CARE-IDS: Confidence-Aware Routing for Edge-Cloud Collaborative Network Intrusion Detection

Weipeng Zeng, Congpu Wang, Ahmad Affandi Supli*, Burra Venkata Durga Kumar
Computing and Data Science
Xiamen University Malaysia
Sepang, Malaysia
{clauvio.weipengzeng, hugh.wangcongpu}@outlook.com
{ahmadaffandi, venkata.burra}@xmu.edu.my

***Abstract*—Efficient Internet of Things intrusion detection requires balancing detection performance with communication and inference cost. This paper proposes CARE-IDS, a confidence-aware routing framework for edge-cloud collaborative intrusion detection, where each traffic sample is first processed by a lightweight Logistic Regression edge detector and then selectively offloaded to a Random Forest cloud detector when the edge prediction is estimated to be unreliable. To avoid manually selecting a fixed confidence threshold, CARE-IDS trains a Logistic Regression router from out-of-fold edge predictions to learn the relationship between edge confidence and prediction errors. Experiments on the DataSense dataset under an eight-class 10-fold cross-validation setting show that CARE-IDS achieves an accuracy of 92.57% and a Macro-F1 score of 89.65%, improving substantially over Edge-only inference while remaining close to Cloud-only performance. Meanwhile, it reduces cloud invocations by 83.36% and achieves a 5.80× model-side inference speedup over Cloud-only inference. These results demonstrate that confidence-aware routing can provide an effective performance-efficiency trade-off for IoT intrusion detection.**

***Index Terms*—Internet of Things, Distributed System, Intrusion Detection System.**

I. Introduction

The rapid deployment of Internet of Things (IoT) devices has significantly increased the scale and diversity of network traffic. Smart cameras, sensors, home automation devices, and industrial IoT terminals continuously generate heterogeneous traffic patterns, making IoT environments vulnerable to various cyberattacks, such as denial-of-service attacks, reconnaissance, brute-force attempts, malware infection, and web-based attacks [1]–[3]. Therefore, efficient and accurate intrusion detection is essential for maintaining the security and reliability of IoT networks. Existing intrusion detection systems can be deployed either at the edge or in the cloud, but these two deployment modes have different limitations. Edge-based intrusion detection performs inference close to the data source, which reduces communication overhead and enables fast local response. However, edge devices are often constrained by computation, memory, and energy resources, making it difficult to deploy complex models with strong detection capability. In contrast, cloud-based intrusion detection can use more powerful models and generally achieves better classification performance, but transmitting all traffic samples to the cloud introduces additional communication cost, inference delay, and potential privacy concerns. This trade-off is particularly important for IoT scenarios, where traffic is continuously generated and detection decisions are expected to be made in a timely manner [4]–[6]. This raises a central question: can an IoT intrusion detection system preserve most of the cloud model's detection capability while invoking the cloud only for samples that are likely to be difficult for the edge model? Recent hierarchical and multi-tier IDS frameworks already exploit multi-stage detection or distributed classification in IoT/IoV environments [7]–[9], but they seldom optimize sample-wise routing between edge and cloud under explicit performance–efficiency trade-offs. Instead of treating edge-only and cloud-only detection as two isolated choices, this paper studies an edge-first cloud-assisted detection strategy. In this strategy, each sample is first examined by a lightweight edge detector, and only potentially unreliable samples are selectively escalated to a stronger cloud detector. The key challenge is how to determine, for each incoming sample, whether the edge prediction is reliable enough to be accepted locally or should be offloaded to the cloud. To address this problem, we propose CARE-IDS, a Confidence-Aware Routing framework for Edge-Cloud Intrusion Detection. CARE-IDS follows a sample-level routing strategy: each traffic sample is first processed by a lightweight edge detector, and only samples with unreliable edge predictions are selectively offloaded to a stronger cloud detector. In our implementation, Logistic Regression is used as the edge detector because of its low computational cost, Random Forest is used as the cloud detector to provide stronger classification capability, and a Logistic Regression router is trained from out-of-fold edge predictions to learn the relationship between edge confidence and edge errors. In this way, CARE-IDS reduces unnecessary cloud invocation while preserving much of the cloud detectors detection capability. The main contributions of this paper are as follows:

*Corresponding author: Ahmad Affandi Supli.

