https://doi.org/10.70882/vrj2bj45

SCIENCE FORUM (JOURNAL OF PURE AND APPLIED SCIENCES) 26, (5 Special) (2026), 542 - 560

Science Forum (Journal of Pure and Applied Sciences)

Journal Homepage: https://atbuscienceforum.com.ng/index.php/jpas

A Resource-Aware Edge–Cloud Framework for IoT Intrusion Detection under Traffic Shift

Ahmed Adnan Hadi[a]

[a]College of Engineering, Al-Karkh University of Science, Baghdad, Iraq, Postal Code: 10080

Abstract

The gateway of an IoT network has to detect attacks with limited computation and limited uplink traffic. We built REC-IDS (Resource-aware Edge–Cloud IDS), in which a decision tree on the gateway classifies every flow and a deferral gate sends a declared share of the flows to a classifier in the cloud. First, we trained the tree and the cloud model on the training data. Second, we took a small labelled window from the beginning of the test period: its first half adapted the models in the adapted setting, and its second half trained the gate and set its threshold. Third, we tested the system on the remaining traffic. We used CIC-DDoS2019, with training and test data from different days, and CICIoT2023, with the first 70% of each class in time for training and the last 30% for testing. On CIC-DDoS2019 with a frozen gateway, the accuracy of the tree alone was 0.880. The learned gate raised it to 0.937, which is at the level of the cloud model alone (0.934), while 11% of the flows were sent and the uplink traffic fell by 89%. On CICIoT2023 the gain was smaller. At a declared budget of 10%, the accuracy went from 0.797 to 0.810 (cloud model 0.810), and after adapting the models to the labelled window it went from 0.817 to 0.835 (cloud model 0.837). Under a random split, the tree alone reached 0.9988 on CIC-DDoS2019 after the removal of conflicting labels and the cloud added nothing. On CICIoT2023 under a random split, the cloud model added one point of accuracy and the gate recovered it with 10% of the flows sent. The benefit of REC-IDS appeared when the test traffic differed from the training traffic: about 90% less uplink traffic at an accuracy close to that of the cloud model. On CICIoT2023 the learned gate was slightly better than a confidence gate.

Keywords

IoT Security;
Intrusion Detection;
Edge–Cloud Computing;
Learning to Defer;
Selective Escalation;
Distribution Shift;
CIC-DDoS2019;
CICIoT2023.

EISSN: 3156-5190 (Online)
ISSN: 1119-4618 (Print)
JPAS
Faculty of Science, ATBU Bauchi

Corresponding Author: Ahmed Adnan Hadi, College of Engineering, Al-Karkh University of Science, Baghdad, Iraq, Postal Code: 10080.
Email: a.algbory@kus.edu.iq © 2026 Faculty of Science, ATBU Bauchi. All rights reserved

1. Introduction

The security of the Internet of Things (IoT) depends on detecting malicious traffic close to the devices. Many IoT devices have limited memory and processing power and are rarely updated, so they can easily become a source of distributed denial of service (DDoS) traffic. Intrusion detection systems (IDS) based on machine learning can recognise the malicious flows, but a large model may not fit on the gateway of the network. The quality of detection has therefore to be judged together with the latency, the memory and the communication that it needs.

Cloud-assisted designs move the model selection, training and distribution to the cloud, and the device only runs the prediction step [1,2]. The Cloud Model as a Service (MaaS) framework of [1] is the closest to our work. It divides the work between a cloud service layer and an IoT device layer and evaluates five classifiers on CIC-DDoS2019 [3]. Its evaluation is, however, difficult to interpret: the data split is not described, accuracies of 0.992–0.999 appear next to an F1 of 0.00 for the rarest class in one model and not more than 0.62 in two others, and the cost is given as training time on a virtual device only. The architecture is described, but no policy is evaluated for deciding which flows the device handles alone and which go to the cloud.

This article studies that policy. A compact classifier on the gateway handles the flows by itself, and a deferral gate sends a limited share of the flows to a stronger classifier in the cloud. Our first design used the doubt of the gateway tree as the gate, which is the usual choice in early-exit and selective classification. On training and test files from different days it behaved poorly, because a flow that the tree doubts is often a flow that the cloud model also gets wrong. We therefore measured how much any gate could gain with an oracle, and found considerable headroom when the gateway model is not updated. This led to the main idea of the article: the gate is trained to predict directly whether the cloud model will correct the gateway tree on a flow. Trained on a small labelled window, this gate brings the accuracy of the gateway to that of the cloud model with about a tenth of the flows leaving the gateway on CIC-DDoS2019. We then repeat the evaluation on a second dataset, CICIoT2023, where the gain is smaller, and we compare our values with those reported by the studies that introduced the two datasets, which use random splits.

The remainder of the article is organized as follows. Section 1.1 presents the literature survey and section 1.2 the contributions. Section 2 gives the materials and methods, which are the framework, the datasets, the evaluation protocols and the evaluation measures. The results and the discussion are in section 3, and the conclusion and the future works in section 4.

1.1 Literature Survey

The literature of IoT intrusion detection has several directions, and we group it by the way in which the work is divided between the devices and the rest of the network. A first group moves the heavy work away from the device. In [1], a cloud service layer builds and distributes the models while the device layer extracts the features and predicts, and Naive Bayes, Decision Tree, KNN, Logistic Regression and a deep neural network were evaluated on a subset of CIC-DDoS2019. In [2], the collaboration of fog and edge nodes was used for a scalable IDS in smart cities, and [4] combined edge devices that train local models, fog nodes that aggregate them and a cloud layer that builds a global model, with trust management and a distributed ledger. In [5], a stacked sparse autoencoder reduces the dimension of the traffic so that edge devices can train, and federated learning

coordinates the cloud and the edge. In these designs the layers exchange models or updates, and the prediction for a single flow remains at one layer. A closer design is the two-stage one. In [6], a decision tree on the edge discards the traffic that it labels normal and forwards the traffic that it labels anomalous to a CNN-BiLSTM-Attention model in the cloud, and [7] forward the data of an industrial sensor network to the cloud only when the edge detectors find it anomalous. In both, the label of the edge model is the forwarding rule, there is no declared budget, and the rule is not trained to predict whether the cloud model would be right. None of these works sends a flow to a stronger model because a weaker one is likely to fail on it.

A second group keeps the raw data on the devices by federated learning. [8] combined federated learning with K-means sampling on NSL-KDD, [9] ran a federated IDS with Raspberry Pi clients and a Jetson server on NSL-KDD and Car-Hacking data, and [10] proposed a semi-decentralised BiLSTM model that is evaluated on CICIoT2023. In [11], the detection and the attack typing were separated in two stages on IoTID20 with SMOTE. Federated learning reduces the exchange of raw data, but every client still has to run a model that fits its own resources, and the model that the client runs is the model that decides.

A third group makes the model small enough to run on the device. [12] ran a deep IDS on a Raspberry Pi 1 with six packet features. [13] studied ensemble feature selection for lightweight IDS, and [14] targeted memory and energy for constrained devices with a lightweight machine-learning approach. Some works report measurements on real hardware. [15] reported a CPU use below 1% for anomaly detection in medical IoT, [16] used a pruned and quantised ensemble on a Raspberry Pi with Kafka and reported 97.3% accuracy with a peak CPU use of 111.8%, and [17] used CTGAN with an explainable model for resource-constrained devices. These works show what a device can run, but a model that is small enough for the device is also a weaker model, and none of them uses a stronger model for the flows on which the small one fails.

Table 1. Summary of the related work; the last column states whether the work decides per flow if a stronger model is consulted.

Study	Strategy	Reported setting	Per-flow deferral
Alsharif & Rawat [1]	Cloud MaaS, five classifiers	CIC-DDoS2019 subset	No
Shi & Lin [6]	Edge tree forwards the flows it labels anomalous to a CNN-BiLSTM-Attention cloud model	UNSW-NB15, NSL-KDD, CIC-IDS2017	By the edge label
Yang et al. [7]	Edge detectors forward the data to the cloud when it looks anomalous	Industrial sensor data	No
Hajj et al. [8]	Federated learning, K-means sampling	NSL-KDD	No
Bhavsar et al. [9]	Federated learning on Pi clients	NSL-KDD, Car-Hacking	No
Alsaleh et al. [10]	Semi-decentralised BiLSTM	CICIoT2023	No
Rahman et al. [2]	Fog and edge collaboration, scalable IDS	Smart-city IoT	No
Qaddoura et al. [11]	Two-stage detection and attack typing	IoTID20, SMOTE	No
Fatima et al. [13]	Ensemble feature selection for light IDS	Resource-constrained IoT	No
Alwaisi et al. [14]	Lightweight machine learning for constrained devices	Anomaly prevention	No
Zachos et al. [15]	Anomaly-based IDS for IoMT	CPU use below 1%	No

Study	Strategy	Reported setting	Per-flow deferral
Hasan & Tasnim [17]	CTGAN with an explainable model	Resource-constrained devices	No
Wardana et al. [4]	Edge–fog–cloud collaboration, ledger	Trust management	No
Thamilarasu & Chawla [12]	Deep IDS on Raspberry Pi 1	Six packet features	No
Musthafa et al. [16]	Pruned, quantised ensemble on Pi	Kafka, 97.3% accuracy	No

Outside the intrusion detection, the question of which samples should be handled by a small model and which by a large one has a long history. The reject option of [18] abstains when the posterior of the classifier is low, and selective classification of deep networks uses the confidence for the same purpose [19]; the study of [20] compares 18 such methods on 44 datasets and finds no single best one. In the edge–cloud setting, distributed deep neural networks classify a sample at a local exit when the normalised entropy is below a threshold and forward the others to the cloud [21], while [22] chooses the layer at which a network is split between a mobile device and a server according to the cost of each part. All these policies rely on the doubt of the small model, or on the cost, but they do not look at whether the large model would be right. Learning to defer addresses this point by training the rule that decides who predicts. [23] learn a classifier together with a deferral rule that accounts for the cost and the accuracy of the expert, and [24] give consistent estimators for it. [25] show that learning the classifier and the rejector together is hard even for linear models, and give an exact algorithm for problems of moderate size. Our gate follows this idea in the setting of an IDS, where the expert is the cloud model and the deferral budget is a communication budget.

