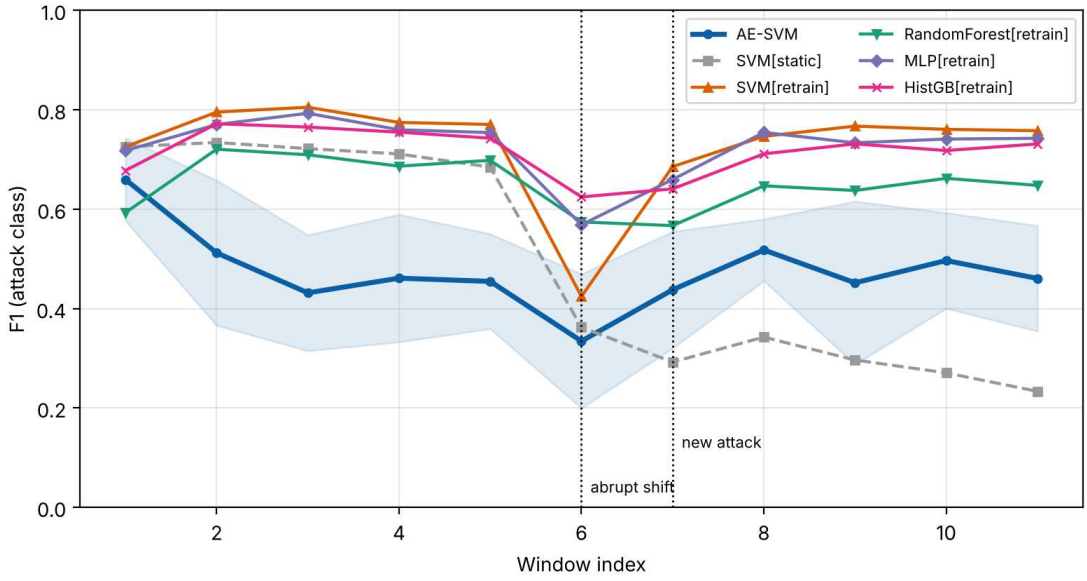


Figure 1. F1 of the attack class on each window (mean over ten seeds; the band shows one standard deviation of AE-SVM). The dotted lines mark the abrupt shift at window 6 and the first appearance of the exfiltration attack at window 7.

6.2 Effect of the abrupt shift

The abrupt shift at window 6 is the clearest feature of the stream, and Table 2 compares the F1 before and after it. All static supervised baselines degraded strongly. The static SVM fell from 0.716 to 0.287, the static class-weighted SVM from 0.704 to 0.267, the static MLP from 0.679 to 0.361, the static gradient-boosted trees from 0.654 to 0.423 and the static Random Forest from 0.572 to 0.346, which are losses between 0.226 and 0.437. AE-SVM fell from 0.504 to 0.473, a

loss of 0.031. The retrained baselines lost almost the same amount, between 0.031 and 0.049, and the variants of AE-SVM without the budget lost 0.030 and 0.027. This pattern means that robustness to the shift is obtained by any method that is updated with labelled data, and it cannot be attributed to the specific adaptive mechanisms of AE-SVM. It also means that AE-SVM operates at a lower level than the retrained baselines both before and after the shift. Compared with the static supervised baselines after the shift, AE-SVM was higher by 0.186 than the static SVM (higher in all ten seeds, $p = 0.002$ before and 0.010 after Holm correction over five comparisons), by 0.206 than the static class-weighted SVM (nine seeds, adjusted $p = 0.016$), by 0.112 than the static MLP (eight seeds, adjusted $p = 0.059$), by 0.127 than the static Random Forest (eight seeds, adjusted $p = 0.059$) and by 0.050 than the static gradient-boosted trees (four seeds, adjusted $p = 0.70$). These post-shift comparisons are exploratory.

Table 2. F1 before the shift (windows 1 to 5) and after it (windows 7 to 11), mean ± standard deviation over ten seeds.