- We propose CARE-IDS, a confidence-aware selective offloading framework for edge-cloud collaborative IoT intrusion detection.
- We construct router supervision from out-of-fold edge predictions, reducing optimistic bias when learning the relationship between edge confidence and edge errors.
- We evaluate CARE-IDS on the DataSense dataset under an eight-class 10-fold cross-validation protocol and analyze the trade-off among Macro-F1, cloud invocation ratio, and model-side inference time.

II. Related Work

A. IoT Intrusion Detection

Intrusion detection has become an important security mechanism for IoT networks because IoT devices are increasingly exposed to attacks. Compared with traditional networks, IoT environments usually contain heterogeneous and resource-constrained devices, which makes it challenging to achieve both accurate and efficient detection. Machine learning methods, including traditional classifiers and deep models, have been widely applied to IoT and IIoT intrusion detection, often combined with feature selection or dimensionality reduction to handle high-dimensional traffic data [10]–[13].

However, there is still a practical trade-off between detection accuracy and deployment efficiency. Lightweight models are suitable for edge deployment because of their low computational cost, but they may have limited classification capability for complex or imbalanced attack categories. More powerful models can improve detection performance, but they usually require more computation and are more suitable for cloud-side deployment. This motivates collaborative intrusion detection methods that combine the efficiency of edge inference with the stronger detection capability of cloud models.

B. Edge-Cloud Collaborative Inference

Edge-cloud collaborative inference has been extensively studied to balance latency, bandwidth and accuracy, including model-partitioning and multi-exit DNN approaches [4], [5], [14]. In this paradigm, edge devices perform local computation close to the data source, while cloud servers provide additional computational capability when local inference is insufficient. Compared with cloud-only inference, edge-cloud collaboration can reduce unnecessary data transmission. Compared with edge-only inference, it can improve prediction quality by allowing difficult samples to be processed by stronger cloud models.

Existing edge-cloud collaboration strategies can be generally divided into model-partitioning methods and sample-offloading methods. In intrusion detection, several works design lightweight edge-side models for resource-constrained IoT/IIoT and vehicular scenarios [14], [15], but typically do not study sample-wise routing policies between edge and cloud. This paper follows the sample-offloading strategy because it is simple, flexible, and suitable for combining a lightweight edge detector with a stronger cloud detector in IoT intrusion detection.

C. Confidence-Aware Routing

Confidence scores derived from predicted class probabilities are widely used as proxies for uncertainty, despite known calibration issues in modern deep networks [16]–[18]. Based on this idea, a common routing strategy is to define a fixed confidence threshold. Samples with confidence lower than the threshold are offloaded to the cloud, while samples with confidence higher than the threshold are handled locally. Although fixed-threshold routing is simple and efficient, its performance depends heavily on the selected threshold. A conservative threshold may offload too many samples and increase communication cost, while an aggressive threshold may keep too many uncertain samples at the edge and reduce detection accuracy. Different from using confidence only as a manually tuned threshold, CARE-IDS studies confidence-aware routing as a controllable edge-cloud offloading mechanism and further introduces a learnable router to automatically select a practical operating point.

III. Methodology

A. Overview of CARE-IDS

CARE-IDS is designed as a confidence-aware edge-cloud collaborative intrusion detection framework. As shown in Fig. 1, each incoming traffic sample is first processed by a lightweight edge detector, which outputs both a preliminary class prediction and a confidence score. A routing module then estimates whether the edge prediction is reliable. Reliable samples are handled locally, while uncertain samples are offloaded to a stronger cloud detector.