The last group of works concerns the evaluation. The datasets of the Canadian Institute for Cybersecurity are widely used, but [26] found labelling errors and artefacts in CICIDS2017, and [27] listed the pitfalls of machine learning in computer security, among them the split of the data in time and the choice of the metric. The change of the traffic between days is an instance of concept drift, which is studied in [28], and the usual reply is to update the model with labelled recent data. Across datasets, [29] trained models on enterprise traffic of 2017 and tested them on IoT datasets of 2023–2024, and the F1 dropped by about 0.20 to 0.27, with tree ensembles degrading less than neural models. The paper that introduced CICIoT2023 [30] trains five classifiers on a blended and shuffled dataset split at random into 80% and 20%, which is the protocol that we compare with in section 3.6. These points guide our protocol. The training and the test data come from different days, results are given by class, and the benefit of the labelled window is measured separately for the gateway and for the cloud model.

The datasets, the labels, the hardware and the metrics of the studies above are different, so their numbers can be used as context but not ranked. Among the works that we reviewed, none trains a deferral gate that decides per flow whether a gateway classifier should consult a cloud classifier under a declared budget of communication, and none evaluates it with training and test data from different days. This is the gap that the article addresses.

1.2 Contributions

The contributions of the article are the following.

1. A budgeted escalation rule: the operator declares the share of flows that may leave the gateway, and the threshold follows from a quantile of the calibration scores.

2. Learned deferral gates, trained on a small labelled window to predict the flows on which the gateway tree is wrong and the cloud model is right.
3. An oracle gate that measures the room left for any gate.
4. A time-separated evaluation on CIC-DDoS2019 and CICIoT2023 that compares the learned gate with uncertainty gates and a random control.

2. Materials and Methods

This section presents the framework, the datasets and the measures. The overview and the pseudocode are in section 2.1, the edge classifier, the uncertainty scores and the budgeted escalation in section 2.2, the learned gates and the oracle in section 2.3, and the cloud classifier and the adaptation settings in section 2.4. The datasets and the evaluation protocols are in sections 2.5 to 2.7, and the parameters and the evaluation measures in sections 2.8 and 2.9. Every equation is followed by the list of its elements.

2.1 Overview of the Framework and Pseudocode

The framework is presented in figure 1. It works in two phases. In the calibration phase, a labelled window of the traffic is used to fit the models, to train the deferral gate and to calculate its threshold. The window is split in two parts: the first adapts the models, and the second, which the models never see, trains the gate and sets the threshold, so that the gate learns the errors of the models on flows that they have not learned. In the detection phase, every flow is classified on the gateway, and only the uncertain flows are sent to the cloud.

In the detection phase, the gateway pre-processes the features of a flow and passes them to the edge tree, which returns a class and its probabilities. The gate turns them into a score. The flows whose score is above the threshold are escalated: the feature vector, 312 bytes, is sent to the cloud classifier, and its label replaces the label of the tree. The other flows keep the label of the tree. The threshold is not tuned directly. It follows from the budget B, the share of flows that the operator allows to leave the gateway, so that the cost of communication is known in advance.

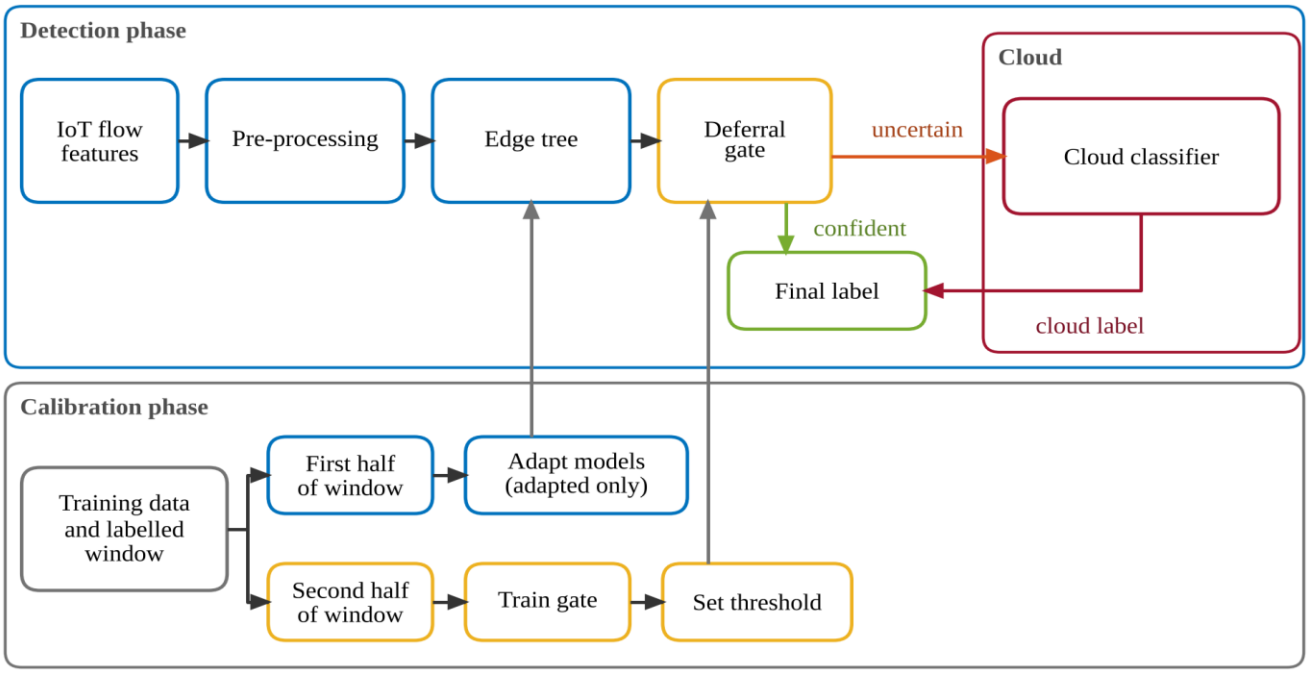


Figure 1. Framework of REC-IDS with the calibration phase and the detection phase.

The steps are given in table 2.

Table 2. Pseudocode of REC-IDS.

```
Data={xi}, i=1,2...N                                                    // Stream of flows
D                                                                       // Training data
W=Wa+Wb                                                                 // Labelled window in two halves
T, M                                                                    // Edge tree, cloud classifier
B                                                                       // Declared budget of escalation
Output
Y={yi}, i=1,2...N
Start
//Calibration phase (once per window)
    1. Fit T on D (and on Wa if the gateway is adapted)                 // Frozen or adapted gateway
    2. Fit M on D and Wa, and select M on Wb
    3. For every flow (z,y) in Wb: t=1 if T(z)!=y and M(z)=y, otherwise 0   // Target of equation 5
    4. Fit the gate g on (P,scores) of Wb to predict t                  // Four contiguous folds
    5. S=out-of-fold scores of g
    6. (tau,p)=Threshold(S,1-B)                                         // Equation 4
//Detection phase (for every flow, on the gateway)
    7. For i=1 until N
        8. z=Scale(xi)
        9. (yedge,P)=T(z)
        10. s=g(P,scores)                                               // Scores of equations 1 to 3
        11. If s>tau or (s=tau and Random()<p)
            12. yi=M(z)                                                 // Escalation to the cloud
        13. Else
            14. yi=yedge                                                // Decision at the gateway
End
```

The pseudocode has a calibration phase and a detection phase. The calibration phase runs once for each labelled window, in the cloud or on a server. The tree is fitted on the training data and also on the first half of the window when the gateway is adapted. The cloud classifier is fitted and selected, and the second half of the window gives for every flow the target t of the deferral gate, which is one when the tree is wrong and the cloud classifier is right. The gate is then fitted on contiguous folds and the threshold is calculated from the out-of-fold scores and the budget. The detection phase is the only part that runs on the gateway for each flow: every flow is classified by the tree and scored by the gate, and the flows whose score is above the threshold are sent to the cloud classifier, whose label replaces the label of the tree.