Method	Pre-shift F1	Post-shift F1	Change
AE-SVM	0.504 ± 0.046	0.473 ± 0.045	-0.031
AE-SVM(-budget)	0.688 ± 0.047	0.658 ± 0.043	-0.030
AE-SVM(-all)	0.699 ± 0.049	0.672 ± 0.059	-0.027
SVM[static]	0.716 ± 0.122	0.287 ± 0.064	-0.428
SVM-weighted[static]	0.704 ± 0.057	0.267 ± 0.097	-0.437
RandomForest[static]	0.572 ± 0.194	0.346 ± 0.148	-0.226
HistGB[static]	0.654 ± 0.138	0.423 ± 0.135	-0.231
MLP[static]	0.679 ± 0.063	0.361 ± 0.106	-0.318
SVM[retrain]	0.774 ± 0.049	0.744 ± 0.033	-0.031
SVM-weighted[retrain]	0.771 ± 0.026	0.737 ± 0.041	-0.034
RandomForest[retrain]	0.682 ± 0.082	0.632 ± 0.060	-0.049
HistGB[retrain]	0.743 ± 0.044	0.707 ± 0.042	-0.036
MLP[retrain]	0.759 ± 0.029	0.726 ± 0.034	-0.033

6.3 Paired comparison with the baselines

Table 3 gives the paired comparison of AE-SVM with each baseline configuration on the seed-level mean F1 over all windows. AE-SVM was significantly better than the static Isolation Forest and the static One-Class SVM (adjusted $p = 0.027$, higher in ten of ten seeds), and it was not significantly different from the static supervised baselines, with differences between −0.065 and +0.014 and adjusted p-values above 0.4. It was significantly worse than all six retrained supervised or one-class baselines, with differences between −0.083 and −0.254, lower in ten of ten seeds and with an adjusted p of 0.027, which is the smallest value that ten seeds permit. The difference to the retrained Isolation Forest (+0.074, nine of ten seeds) was not significant after correction (adjusted $p = 0.059$).

Table 3. Paired Wilcoxon tests of AE-SVM against each baseline configuration on the seed-level mean F1 (ten seeds; Holm correction over 14 comparisons).

Baseline	Mean F1 difference (AE-SVM − baseline)	Seeds where AE-SVM higher (of 10)	p	p (Holm)
IsolationForest[static]	+0.160	10	0.0020	0.0273
OneClassSVM[static]	+0.135	10	0.0020	0.0273
IsolationForest[retrain]	+0.074	9	0.0098	0.0586
RandomForest[static]	+0.014	3	0.8457	0.9844
SVM-weighted[static]	+0.004	7	0.4922	0.9844
SVM[static]	-0.014	3	0.2754	0.8262
MLP[static]	-0.044	3	0.1055	0.4219
HistGB[static]	-0.065	1	0.0840	0.4199
OneClassSVM[retrain]	-0.083	0	0.0020	0.0273
RandomForest[retrain]	-0.175	0	0.0020	0.0273
HistGB[retrain]	-0.241	0	0.0020	0.0273
SVM-weighted[retrain]	-0.244	0	0.0020	0.0273
MLP[retrain]	-0.252	0	0.0020	0.0273
SVM[retrain]	-0.254	0	0.0020	0.0273

6.4 Ablation

Table 4 reports the five variants of AE-SVM. Removing the adaptive kernel width changed the mean F1 by +0.003 (0.477 against 0.475), higher without the mechanism in seven of the ten seeds and not significant ($p = 0.275$), so the adaptive kernel width gave no measurable benefit on this benchmark. The Page-Hinkley test, which accelerates this adaptation, signalled a change zero times in the first seed of the first process and once in the first seed of the second process, the only two seeds for which the counts were logged, so the boost was rarely used, and the result is consistent with the smoothed median heuristic following the gradual drift without help, although we did not test this directly. Removing the adaptive class costs lowered the mean F1 by 0.025 (0.450 against 0.475), and the full model was better in ten of ten seeds (adjusted $p = 0.0078$). The variant without adaptive costs also had a higher false-alarm rate (0.234 against 0.208) and a lower precision (0.314 against 0.338), so the cost update helped by making the detector less trigger-happy and not by increasing recall. Removing the budget raised the mean F1 by 0.171 (0.645 against 0.475) and removing all three components by 0.182 (0.657), in all ten seeds, and the false-alarm rate fell from 0.208 to 0.095 and 0.089. The budget is therefore the dominant factor in the accuracy of AE-SVM, and the combination of the adaptive mechanisms did not compensate for its cost.