The framework consists of three components: an edge detector, a cloud detector, and a confidence-aware router. The edge detector provides low-cost first-stage inference, the cloud detector serves as a stronger fallback model, and the router determines whether cloud assistance is needed for each sample. This design allows CARE-IDS to separate the classification decision from the offloading decision, thereby improving detection performance without invoking the cloud for every sample.

B. Confidence-Aware Routing Decision

Let f_e denote the edge detector and f_c denote the cloud detector. For an input sample x, the edge detector outputs a probability distribution over K traffic classes,

$$\mathbf{p}_e(x) = [p_e(1 \mid x), p_e(2 \mid x), \ldots, p_e(K \mid x)], \tag{1}$$

and the preliminary edge prediction

$$\hat{y}_e(x) = \arg\max_k p_e(k \mid x). \tag{2}$$

The router does not directly use the original traffic feature vector as its input. Instead, it uses uncertainty-related information derived from the edge output. This design keeps the router lightweight and allows it to focus on estimating the reliability of the edge prediction. In general, the router input is represented as

$$u(x) = \phi(\mathbf{p}_e(x), \hat{y}_e(x)), \tag{3}$$

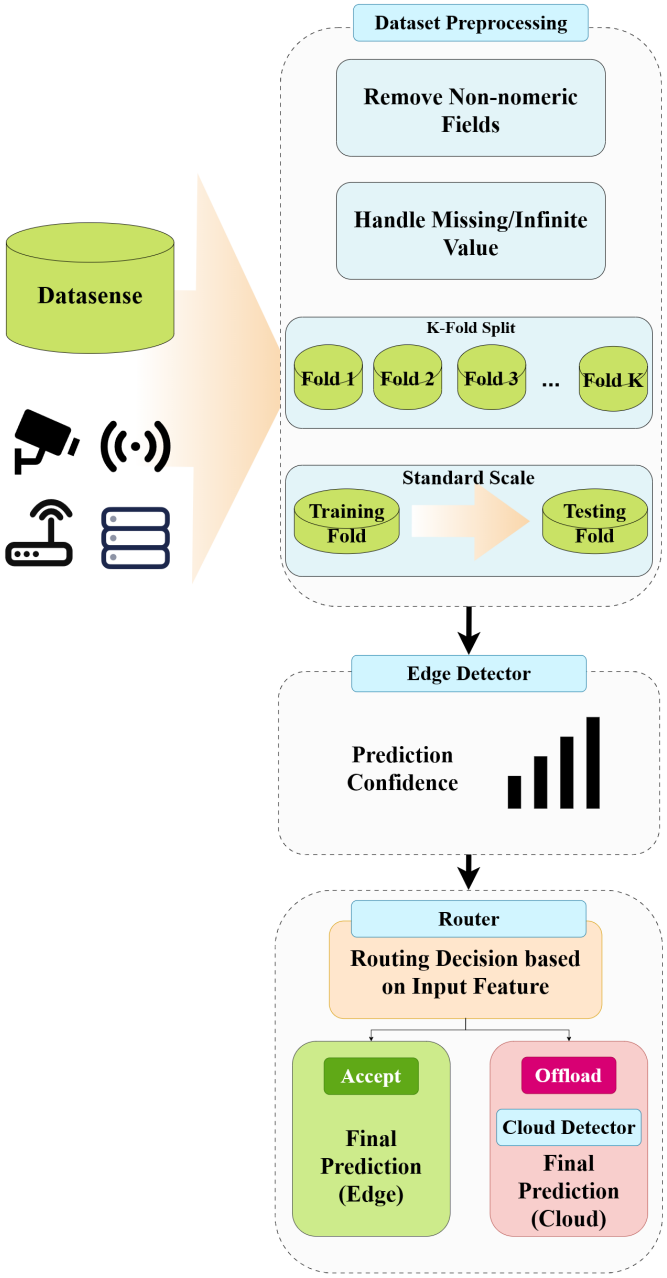


Fig. 1. Overview of CARE-IDS.

where $\phi(\cdot)$ denotes the uncertainty feature extraction function.