2.2 Edge Classifier, Uncertainty Scores and Budgeted Escalation

The edge classifier is a decision tree of maximum depth 12 with at least 20 samples per leaf, trained on the class-balanced training sample. For each flow it gives the class probabilities, from which four scores are calculated, where a larger value means more uncertainty: the confidence score 1 – p1, the margin score 1 – (p1 – p2), the entropy in equation 1 and the novelty in equation 2, where p1 and p2 are the largest and the second largest class probabilities.

$$s_{ent} = -\Sigma\, p_j \ln p_j / \ln K \qquad \text{-1-}$$

Where,

s_{ent} denotes the entropy score of the flow.

p_j denotes the probability that the tree gives to class j.

K denotes the number of classes.

$\ln K$ denotes the factor that scales the score to the interval from 0 to 1.

$$s_{nov} = (1/d)\, \Sigma_j \min(|z_j - m_{kj}| / s_{kj}\ , 10) \qquad \text{-2-}$$

Where,

s_{nov} denotes the novelty score of the flow.

d denotes the number of features.
z_j denotes the value of feature j after the logarithm and the standardisation.
k denotes the class predicted by the tree.
m_{kj} denotes the median of feature j over the training flows of class k.
s_{kj} denotes the robust scale of feature j in class k, which is the largest of 1.4826 times the median absolute deviation, half of the standard deviation, and 0.05.
10 denotes the cap that limits the effect of one feature.

The first three scores measure the doubt of the tree itself. The novelty measures how far the flow is from the training flows of the predicted class, so it can flag a confident prediction on traffic that the tree has not seen. The combined score averages the empirical distribution functions of the confidence and the novelty on the calibration window, as in equation 3. A random gate, which gives a uniform random number to each flow, is the control, and a gate that does not beat it at the same escalation rate has no value for the selection.

$$s_{combo} = \tfrac{1}{2} [\, F_{conf}(s_{conf}) + F_{nov}(s_{nov}) \,] \qquad -3-$$

Where,
s_{combo} denotes the combined score of the flow.
s_{conf} denotes the confidence score 1 – p1 of the flow.
s_{nov} denotes the novelty score of equation 2.
F_{conf}, F_{nov} denote the empirical distribution functions of the two scores over the flows of the calibration window, so that each term lies between 0 and 1.

For a gate and a declared budget B, a flow is escalated when its score exceeds the threshold given in equation 4.

$$\theta_B = Q(\, S_{cal} \,,\, 1 - B \,) \qquad -4-$$

Where,
θ_B denotes the threshold for the budget B.
B denotes the declared share of flows that may be sent to the cloud, with $0 \le B \le 1$.
S_{cal} denotes the set of the scores of the flows of the calibration window.
$Q(S,q)$ denotes the quantile of order q of the set S.

Scores are often tied, for example when many leaves of the tree are pure or when a gate has few distinct values, and a deterministic rule would send too many or too few flows. The flows whose score equals the threshold are therefore escalated with a probability chosen so that exactly the share B of the calibration flows is escalated. On the test day the realised escalation rate can differ from B if the distribution of the scores has shifted, and we report it next to every result. The grid of budgets is 0, 2, 5, 10, 15, 20, 30, 40, 50, 75 and 100%.

2.3 Learned Deferral Gates and the Oracle

The scores above measure how much the tree doubts a flow, but the escalation needs a different quantity: a flow is worth sending only if the cloud model classifies it correctly while the tree does not. For each labelled flow we define the target in equation 5, and the gate is trained to predict it.

$$t = 1 \,[\, \hat{y}_{edge} \neq y \;\; and \;\; \hat{y}_{cloud} = y \,] \qquad -5-$$

Where,
t denotes the target of the flow, equal to 1 if the escalation corrects the tree and 0 otherwise.
$1[\cdot]$ denotes the indicator function.
y denotes the true class of the flow.
$\hat{y}_{edge}$ denotes the class predicted by the tree.
$\hat{y}_{cloud}$ denotes the class predicted by the cloud model.

Three gates were built. The first, called ppv, gives a flow the smoothed rate of t among the labelled flows to which the tree assigned the same class, as in equation 6. It has only K parameters and can be recomputed on the gateway. The second, called learn, is a gradient boosting classifier with 120 iterations and 15 leaves on the class probabilities of the tree and its four scores. The third, called learn_raw, receives also the 78 flow features.

$$s_{ppv}(k) = (m_k + 20\, \bar{t} \,) / (n_k + 20) \qquad -6-$$

Where,
$s_{ppv}(k)$ denotes the score of a flow that the tree assigns to class k.
n_k denotes the number of labelled flows assigned to class k.
m_k denotes the number of them with t = 1.
$\bar{t}$ denotes the rate of t = 1 over all labelled flows.

20 denotes the weight, in flows, of this overall rate as a prior.

The target is known only on the labelled window, which is divided into four contiguous folds inside each class. Contiguous folds are needed because neighbouring flows of a capture are almost duplicates, and random folds would leak the labels. The scores of the window that sets the threshold are out-of-fold, and the scores of the test flows come from a model fitted on the whole window. As an upper bound, we define an oracle that gives the score 1 to the flows with $t = 1$ using the true labels. The oracle cannot be deployed. Its role is to measure the largest gain that any gate can obtain at a given budget, and so to tell whether a better gate is worth building.

2.4 Cloud Classifier and Adaptation Settings

The cloud classifier is selected for each seed between a multilayer perceptron with 256, 128 and 64 units and a histogram gradient boosting classifier with 63 leaves and 300 iterations for CIC-DDoS2019 and 31 leaves and 100 iterations for CICIoT2023 by the macro F-measure on the calibration window. We use three settings. In the first, no model is adapted, and all models are trained on the data of the training day only. In the second, the gateway is frozen: the tree is trained on the training day, and only the cloud models are refitted with the adaptation half of the labelled window, repeated three times to give it weight. This is the situation when the gateway cannot be updated. In the third, the tree is refitted with the same data as well. The other half of the window selects the cloud model and fits the thresholds and the gates, so no model is evaluated on the data that fitted it. The labelled window comes from the same day as the test flows, which means that an operator or a trusted detector labels a sample of the recent traffic.

2.5 CIC-DDoS2019

CIC-DDoS2019 [3] contains benign traffic and DDoS attacks. We used the version with separate training-day and test-day CSV files. The columns that identify the flow or the host were dropped, which leaves 78 numeric features. Infinite values became missing values, missing values were replaced by the training median, and the features were passed through a signed logarithm and standardised. A random 10% of the rows of each file was read, duplicated feature vectors were removed inside the training data, and the test rows whose feature vector equals a training row were removed without looking at the label. As a sensitivity check, the experiment was repeated after removing every row whose feature vector occurs with more than one label inside the training data or the test data, which are 3,238,178 training rows and 754,040 test rows, before any evaluation; the results are given in section 3.4. The training classes were balanced by capping the largest at 100,000 flows and raising the small ones to at least 2000.

2.6 CICIoT2023

CICIoT2023 [30] is the second dataset. Its per-class CSV files have 39 numeric features and no label column, because the label is the name of the folder, and each row summarises a window of packets and not a complete flow; we keep the word flow for brevity. Before any run, we merged the 34 folders of benign traffic and 33 attacks into eight classes, namely benign and the seven attack groups of the dataset, which are DDoS, DoS, Mirai, reconnaissance, spoofing, brute force and web-based attacks. Inside each class, the files were read in the order of capture, at most 100,000 rows per folder were kept at an even stride over the whole capture, the first 70% of the rows of each class form the training data, the last 30% the test data, and 1% of the rows between them were dropped because windows near the border overlap in time. This gives 1,711,337 training rows and 708,997 test rows before the removal of duplicated rows. The pre-processing and the class cap are those used for

CIC-DDoS2019. The split is by position in the capture, since the dataset has no separate days, so it tests the stability of the models over time inside the capture and is not a test on other devices or networks.

2.7 Evaluation Protocols

The frozen and the adapted settings of section 2.4 are evaluated on a calibration window and a test set that do not overlap. For CIC-DDoS2019 the training day and the test day are different files. For CICIoT2023 the split is by position in the capture, and its random 80/20 counterpart, used by the study that introduced it, is run separately and reported in section 3.6.

The calibration window is the first 20% of each test class, and only the remaining 80% is reported, which gives 635,199 test flows. In the family space, all reflection-based attacks are merged in one class named Reflection, which leaves four classes. This merge was decided after the first results, because most of the errors of the tree in the original space are confusions between reflection types, and the accuracy of the two spaces is not comparable. The frozen gateway is evaluated in the original space and the adapted gateway in the family space. The counts by class are given below.