Table 4. Ablation of AE-SVM under the drift protocol (mean ± standard deviation over ten seeds; the worst-window F1 is the mean over seeds).

Variant	F1	Recall	FAR	Worst-window F1
AE-SVM	0.475 ± 0.032	0.900 ± 0.013	0.208 ± 0.037	0.244
AE-SVM(-gamma)	0.477 ± 0.029	0.902 ± 0.012	0.207 ± 0.034	0.246
AE-SVM(-cost)	0.450 ± 0.042	0.911 ± 0.019	0.234 ± 0.049	0.231
AE-SVM(-budget)	0.645 ± 0.039	0.865 ± 0.019	0.095 ± 0.027	0.343
AE-SVM(-all)	0.657 ± 0.049	0.856 ± 0.027	0.089 ± 0.033	0.351

6.5 Attack families and the unseen attack

Table 5 gives the recall for each attack family. AE-SVM had the highest recall in every family among the methods shown, for example 0.847 for the man-in-the-middle attack, which is the hardest family for all methods (0.327 for the retrained SVM, 0.218 for the retrained Random Forest and 0.350 for the retrained gradient-boosted trees). This should be read together with its false-alarm rate of 0.208: it reflects an operating point that favours recall, and it does not show that AE-SVM separates the families better.

Table 5. Mean recall per attack family over windows 1 to 11 and ten seeds (the exfiltration family is averaged over the windows in which it exists).

Method	BruteForce	DDoS	MITM	PortScan	Exfiltration
AE-SVM	0.914	0.938	0.847	0.952	0.761
SVM[static]	0.717	0.825	0.500	0.856	0.618
SVM[retrain]	0.660	0.824	0.327	0.864	0.479
SVM-weighted[retrain]	0.826	0.904	0.645	0.923	0.694
RandomForest[retrain]	0.546	0.715	0.218	0.748	0.329
HistGB[retrain]	0.628	0.762	0.350	0.789	0.400
MLP[retrain]	0.727	0.838	0.479	0.857	0.533
IsolationForest[retrain]	0.336	0.454	0.197	0.497	0.324
OneClassSVM[retrain]	0.735	0.867	0.516	0.895	0.709

The exfiltration family, which first appears in window 7, tests behaviour on an unseen attack (Table 6). In window 7 no supervised model has seen it. The recall of AE-SVM was 0.398 ± 0.190, the highest recall was that of the retrained One-Class SVM (0.730 ± 0.135), which models normal behaviour only, the static SVM reached 0.529 ± 0.218, and the retrained tree ensembles and the MLP almost never detected the attack (0.007 for the Random Forest, 0.030 for the gradient-boosted trees, 0.067 for the MLP). In window 8, after its labels had been released, AE-SVM reached 0.780, the highest value among the methods shown, against 0.719 for the retrained class-weighted SVM and 0.697 for the retrained One-Class SVM. The first-window results confirm that discriminative models trained on known attacks tend to miss a new family, and that a detector with a high-recall operating point misses it less often, at the cost of false alarms.

Table 6. Recall for the exfiltration family in the window where it first appears and in the following window (mean ± standard deviation over ten seeds for the first window).