In the original label space, the 635,199 final test flows are BENIGN with 4,575, DrDoS_LDAP with 9,838, DrDoS_MSSQL with 95,849, DrDoS_NetBIOS with 7,867, DrDoS_UDP with 126,278, Syn with 390,735, UDP-lag with 57; the calibration window has 158,800 flows, and in the adapted settings it is split in half between the adaptation of the models and the thresholds, the gates and the choice of the cloud model.

2.8 Parameters of the Experiments

The edge tree has depth 12 and at least 20 samples per leaf. The cloud model is a gradient boosting classifier with 63 leaves and 300 iterations for CIC-DDoS2019 and 31 leaves and 100 iterations for CICIoT2023, or an MLP with 256, 128 and 64 units, selected on the calibration window. The learned gates are gradient boosting models with 120 iterations and 15 leaves fitted with four contiguous folds, and the declared budgets are 0, 2, 5, 10, 15, 20, 30, 40, 50, 75 and 100%. The training sample is 10% of the rows of CIC-DDoS2019 or at most 100,000 rows per folder of CICIoT2023, with a class cap of 100,000 and a class floor of 2000. The calibration window is the first 20% of each test class, and half of it is added to the training data and repeated 3 times in the adapted settings. The payload per escalated flow is 312 bytes for the 78 float32 values of CIC-DDoS2019 and 156 bytes for the 39 values of CICIoT2023. The number of seeds is 10 without adaptation, 3 with adaptation and 5 for the comparison with Cloud MaaS for CIC-DDoS2019, and 3 for CICIoT2023.

2.9 Evaluation Measures

For a test set of N flows with true classes y_i and predicted classes ŷ_i, the accuracy is given in equation 7. For a class c, the precision, the recall and the F-measure are given in equations 8, 9 and 10, and their unweighted mean over the K classes, the macro F-measure, in equation 11.

$$Acc = (1/N) \Sigma_i \, 1[\, \hat{y}_i = y_i \,] \qquad -7-$$

Where,

Acc denotes the accuracy.

N denotes the number of test flows.

y_i, $\hat{y}_i$ denote the true and the predicted class of flow i.

$1[\cdot]$ denotes the indicator function.

$$P_c = TP_c / (TP_c + FP_c) \qquad -8-$$

$$R_c = TP_c / (TP_c + FN_c) \qquad -9-$$

$$F_c = 2 P_c R_c / (P_c + R_c) \qquad -10-$$

$$F_{macro} = (1/K) \Sigma_c F_c \qquad -11-$$

Where,

P_c, R_c, F_c denote the precision, the recall and the F-measure of class c.

TP_c denotes the number of flows of class c that are predicted as c.

FP_c denotes the number of flows of other classes that are predicted as c.
FN_c denotes the number of flows of class c that are predicted as another class.
F_{macro} denotes the macro F-measure.
K denotes the number of classes.

The operational measures describe the escalation. The realised escalation rate is given in equation 12, and the uplink traffic in equation 13.

$$\rho = (1/N)\ \Sigma_i\ 1[\ s_i > \theta_B\ ] \qquad \text{-12-}$$

Where,

ρ denotes the realised escalation rate.
s_i denotes the score of flow i given by the gate.
θ_B denotes the threshold of equation 4, with the random rule of section 2.2 for the flows whose score equals it.

$$U = 1000\ \rho\ b \qquad \text{-13-}$$

Where,

U denotes the uplink traffic in bytes per 1000 flows.
b denotes the payload of one escalated flow: 312 bytes for the 78 features of CIC-DDoS2019 and 156 bytes for the 39 features of CICIoT2023, stored as 32-bit numbers.
$1000\ b$ denotes the traffic when all flows are sent, so that the saving of uplink traffic is 1 – ρ.

To compare a framework with the tree and the cloud model, we use the share of the gain of the cloud model that is obtained in equation 14 and the reduction of the error with respect to the tree in equation 15. The results are means over the seeds, with the standard deviation between the seeds, with r seeds, the mean μ = Σ v_s / r and the sample standard deviation. No setting was tuned on the test flows. The accuracy is dominated by the two large attack classes, so the macro F-measure and the results by class complete it. The size of the gateway model and the inference time per flow are also reported.

$$G = (\ M_{RA} - M_{edge}\ ) / (\ M_{cloud} - M_{edge}\ ) \qquad \text{-14-}$$

Where,

G denotes the share of the gain of the cloud model that is obtained.
M_{edge}, M_{RA}, M_{cloud} denote the value of one measure, accuracy or macro F-measure, for the tree, for REC-IDS and for the cloud model.

The gain G is not defined when the cloud model and the tree have the same value of the measure, and we do not report it in that case.

$$E = 1 - (\ 1 - Acc_x\ ) / (\ 1 - Acc_{edge}\ ) \qquad \text{-15-}$$

Where,

E denotes the reduction of the error rate.
Acc_x denotes the accuracy of the compared model x.
Acc_{edge} denotes the accuracy of the tree.

3. Results and Discussion

The parameters are those of section 2.8. We first compare the edge tree and the cloud model alone and measure the room for the escalation with the oracle in section 3.1. The results for the frozen gateway, including the results by class, are given in section 3.2 and for the adapted gateway in section 3.3. After that, the sensitivity to conflicting labels is in section 3.4, the comparison with Cloud MaaS on CIC-DDoS2019 in section 3.5, the second dataset CICIoT2023 with its original study in section 3.6, and the resource measurements in section 3.7. Most results are drawn as figures, and the tables are kept for definitions and for the values printed in the cited studies.

3.1 The Edge Tree and the Cloud Model Alone, and the Room for Escalation

Without adaptation, the cloud model is not the stronger one. With the original labels, its macro F-measure is 0.4688 against 0.7428 for the edge tree, and with the family labels it is 0.6914 against 0.7305. Sending flows to such a model costs accuracy, and this is the first reason why a gate that only measures the doubt of the tree cannot do much. A labelled window changes this. With a frozen gateway, the cloud model reaches an accuracy of 0.9343 and a macro F-measure of 0.8424, against 0.8795 and 0.7376 for the tree. When the tree is adapted too, the two models are level, 0.9936 for the tree and 0.9927 for the cloud model, and the error of the tree is already below 1%.

The oracle tells how much any gate can gain. With a frozen gateway, the share of the test flows on which the tree is wrong and the cloud model is right is between 5.4% and 6.5% in the three seeds. An oracle that escalates 10.5% of the flows reaches an accuracy of 0.9382 and a macro F-measure of 0.8735, which is higher than the cloud model alone, 0.9343 and 0.8424. With an adapted gateway, the same share is only 0.25–0.29%, and the oracle at 10% improves the macro F-measure of the tree from 0.7479 to 0.7540. The escalation thus has much to gain when the gateway model is stale, and little when it is up to date.

3.2 Frozen Gateway: Learned Deferral and Results by Class

The results for the frozen gateway at a declared budget of 10% are given in figure 2a and the curves in figure 4a. The learned gates are much better than the uncertainty gates. The learned gate escalates 11.3% of the flows and reaches an accuracy of 0.9372, which is the level of the cloud model alone, 0.9343, and of the oracle, 0.9382, while the entropy gate, the best of the uncertainty gates, reaches 0.8949 and the random gate 0.8851. The errors go from 12.0% to 6.3% of the flows, which is 48% fewer, and the uplink traffic is 89% lower than in the cloud-only operation, about 35,225 bytes per 1000 flows against 312,000.

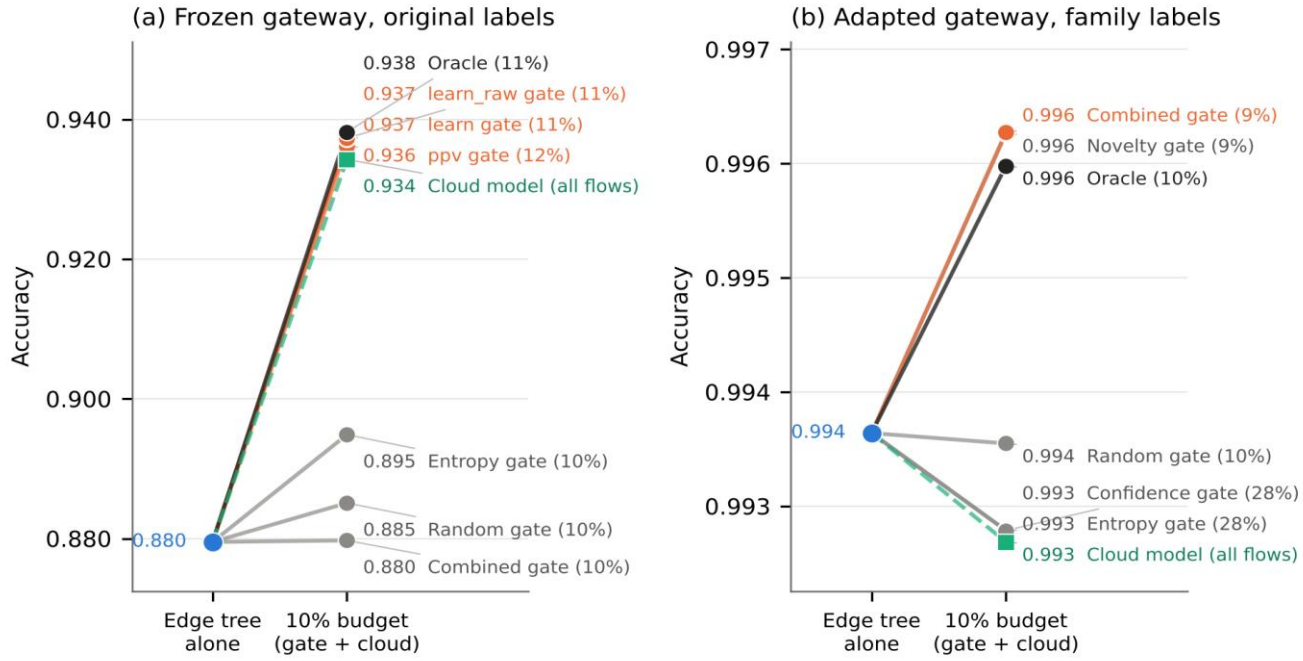


Figure 2. Accuracy of the gates on CIC-DDoS2019 at a declared budget of 10% with the share of flows sent in brackets. Panel a is the frozen gateway with original labels and panel b the adapted gateway with family labels, as means over 3 seeds.

The advantage does not depend on the exact budget in figure 4a: the learned gates reach the level of the cloud model at a declared budget of 10% and stay there up to 50%, and at 5% they already recover most of the gap, 0.7989 for learn against 0.7529 for entropy and 0.7407 for the random gate in macro F-measure. The realised escalation of the ppv gate is below the declared one at large budgets, 41% at 50%, because it has few distinct scores.

The macro F-measure shows the same order. The three learned gates reach 0.8503 for ppv, 0.8482 for learn and 0.8524 for learn_raw, which is 83%, 81% and 84% of the gain of the oracle and at least the level of the cloud model alone, 0.8424, while the entropy gate reaches 0.7819 and the combined gate 0.7396, which is the level of the random gate, 0.7437. The simple ppv gate is as good as the gradient boosting gates. It has only one number per class, so it

can run on the gateway itself. Part of the gain in the macro F-measure comes from the class UDP-lag, 57 test flows: the tree labels 13,112 Syn flows as UDP-lag, and the cloud model does it for two. The accuracy, on which this class has almost no effect, shows the same gain.

The accuracy and the macro F-measure hide the classes, so figure 3 gives the recall by class of the frozen gateway at a declared budget of 10%. Every class gains from the escalation and none is lost. The largest gains are for DrDoS_LDAP from 0.703 to 0.878, DrDoS_NetBIOS from 0.822 to 0.937 and DrDoS_UDP from 0.617 to 0.718, and the recall of Syn reaches 1.000. The recall of the learned gate is equal to or slightly higher than that of the cloud model in every class. Two weak points remain. Most flows of DrDoS_UDP are labelled DrDoS_SSDP by all the models, a confusion between reflection attacks that is the reason for the family label space of the adapted setting, and for UDP-lag 21 of the 57 flows are found by the cloud model and by the gate, while the tree finds none.

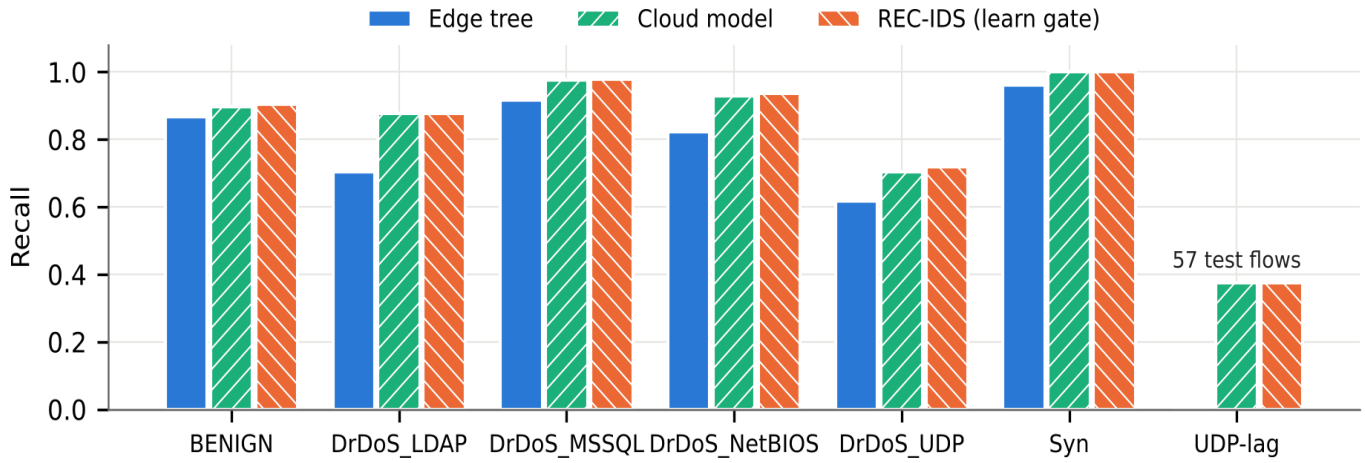


Figure 3. Recall by class on CIC-DDoS2019 at a declared budget of 10%, frozen gateway, mean over 3 seeds.

The bootstrap intervals of the test flows, seed 42, support the same order. The macro F-measure of the learned gate is 0.840 with a 95% interval of 0.827–0.853 against 0.748 with an interval of 0.747–0.750 for the tree, and the cloud model gives 0.834 with an interval of 0.822–0.848. The interval of the gate overlaps with that of the cloud model, so we claim that the gate reaches the level of the cloud model with 11% of the flows, not that it exceeds it. The random gate improves the tree by only 0.005.

3.3 Adapted Gateway: Combined Gate

When the tree is adapted as well, the room is small, and the gates are compared in figure 2b and in figure 4b. The combined gate escalates 9.2% of the flows and brings the accuracy from 0.9936 to 0.9963, which removes 41% of the errors of the tree and equals the oracle, 0.9960. The macro F-measure goes from 0.7479 to 0.7531, which is 85% of the gain of the oracle, 0.7540, and the uplink is 91% lower than in the cloud-only operation. The random gate gives no gain, 0.9935, and the novelty gate behaves like the combined gate. The confidence, margin and entropy gates escalate 28% of the flows at a nominal 10% and do not improve the tree, which shows that their thresholds, set on the calibration window, do not transfer to the test flows once the tree is refitted. The choice of the gate therefore follows the situation: the learned gate when the gateway is frozen, and the combined gate when it is updated.

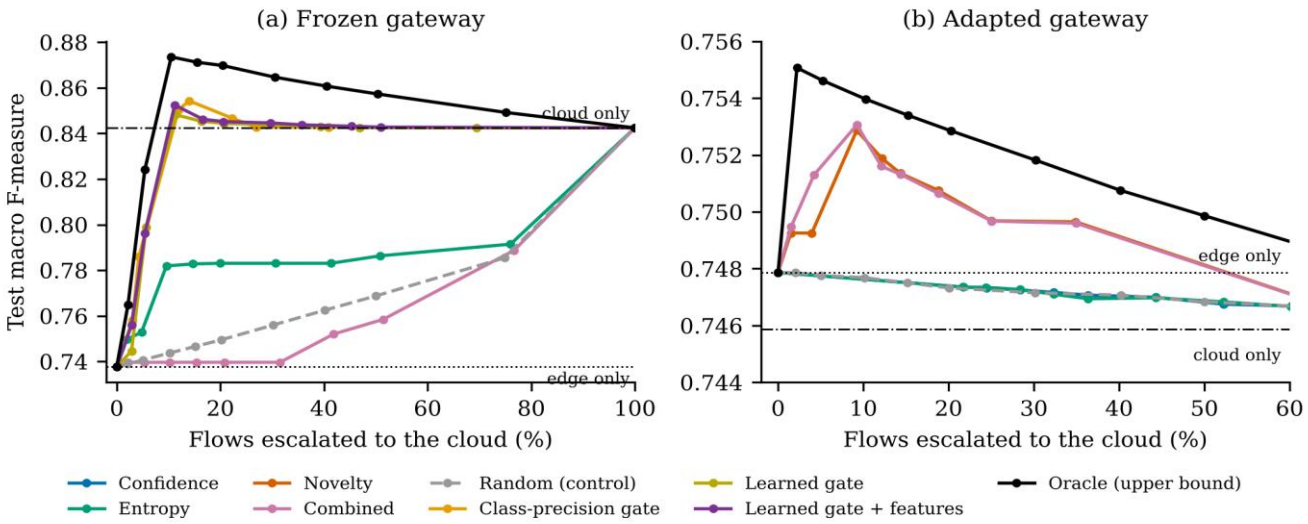


Figure 4. Macro F-measure against the share of flows escalated on CIC-DDoS2019. Panel a is the frozen gateway with original labels and panel b the adapted gateway with family labels.