Method	Window 7 (first appearance)	Window 8
AE-SVM	0.398 ± 0.190	0.780
SVM[static]	0.529 ± 0.218	0.544
SVM[retrain]	0.116 ± 0.082	0.426
SVM-weighted[retrain]	0.286 ± 0.140	0.719
RandomForest[retrain]	0.007 ± 0.013	0.192
HistGB[retrain]	0.030 ± 0.024	0.306
MLP[retrain]	0.067 ± 0.054	0.543
IsolationForest[retrain]	0.322 ± 0.127	0.302
OneClassSVM[retrain]	0.730 ± 0.135	0.697

6.6 Memory budget, model size and latency

Table 7 and Figure 2 show the effect of the budget. With the budget raised from 100 to 500 support vectors the mean F1 hardly changed (0.425, 0.426 and 0.434), at 1,000 it was 0.475, at 2,000 it was 0.563, and without a budget it was 0.645. The model without a budget kept on average 8,953 support vectors and occupied 3,746 KB, whereas with a budget of 1,000 it kept 321 support vectors and occupied 265 KB, which is 28 times fewer support vectors and 14 times less storage, and with a budget of 100 it kept 60 support vectors and occupied 37 KB. The median latency of a single flow fell from 0.851 ms without a budget to 0.260 ms with a budget of 1,000 and to 0.223 ms with a budget of 100. The latencies of the smallest budgets are dominated by the fixed overhead of the library call, so the floor of about 0.22 ms should not be attributed to the kernel computation. No bounded setting in the sweep reached the F1 of the unbounded model, and even the largest budget, with 1,487 support vectors, lost 0.082, so on this benchmark the memory bound has a real accuracy price.

Table 7. Effect of the support-vector budget (accuracy over ten seeds; size and latency from the quiet single-seed run).

Budget B	Mean F1	Worst-window F1	Support vectors	Size (KB)	Median latency (ms)	P95 latency (ms)
100	0.425 ± 0.045	0.228	60	37	0.223	0.298
250	0.426 ± 0.039	0.235	106	77	0.226	0.309
500	0.434 ± 0.037	0.231	165	139	0.280	0.401
1000	0.475 ± 0.032	0.244	321	265	0.260	0.481
2000	0.563 ± 0.045	0.268	1487	716	0.348	0.578
unbounded	0.645 ± 0.039	0.343	8953	3746	0.851	1.210

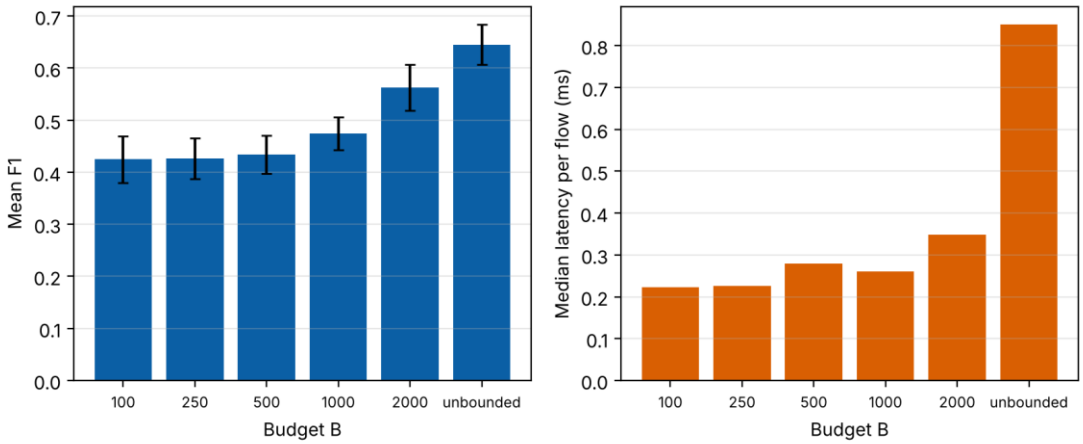


Figure 2. Mean F1 (left, error bars show one standard deviation over ten seeds) and median single-flow latency (right) for different budgets.

Table 8 compares AE-SVM with the static baselines in size and latency, measured as described in Section 5.6. The AE-SVM model with a budget of 1,000 occupied 258 KB and had a median latency of 0.320 ms (95th percentile 0.406 ms), whereas the variant without a budget occupied 3,851 KB with 0.821 ms, which is 15 times more storage and 2.6 times the latency. The Random Forest occupied 2,363 KB, which is 9.2 times more than AE-SVM, and needed 23.4 ms, which is 73 times more. The gradient-boosted trees needed 1.5 times the storage and 2.4 times the latency. The advantage over the other small models was nil: the static SVMs, the MLP and the One-Class SVM were smaller or equally fast (107 KB and 0.296 ms for the static SVM, 127 KB and 0.250 ms for the MLP), because they were trained on a single window of 2,500 flows and therefore have small support sets or few parameters. The saving of AE-SVM is therefore relative to an unbounded kernel model and to tree ensembles, and not relative to every small model.