3.4 Sensitivity to Conflicting Labels

Many feature vectors occur with more than one label, and the test flows of UDPLag.csv carry four different labels, so a part of the error of every model cannot be removed by any classifier. We therefore repeated the frozen-gateway experiment after removing every row whose feature vector occurs with more than one label, in the training and in the test data, which removes 3,238,178 and 754,040 rows and leaves 912,897 test flows.

The removal makes the tree weaker, with an accuracy of 0.6364, so the room for the escalation grows to about one third of the flows, that is, the tree is wrong and the cloud model is right on 33% of them. The conclusion of the previous sub-sections holds: the learned gates reach the accuracy of the oracle, 0.9642 for learn and 0.9654 for ppv against 0.9643, and exceed the cloud model alone, 0.9427, while the entropy gate, 0.7031, and the random gate, 0.7590, stay far below. What changes is the cost: the gates need about 40% of the flows, instead of 11%, and the saving of the uplink traffic is 58% instead of 89%. At a budget of 10%, the learn gate sends only 6% of the flows, because the scores of the labelled window do not transfer to the cleaned test flows.

3.5 Comparison with Cloud MaaS

The study used for the comparison is Cloud MaaS [1], which evaluated five classifiers on a subset of 725,165 records of CIC-DDoS2019 with four labels, benign, Syn, UDP and UDPLag, once with 15 features selected by a random forest and once with 4 pre-selected features. The experiments ran in Raspberry Pi. Table 3 reproduces the values printed in their article, where the macro F-measure is derived by us as the mean of the four printed class F-measures. A fifth classifier, a deep neural network, is omitted because no values by class are given for it, accuracy 0.826.

Table 3. The detection results reported by Cloud MaaS [1]; F measure order: benign / Syn / UDP / UDPLag.

Model	Accuracy	F measure, 15 features	Macro F, derived,	F measure, 4 features	Macro F, derived,
Naive Bayes	0.998	0.95 / 1.00 / 1.00 / 0.62	0.893	0.26 / 1.00 / 0.99 / 0.00	0.563
Decision Tree	0.999	1.00 / 1.00 / 1.00 / 1.00	1.000	0.95 / 1.00 / 1.00 / 0.75	0.925
KNN	0.999	0.99 / 1.00 / 1.00 / 0.59	0.895	0.94 / 1.00 / 1.00 / 0.70	0.910
Logistic Regression	0.992	0.30 / 1.00 / 0.97 / 0.00	0.568	0.32 / 1.00 / 0.99 / 0.00	0.578

The accuracy is almost saturated while the F measure of UDPLag differs strongly, 0.62, 0.59 and 0.00 for Naive Bayes, KNN and Logistic Regression with 15 features, and the Decision Tree is the only model with perfect scores on all classes. Figure 7 places the values of Cloud MaaS and of the original dataset studies next to ours.

The two evaluations differ in the split of the data, since Cloud MaaS does not describe it while ours separates the training day from the test day, in the label space, in the number of features, 15 or 4 against 78, and in the removal of duplicated records, so figure 7 shows how the model families behave when the test traffic comes from another day, and it does not rank the studies. Without adaptation, the accuracy of Naive Bayes and Logistic Regression falls to 0.383 and 0.384 from 0.998 and 0.992 in Cloud MaaS, and after the adaptation REC-IDS reaches 0.9963. The macro F-measure stays below 0.75 because of the class UDP-lag, on which our edge tree has an F measure of 0.016 against 1.00 for the Decision Tree of Cloud MaaS. The weakness of this class comes from the labels: in the 10% sample of the test file named UDPLag.csv only 158 of 72,310 rows carry the label UDP-lag, while 60,597 carry Syn, 11,129 carry UDP and 426 are benign, and many of them have the feature vector of flows with other labels in section 3.4. We therefore ran the protocol of Cloud MaaS, with pooled flows, four labels, 15 features selected by a random forest and a random split into 70% and 30%, with all rows and after the removal of the rows whose feature vector has several labels, which are 190,939 training rows and 78,721 test rows. With all rows, the edge tree has an accuracy of 0.9732 and every model finds no more than 2% of the UDP-lag flows. After the removal, only 94 of the 5,725 UDP-lag test flows remain, and the accuracy of the edge tree is 0.9988, inside the range of 0.992 to 0.999 of Cloud MaaS; its macro F-measure is 0.995 on the other three classes and 0.7843 with UDP-lag, F measure 0.153. On this protocol the cloud model and the gate give no gain: the cloud model has a macro F-measure of 0.7659 against 0.7843 for the tree, and the oracle gains 0.0003 in accuracy. The edge tree is thus sufficient when the test flows resemble the training flows, and the escalation pays when they come from another day.

The paper that introduced the dataset [3] reports weighted precision, recall and F-measure for a training day and a testing day: ID3 0.78, 0.65 and 0.69, random forest 0.77, 0.56 and 0.62, Naive Bayes 0.41, 0.11 and 0.05, and logistic regression 0.25, 0.02 and 0.04. The weighted recall equals the accuracy, which is 0.65 for ID3 against 0.8822 for our edge tree without adaptation and 0.9372 for the frozen gateway with the learned gate. The label sets differ and the paper also mentions five-fold cross validation, so we use these values as a reference and not as a ranking. Our evaluation adds the results by class, the inference time, the model sizes and the effect of a per-flow deferral.

3.6 Second Dataset: CICIoT2023

Figure 5 gives the results on CICIoT2023 for the edge tree, the cloud model and REC-IDS at a declared budget of 10%, with a frozen gateway and with an adapted one, mean and standard deviation over 3 seeds. The cloud models are a smaller configuration than for CIC-DDoS2019 in section 2.8. With a frozen gateway, the cloud model raises the accuracy of the tree from 0.797 to 0.810 and the macro F-measure from 0.635 to 0.676. REC-IDS reaches an accuracy of 0.810 and a macro F-measure of 0.662 with 10.2% of the flows sent, which is 96% of the gain in accuracy and 65% of the gain in macro F-measure of the cloud model, and the uplink traffic is 90% lower than when every flow is sent. The gain is much smaller than on

CIC-DDoS2019, because the cloud model itself is better than the tree by only 1.4 points of accuracy, and the tree is wrong while the cloud model is right on only 3.1% of the test flows.

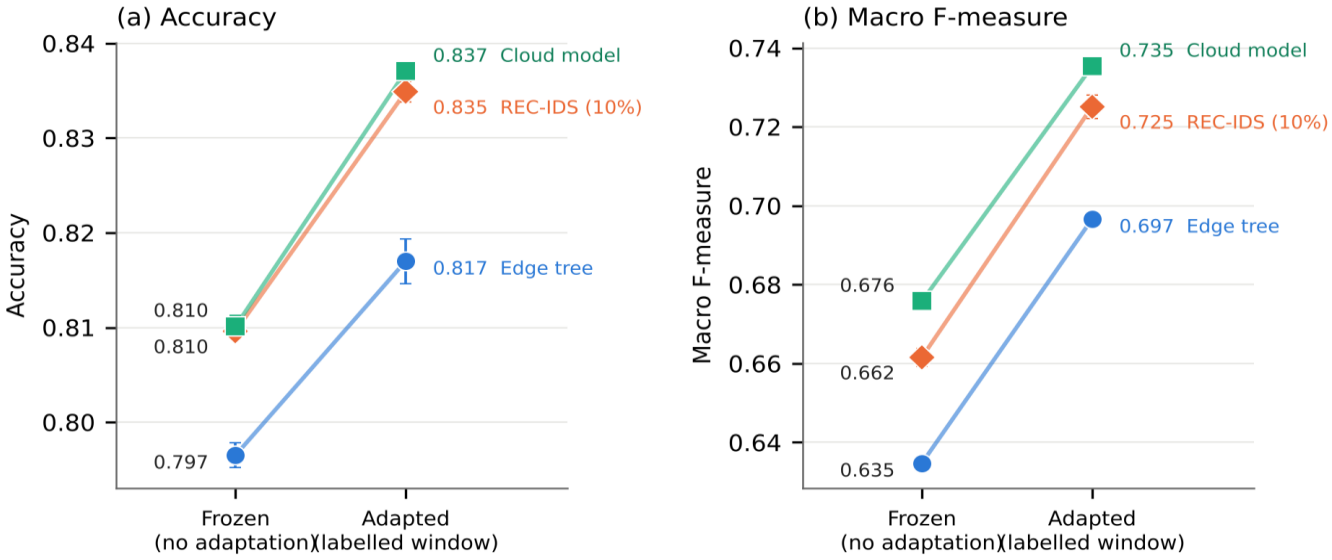


Figure 5. Accuracy and macro F-measure on CICIoT2023 at a declared budget of 10%, with a frozen and an adapted gateway, mean over 3 seeds.