Table 8. Serialised size and single-flow latency of the models trained on the final window (quiet single-seed run, workstation CPU, single thread).

Model	Size (KB)	Median latency (ms/flow)	P95 latency (ms/flow)	Support vectors
AE-SVM	257.7	0.320	0.406	314
AE-SVM(-budget)	3851.1	0.821	1.048	9206
SVM[static]	107.5	0.296	0.631	
SVM-weighted[static]	154.3	0.259	0.416	
RandomForest[static]	2363.4	23.413	34.631	
HistGB[static]	398.1	0.780	0.956	
MLP[static]	126.8	0.250	0.336	
IsolationForest[static]	1207.1	5.347	7.206	
OneClassSVM[static]	32.2	0.189	0.329	

6.7 Stationary evaluation

Under stratified cross-validation, which ignores time, the contrast with the drift results is informative (Table 9). The SVM, the class-weighted SVM and the MLP reached F1 values of 0.782 ± 0.018, 0.777 ± 0.017 and 0.775 ± 0.020, whereas AE-SVM reached 0.416 ± 0.036, with a recall of 0.938, a false-alarm rate of 0.266 and a precision-recall area of 0.797. Its precision-recall area was above that of the One-Class SVM and the Isolation Forest and below that of all supervised baselines. In this protocol there is no drift to blame, so the result isolates the cost of compressing a training set of up to 8,000 flows into at most 1,000 prototypes. It is a substantial cost on this benchmark.

Table 9. Stationary five-fold results (F1 mean ± pooled standard deviation over ten seeds and five folds; other columns are means).

Method	F1	Recall	FAR	PR-AUC
SVM	0.782 ± 0.018	0.658	0.003	0.892
SVM-weighted	0.777 ± 0.017	0.838	0.033	0.881
MLP	0.775 ± 0.020	0.746	0.018	0.850
HistGB	0.754 ± 0.025	0.631	0.004	0.865
RandomForest	0.692 ± 0.035	0.557	0.005	0.822
OneClassSVM	0.600 ± 0.023	0.689	0.062	0.693
AE-SVM	0.416 ± 0.036	0.938	0.266	0.797
IsolationForest	0.369 ± 0.045	0.289	0.028	0.419

6.8 Explanations

The gradient attributions satisfied the completeness identity with a mean absolute residual of 2.1×10^{-5} on decision scores of the order of one, using 64 integration steps, and one explanation took 1.93 ms per flow on the workstation with a model of 314 support vectors, which is about six times the cost of one prediction in this implementation. Table 10 compares the top three attributed features with the signature features that the simulator added. The precision was 0.633 for brute force, 0.800 for the flood, 0.733 for exfiltration, 0.800 for the man-in-the-middle attack and 0.833 for the port scan, against chance levels between 0.125 and 0.250, so the attributions recover the mechanics of the attacks well above chance in every family. The case studies of the first seed show the same picture. The five most important features were, with an asterisk marking signature features, flows per source* and push count* for brute force; SYN count*, packet rate* and byte rate* for the flood; forward bytes* and byte rate* for exfiltration; time-to-live variability* and inter-arrival variability* for the man-in-the-middle attack; and reset count*, flows per source*, port entropy* and distinct ports* for the port scan. One feature, the variability of time-to-live, appeared among the five most important features of every family although it belongs only to the signature of the man-in-the-middle attack, and it ranked first for exfiltration and for the port scan. We did not investigate the cause, and it may reflect the sensitivity of the model to a feature that moved with the abrupt shift or the dependence of the attribution on the benign reference.

Table 10. Agreement of the three most important attributed features with the signature features of each attack (mean ± standard deviation over ten seeds).

Attack family	Signature features	Top-3 precision	Chance level
BruteForce	5	0.633 ± 0.105	0.208
DDoS	6	0.800 ± 0.172	0.250
Exfiltration	5	0.733 ± 0.211	0.208
MITM	3	0.800 ± 0.172	0.125
PortScan	6	0.833 ± 0.176	0.250

6.9 Answers to the research questions

The first question asked whether adapting the kernel width and the class costs improves detection under drift compared with static and retrained detectors. The answer on this benchmark is only partly positive. AE-SVM was far less affected by the abrupt shift than static supervised models and significantly better than the static SVM after it, but the retrained baselines were equally robust and clearly more accurate. Of the two adaptive mechanisms, the kernel width had no measurable effect and the class costs had a small consistent benefit of 0.025 F1. The second question asked what the memory bound costs and saves. It saves a factor of about 28 in support vectors, 14 in storage and 2.6 to 3.3 in latency relative to the unbounded variant, and about 9 in storage and 73 in latency relative to a Random Forest, and it costs 0.17 in F1 under drift and a larger loss under stationary conditions. The third question asked whether exact attributions can explain decisions at low cost and consistently with the attack mechanics. The answer is yes on this benchmark, with a verified completeness identity, a cost of about 1.9 ms per flow and a top-three agreement with the known signatures of 0.63 to 0.83.

7. Scope and Limitations

The most important limitation is that all results come from a simulator. The simulator makes the drift, the class balance and the unseen attack known by construction, which is what allows the paired and ablation analyses, but its features are Gaussian with sparse additive attack signatures, its drift is linear and its abrupt shift has a single direction, so conclusions about the relative ranking of methods may not carry over to real traffic, where features are heavy-tailed, discrete and correlated, and where attacks overlap with benign behaviour in other ways. The method, in particular, was adjusted on development streams from the same generator, so its design may be tuned to the properties of the simulator even though the evaluation seeds are disjoint. The study should be repeated on public datasets such as UNSW-NB15, ToN-IoT, Edge-IIoTset and CIC-IoT-2023 [10,11,12,13] with the same code, and no claim of this paper should be taken to hold for them until that is done.

Several comparisons are incomplete. None of the baselines was tuned, so their absolute scores are probably lower than those of tuned models, although this cannot explain the gap in favour of the retrained baselines because the method itself was partly tuned. The gradient-boosted classifier is not XGBoost, and no deep autoencoder, no Hoeffding tree with a change detector

[16,17], no federated variant and no post hoc explainer such as SHAP or LIME [30,31] was run, so the cost and the quality of the intrinsic explanations are not compared with those of the usual alternatives. The retrained baselines use the two latest windows, which is one of many reasonable choices. Size and latency were measured on a cloud workstation with a single-flow call in an interpreted library, and neither peak memory nor energy was measured, so no claim is made about Raspberry Pi 4, Jetson Nano or other edge devices, and the latency of the smallest models is dominated by call overhead.

Several properties of the method limit its use. The budget has a large accuracy cost on this benchmark, and no budget in the sweep recovered the accuracy of the unbounded model. The detector operates at a high-recall, high-false-alarm point (a false-alarm rate of 0.208 under drift and 0.266 under stationary conditions), which would overwhelm analysts in practice unless the decision threshold is recalibrated, which we did not do. The adaptive kernel width gave no benefit, and the change test fired rarely, so that the claimed value of drift-aware bandwidth control is not supported here. The adaptive mechanisms rely on labels that arrive with a delay of one window and are complete, and the sensitivity to longer delays, label noise and partial labelling was not studied. Kernel SVM training scales faster than linearly with the number of samples, and very high-rate streams may require sharding. The feature scaler is fitted on the first window and then fixed, so a feature that moves far outside its initial range can dominate the distances.