The learned gate is better than the random gate, macro F-measure 0.662 against 0.639, but on this dataset it is not clearly better than the simple uncertainty gates. At the same budget, the confidence gate has an accuracy of 0.802 against 0.810 for the learned gate, and the same macro F-measure, 0.662 against 0.662. With the adapted models, the learned gate is ahead by 1.1 points of accuracy over the confidence gate and by 0.004 of macro F-measure, and figure 6 shows the macro F-measure only. The benefit that we can claim on CICIoT2023 is therefore that of the budgeted escalation, and not a large advantage of the learned target over the confidence of the tree. The oracle shows that a better gate could still gain about 1.7 points of accuracy in figure 6.

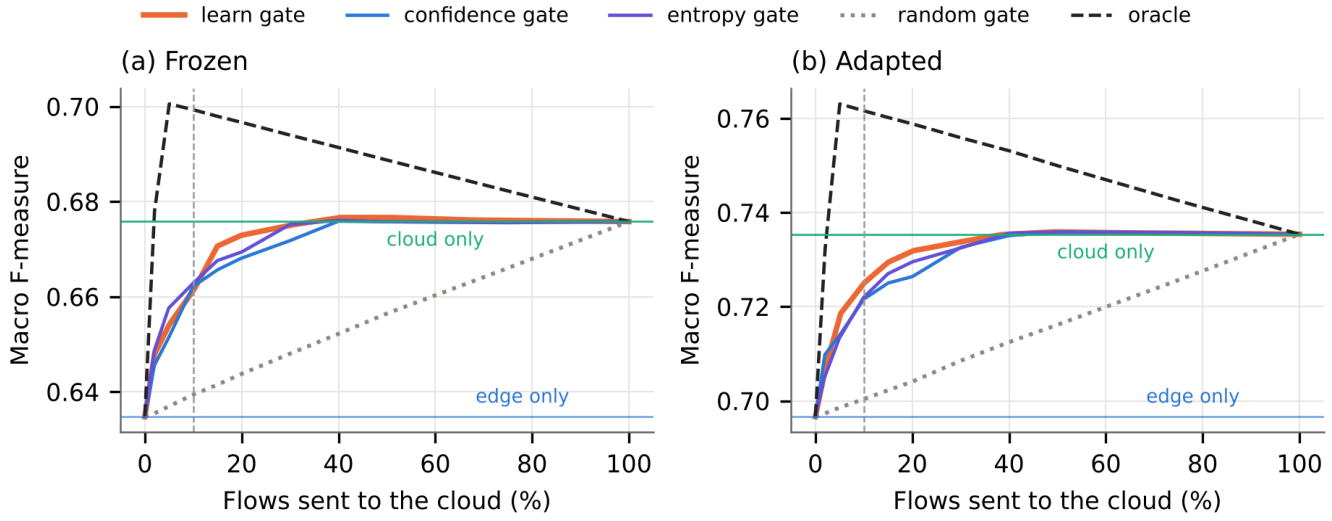


Figure 6. Macro F-measure against the share of flows sent to the cloud on CICIoT2023. Panel a is the frozen setting and panel b the adapted models. The vertical line marks the budget of 10%.

When the models are refitted with a labelled window of 57,932 rows from the first part of the test period, and the results are reported on the rest, the accuracy of the tree goes from 0.797 to 0.817, that of REC-

IDS from 0.810 to 0.835 and its macro F-measure from 0.662 to 0.725, with 10.2% of the flows sent. REC-IDS is 1.8 points above the adapted tree and 0.2 points below the adapted cloud model, 0.837. These labels come from the same capture as the test flows, so in practice an analyst or a trusted detector has to provide them, and the gain should be read as the value of that labelling effort. The escalation helps most on the weak classes: for web-based attacks, whose test set is small, 7,205 flows, the F-measure of the tree is 0.086, that of the cloud model 0.239 and that of REC-IDS 0.162 with a frozen gateway, and 0.348, 0.453 and 0.413 when adapted. Web-based and brute-force attacks remain the weak point on this dataset.

To see how many labels are needed, we repeated the adapted run with a window of half the size, 28,967 rows instead of 57,932, and one seed. The accuracy of the tree is 0.817, that of REC-IDS 0.835 with 10.4% of the flows sent, and that of the cloud model 0.832, so the gain in accuracy is about the same as with the full window. The macro F-measure is 0.690 for the tree, 0.717 for REC-IDS and 0.724 for the cloud model, which is lower for REC-IDS than with the full window, where it was 0.725 as the mean of 3 seeds, but the two values do not come from the same seeds. The 95% bootstrap intervals over the test flows are 0.690 with an interval of 0.687–0.693 for the tree, 0.717 with an interval of 0.714–0.721 for REC-IDS and 0.724 with an interval of 0.721–0.727 for the cloud model: In the macro F-measure REC-IDS is above the tree and below the cloud model, and the intervals do not overlap, although they come close at the third decimal, as the upper limit of REC-IDS is 0.7206 and the lower limit of the cloud model is 0.7211. These intervals describe the test flows of one seed and not the variation between seeds, so with half the labels the gain is confirmed, but its stability is not.

After the removal of the rows with conflicting labels, which are 141,790 of the 1,711,337 training rows and 44,381 of the 708,997 test rows, the accuracy of the tree is 0.789, that of REC-IDS 0.811 and that of the cloud model 0.810, so the ceiling of about 0.81 on this dataset is not caused by conflicting labels, in contrast with CIC-DDoS2019. It may come from the change of the traffic over the capture and from the limits of the 39 features. The CICIoT2023 study [30] shuffles the blended dataset and splits it at random into 80% and 20%. For eight classes, they report accuracies of 0.9944 for a random forest and 0.9911 for a deep neural network, and F1-scores of 0.719 and 0.697 in their Table 6. We then ran a random split of our own on CICIoT2023, with the rows drawn at random and the duplicate rows kept. The tree alone reached an accuracy of 0.841, the cloud model 0.851, and REC-IDS at a declared budget of 10% reached 0.851 with 10.2% of the flows sent, so the gate recovered the whole gain in accuracy of the cloud model. Its macro F-measure was 0.695 against 0.677 for the tree and 0.712 for the cloud model. Over 3 seeds the random gate gained only 0.003 to 0.005 of macro F-measure over the tree, against 0.016 to 0.021 for the learned gate. Our accuracy of 0.810 to 0.835 is therefore not comparable with theirs, and it is lower. Figure 7 puts the values reported by Cloud MaaS and by the CICIoT2023 study next to ours for each protocol.

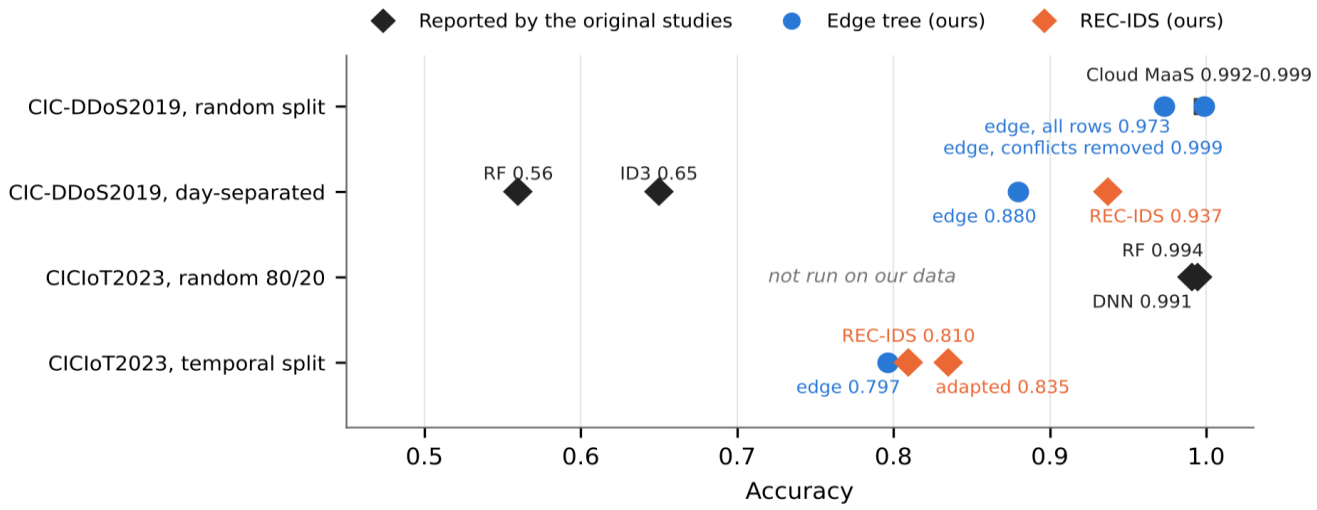


Figure 7. Accuracy under the protocols of the original studies and under our time-separated protocols. For CICIoT2023 the random split gave 0.851 for REC-IDS and for the cloud model.

3.7 Resource Measurements

The gateway needs 77 KB in the adapted setting for the edge tree, the pre-processing and the novelty statistics. The edge stage takes less than a microsecond per flow in batch mode and 0.74 ms per flow in single-flow mode, against 2.77 ms for the cloud model. In the frozen setting, the learned gate sends about 35,225 bytes per 1000 flows, and the combined gate in the adapted setting about 28,817 bytes, against 312,000 when every flow is sent. The gateway tree is about twenty times larger than the 3.2 KB tree of Cloud MaaS, because of the 78 features and the day-separated data.

Table 4. The resource measurements, as means over seeds, with timings in Python on a laptop and without network delay.

Measure	CIC-DDoS2019, no adaptation	CIC-DDoS2019, adapted	CICIoT2023, frozen
Edge tree size (KB)	254.6	70.4	161.3
Gateway total size (KB)	266.2	77.2	166.3
Edge stage, batch, ms per flow,	0.0022	0.0008	0.0008
Edge stage, single flow, median / p95, ms,	1.24 / 1.78	0.74 / 0.99	1.45 / 2.09
Cloud model, single flow, median / p95, ms,	2.51 / 3.23	2.77 / 3.87	11.82 / 20.75

4. Conclusion

This article developed REC-IDS, an edge-cloud IDS in which a decision tree on the gateway classifies every flow and a deferral gate sends a declared share of the flows to a cloud classifier. The gate is trained on a small labelled window to predict the flows on which the tree is wrong and the cloud model is right, and an oracle gate measures how much any gate can gain. On CIC-DDoS2019 with training and test data from different days, a frozen gateway with the learned gate raised the accuracy from 0.880 to 0.937, the level of the cloud model alone, 0.934, with 11% of the flows sent and 89% less uplink traffic. With an adapted gateway, the combined confidence and novelty gate removed 41% of the errors of the tree with 9% of the flows escalated. On CICIoT2023, with a split in time inside the capture, the gain is smaller: REC-IDS reached an accuracy of 0.810 against 0.797 for the tree and 0.810 for the cloud model, with 10%

of the flows sent, and 0.835 when the models were adapted to a labelled window.

The evidence supports a limited claim. When the test traffic differs from the training traffic, a budgeted escalation keeps most of the accuracy of a cloud model with about a tenth of the flows leaving the gateway, and on CIC-DDoS2019 a learned gate is clearly better than the uncertainty gates, whereas on CICIoT2023 it is better than the confidence gate by about one point of accuracy at most. On CICIoT2023 under a random split of our own, the cloud model adds one point of accuracy (0.851 against 0.841) and REC-IDS recovers all of it with a tenth of the flows sent.

Future work is to study adaptation without human labels, for example by using the labels that the cloud model returns for the escalated flows to update the gateway, repeat the label-efficiency runs with several seeds and smaller windows.

References

1. Alsharif, M., & Rawat, D. B. (2021). Study of machine learning for cloud assisted IoT security as a service. *Sensors*, *21*(4), 1034. https://doi.org/10.3390/s21041034
2. Rahman, M. A., Asyhari, A. T., Leong, L. S., Satrya, G. B., Hai Tao, M., & Zolkipli, M. F. (2020). Scalable machine learning-based intrusion detection system for IoT-enabled smart cities. *Sustainable Cities and Society*, *61*, 102324. https://doi.org/10.1016/j.scs.2020.102324
3. Sharafaldin, I., Lashkari, A. H., Hakak, S., & Ghorbani, A. A. (2019). Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE.
4. Wardana, A. A., Kołaczek, G., & Sukarno, P. (2024). Lightweight, trust-managing, and privacy-preserving collaborative intrusion detection for Internet of Things. *Applied Sciences*, *14*(10), 4109. https://doi.org/10.3390/app14104109
5. Yang, R., He, H., Xu, Y., Xin, B., Wang, Y., Qu, Y., & Zhang, W. (2023). Efficient intrusion detection toward IoT networks using cloud–edge collaboration. *Computer Networks*, *228*, 109724. https://doi.org/10.1016/j.comnet.2023.109724
6. Shi, F., & Lin, Z. (2026). Lightweight edge detection and high-precision cloud classification: A cloud-edge collaborative two-stage NIDS architecture. *Applied Sciences*, *16*(14), 7302. https://doi.org/10.3390/app16147302
7. Yang, T., Jiang, X., Li, W., Liu, P., Wang, J., Hao, W., & Yang, Q. (2025). Cloud-edge collaborative data anomaly detection in industrial sensor networks. *PLOS ONE*, *20*(6), e0324543. https://doi.org/10.1371/journal.pone.0324543
8. Hajj, S., Azar, J., Bou Abdo, J., Demerjian, J., Guyeux, C., Makhoul, A., & Ginhac, D. (2023). Cross-layer federated learning for lightweight IoT intrusion detection systems. *Sensors*, *23*(16), 7038. https://doi.org/10.3390/s23167038
9. Bhavsar, M. H., Bekele, Y. B., Roy, K., Kelly, J. C., & Limbrick, D. B. (2024). FL-IDS: Federated learning-based intrusion detection system using edge devices for transportation IoT. *IEEE Access*, *12*, 52215–52226. https://doi.org/10.1109/ACCESS.2024.3386631
10. Alsaleh, S., Menai, M. E. B., & Al-Ahmadi, S. (2025). A heterogeneity-aware semi-decentralized model for a lightweight intrusion detection system for IoT networks based on federated learning and BiLSTM. *Sensors*, *25*(4), 1039. https://doi.org/10.3390/s25041039
11. Qaddoura, R., Al-Zoubi, A. M., Faris, H., & Almomani, I. (2021). A multi-layer classification approach for intrusion detection in IoT networks based on deep learning. *Sensors*, *21*(9), 2987. https://doi.org/10.3390/s21092987
12. Thamilarasu, G., & Chawla, S. (2019). Towards deep-learning-driven intrusion detection for the Internet of Things. *Sensors*, *19*(9), 1977. https://doi.org/10.3390/s19091977
13. Fatima, M., Rehman, O., Rahman, I. M. H., Ajmal, A., & Park, S. J. (2024). Towards ensemble feature selection for lightweight intrusion detection in resource-constrained IoT devices. *Future Internet*, *16*(10), 368. https://doi.org/10.3390/fi16100368
14. Alwaisi, Z., Kumar, T., Harjula, E., & Soderi, S. (2024). Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention. *Internet of Things*, *28*, 101398. https://doi.org/10.1016/j.iot.2024.101398
15. Zachos, G., Mantas, G., Porfyrakis, K., de Bastos, J. M. C. S., & Rodriguez, J. (2025). Anomaly-based intrusion detection for IoMT networks: Design, implementation, dataset generation, and ML algorithms evaluation. *IEEE*

Access, *13*, 41994–42028. https://doi.org/10.1109/ACCESS.2025.3547572

16. Musthafa, M. B., Huda, S., Nguyen, T. T., Kodera, Y., & Nogami, Y. (2025). Optimized ensemble deep learning for real-time intrusion detection on resource-constrained Raspberry Pi devices. *IEEE Access*, *13*, 113544–113556. https://doi.org/10.1109/ACCESS.2025.3584373
17. Hasan, T., & Tasnim, S. (2025). Real-time explainable IoT security with machine learning and CTGAN-enhanced detection for resource-constrained devices. *Ad Hoc Networks*, *178*, 103937. https://doi.org/10.1016/j.adhoc.2025.103937
18. Chow, C. K. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, *16*(1), 41–46.
19. Geifman, Y., & El-Yaniv, R. (2017). Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*.
20. Pugnana, A., Perini, L., Davis, J., & Ruggieri, S. (2024). Deep neural network benchmarks for selective classification. *Journal of Data-centric Machine Learning Research*.
21. Teerapittayanon, S., McDanel, B., & Kung, H. T. (2017). Distributed deep neural networks over the cloud, the edge and end devices. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE.
22. Kang, Y., Hauswald, J., Gao, C., Rovinski, A., Mudge, T., Mars, J., & Tang, L. (2017). Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17)*. https://doi.org/10.1145/3037697.3037698
23. Madras, D., Pitassi, T., & Zemel, R. (2018). Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*.
24. Mozannar, H., & Sontag, D. (2020). Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning* (PMLR 119, pp. 7076–7087).
25. Mozannar, H., Lang, H., Wei, D., Sattigeri, P., Das, S., & Sontag, D. (2023). Who should predict? Exact algorithms for learning to defer to humans. *arXiv*:2301.06197.
26. Engelen, G., Rimmer, V., & Joosen, W. (2021). Troubleshooting an intrusion detection dataset: The CICIDS2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)* (pp. 7–12). IEEE. https://doi.org/10.1109/SPW53761.2021.00009
27. 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. In *Proceedings of the 31st USENIX Security Symposium* (pp. 3971–3988). USENIX Association.
28. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), Article 44. https://doi.org/10.1145/2523813
29. Elangovan, R., Parthasarathy, D. D., Jawahar, M., Kaliyaperumal, P., Balusamy, B., Yogarayan, S., & Venkatesan, V. (2026). Cross-dataset temporal and semantic generalization of intrusion detection models for the Future Internet. *Future Internet*, *18*(4), 194. https://doi.org/10.3390/fi18040194
30. 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. https://doi.org/10.3390/s23135941