The explanations describe how the model reaches its score and not why an attack occurs, they depend on the choice of the benign reference, and one feature appeared in the explanations of all attack families for a reason that we did not establish. Their agreement with the simulator signatures shows that they reflect what the model learned about known mechanics, and it does not show that they help analysts, which requires a user study. Finally, the adversarial robustness of the detector was not evaluated. An attacker who knows that the detector adapts its kernel and its class costs to recent traffic might try to poison the adaptation, and defences against such manipulation deserve separate study. Threats to internal validity such as leakage and tuning bias were reduced by fitting all preprocessing on training data only, by using chronological windows, by disjoint development and evaluation seeds and by applying fixed settings to all baselines, and the exploratory analyses of Sections 6.2 and 6.5 should be treated as hypotheses for future work.

8. Conclusion

This paper evaluated AE-SVM, a bounded-memory kernel detector for IoT traffic that combines a drift-aware kernel width, class costs adapted from observed recall, a hard budget on support vectors enforced by weighted prototype compression, and exact gradient-based explanations. On a documented simulated benchmark with ten seeds, the evaluation gave a nuanced answer. The detector was much less sensitive to an abrupt shift than static supervised models, but periodically retrained baselines were equally robust and more accurate, so the benefit is that of updating and not of the specific adaptive mechanisms. The adaptive kernel width had no measurable effect, the

adaptive class costs helped slightly, and the memory bound reduced the model by a factor of about 28 in support vectors and 14 in storage at a cost of 0.17 in F1. The explanations satisfied their completeness identity and agreed with the known attack signatures well above chance at a cost of about 1.9 ms per flow. These findings support a restrained claim: a bounded explainable kernel detector is feasible and its costs and benefits can be quantified, but on this benchmark it does not outperform retrained supervised models in detection quality. The next steps are to repeat the evaluation on public datasets and on real edge devices, to recalibrate the decision threshold to control false alarms, to improve the compression so that the accuracy cost of the budget is reduced, and to compare the explanations with post hoc explainers and with analysts. Future work also includes federated operation across gateways [37], semi-supervised adaptation that tolerates missing labels, and robustness to adversaries who try to poison the adaptation process.

Declarations

CRediT authorship contribution statement. [Describe each author's contribution using the CRediT taxonomy.]

Declaration of competing interest. [State that the authors have no known competing financial interests or personal relationships that could have influenced the work, or describe them.]

Funding. [State the funding sources, or that there was none.]

Data availability. No third-party dataset was used for the reported results. The data are generated by the simulator that is part of the code, which is available at [repository address] together with the configuration and the seeds of every reported run.

Declaration of generative AI and AI-assisted technologies in the writing process. [State how AI tools were used in drafting the manuscript and the code, as the journal requires, and confirm that the authors reviewed the content and take full responsibility for it.]

References

[1] Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine Learning 20(3), 273–297.

[2] Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: a survey. ACM Computing Surveys 41(3), Article 15.

[3] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E., 2009. Anomaly-based network intrusion detection: techniques, systems and challenges. Computers & Security 28(1–2), 18–28.

[4] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2, Article 20.

[5] Sommer, R., Paxson, V., 2010. Outside the closed world: on using machine learning for network intrusion detection. Proceedings of the IEEE Symposium on Security and Privacy, 305–316.

[6] Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K., 2022. Dos and don'ts of machine learning in computer security. Proceedings of the 31st USENIX Security Symposium, 3971–3988.

[7] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L., 2016. Edge computing: vision and challenges. IEEE Internet of Things Journal 3(5), 637–646.

[8] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y., 2018. N-BaIoT: network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing 17(3), 12–22.

[9] Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A., 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. Proceedings of the Network and Distributed System Security Symposium (NDSS).

[10] Moustafa, N., Slay, J., 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems. Proceedings of the Military Communications and Information Systems Conference (MilCIS), 1–6.

[11] Moustafa, N., 2021. A new distributed architecture for evaluating AI-based security systems at the edge: network TON_IoT datasets. Sustainable Cities and Society 72, 102994.

[12] Ferrag, M.A., Friha, O., Hamouda, D., Maglaras, L., Janicke, H., 2022. Edge-IIoTset: a new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access 10, 40281–40306.

[13] Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., Ghorbani, A.A., 2023. CICIoT2023: a real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors 23(13), 5941.

[14] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), 108–116.

[15] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Computing Surveys 46(4), Article 44.

[16] Bifet, A., Gavaldà, R., 2007. Learning from time-changing data with adaptive windowing. Proceedings of the SIAM International Conference on Data Mining, 443–448.

[17] Domingos, P., Hulten, G., 2000. Mining high-speed data streams. Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 71–80.

[18] Page, E.S., 1954. Continuous inspection schemes. Biometrika 41(1–2), 100–115.

[19] Liu, F.T., Ting, K.M., Zhou, Z.-H., 2008. Isolation forest. Proceedings of the IEEE International Conference on Data Mining, 413–422.

[20] Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C., 2001. Estimating the support of a high-dimensional distribution. Neural Computation 13(7), 1443–1471.

[21] Breiman, L., 2001. Random forests. Machine Learning 45(1), 5–32.

[22] Chen, T., Guestrin, C., 2016. XGBoost: a scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

[23] Veropoulos, K., Campbell, C., Cristianini, N., 1999. Controlling the sensitivity of support vector machines. Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Support Vector Machines, 55–60.

[24] He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21(9), 1263–1284.

[25] Dekel, O., Shalev-Shwartz, S., Singer, Y., 2008. The Forgetron: a kernel-based perceptron on a budget. SIAM Journal on Computing 37(5), 1342–1372.

[26] Wang, Z., Crammer, K., Vucetic, S., 2012. Breaking the curse of kernelization: budgeted stochastic gradient descent for large-scale SVM training. Journal of Machine Learning Research 13, 3103–3131.

[27] Cauwenberghs, G., Poggio, T., 2000. Incremental and decremental support vector machine learning. Advances in Neural Information Processing Systems 13, 409–415.

[28] Garreau, D., Jitkrittum, W., Kanagawa, M., 2017. Large sample analysis of the median heuristic. arXiv preprint arXiv:1707.07269.

[29] Sundararajan, M., Taly, A., Yan, Q., 2017. Axiomatic attribution for deep networks. Proceedings of the 34th International Conference on Machine Learning, PMLR 70, 3319–3328.

[30] Lundberg, S.M., Lee, S.-I., 2017. A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems 30, 4765–4774.

[31] Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "Why should I trust you?": explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–1144.

[32] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F., 2020. Explainable

artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion 58, 82–115.

[33] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D., 2018. A survey of methods for explaining black box models. ACM Computing Surveys 51(5), Article 93.

[34] Marino, D.L., Wickramasinghe, C.S., Manic, M., 2018. An adversarial approach for explainable AI in intrusion detection systems. Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society (IECON), 3237–3243.

[35] Wang, M., Zheng, K., Yang, Y., Wang, X., 2020. An explainable machine learning framework for intrusion detection systems. IEEE Access 8, 73127–73141.

[36] Han, S., Pool, J., Tran, J., Dally, W.J., 2015. Learning both weights and connections for efficient neural networks. Advances in Neural Information Processing Systems 28, 1135–1143.

[37] McMahan, H.B., Moore, E., Ramage, D., Hampson, S., Agüera y Arcas, B., 2017. Communication-efficient learning of deep networks from decentralized data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR 54, 1273–1282.

[38] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

[39] Holm, S., 1979. A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6(2), 65–70.

[40] Wilcoxon, F., 1945. Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83.

[41] Matthews, B.W., 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA) – Protein Structure 405(2), 442–451.

[42] Saito, T., Rehmsmeier, M., 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE 10(3), e0118432.

[43] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[44] Chang, C.-C., Lin, C.-J., 2011. LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2(3), Article 27.

[45] Tsang, I.W., Kwok, J.T., Cheung, P.-M., 2005. Core vector machines: fast SVM training on very large data sets. Journal of Machine Learning Research 6, 363–392.

[46] Burges, C.J.C., 1996. Simplified support vector decision rules. Proceedings of the 13th International Conference on Machine Learning, 71–77.

[47] Sculley, D., 2010. Web-scale k-means clustering. Proceedings of the 19th International Conference on World Wide Web, 1177–1178.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- paperAE.zip