The basic routing feature is the top-1 confidence, defined as the maximum predicted probability. Additional features such as the margin between the largest and second-largest probabilities and the entropy of the predictive distribution can also be included to provide a richer description of edge-side uncertainty. In the ablation study, different combinations of confidence, margin, entropy, and predicted-class information are evaluated to analyze their influence on routing performance.

Given the uncertainty feature vector $u(x)$, the router g_θ outputs a binary routing decision,

$$r(x) = g_\theta(u(x)), \quad r(x) \in \{0, 1\}, \tag{4}$$

where $r(x) = 0$ means that the edge prediction is accepted locally, and $r(x) = 1$ means that the sample is offloaded to the cloud detector. The final prediction of the collaborative framework is therefore defined as

$$\hat{y}(x) = \begin{cases} \hat{y}_e(x), & r(x) = 0, \\ f_c(x), & r(x) = 1. \end{cases} \tag{5}$$

This decision rule allows the framework to preserve the efficiency of edge inference for reliable samples while invoking the cloud detector only when the edge output appears uncertain.

C. Out-of-Fold Supervision for the CARE-IDS Router

The learnable router in CARE-IDS is trained to estimate whether the edge prediction is reliable. Since the true correctness of the edge prediction is unavailable during inference, supervised router labels are constructed from the training data. For a training sample (x_i, y_i), the router label is defined according to whether the edge prediction matches the ground-truth label:

$$z_i = \begin{cases} 0, & \hat{y}_{e,i} = y_i, \\ 1, & \hat{y}_{e,i} \neq y_i. \end{cases} \tag{6}$$

A label $z_i = 0$ indicates that the edge prediction is correct and can be treated as reliable. A label $z_i = 1$ indicates that the edge prediction is incorrect and that the sample should be regarded as a candidate for cloud offloading. Therefore, router learning is formulated as a binary classification problem that maps edge-side uncertainty features to routing decisions.

A naive way to construct router training data is to train the edge detector on the full training set and then use its predictions on the same samples to generate router labels. However, this strategy can produce overly optimistic confidence estimates, because the edge model is evaluated on the samples it has already seen during training. As a result, the learned router may not accurately reflect the relationship between edge uncertainty and edge errors on unseen data.

To reduce this risk, the router training data are generated using out-of-fold edge predictions. Within each training fold, the training data are further divided into several inner stratified folds. For each inner fold, a temporary edge detector is trained on the remaining inner folds and then used to predict the held-out fold. Thus, each training sample receives an edge prediction and an edge probability distribution from a model that was not trained on that sample.

For each held-out sample, the uncertainty feature vector u_i is extracted from the out-of-fold edge probability distribution, and the corresponding router label z_i is generated by comparing the edge prediction with the ground-truth label. By aggregating all held-out predictions, the router training dataset is obtained as:

$$\mathcal{D}_r = \{(\mathrm{u}_i, \mathrm{z}_i)\}_{i=1}^{N}. \tag{7}$$

The router is then trained on $\mathcal{D}_r$ as a binary classifier. This out-of-fold construction makes the router learn from confidence-error relationships that better approximate test-time behavior, thereby reducing information leakage and improving the generalization of routing decisions.

After the router training dataset is constructed, the final edge detector and cloud detector are trained using the full training data. During testing, the final edge detector provides the probability distribution used by the router, while the cloud detector is invoked only for samples selected by the router.

During inference, each sample is first processed by the edge detector to obtain the probability vector and preliminary prediction. The router then decides whether to accept this prediction or send the sample to the cloud detector. Thus each sample requires one edge inference and one router evaluation, and cloud inference is triggered only when deemed necessary.

IV. Experiment

A. Dataset and Label Mapping

The experiments are conducted on the DataSense dataset, a real-time sensor-based benchmark dataset for IIoT attack analysis [2]. The dataset was collected from a realistic IIoT testbed containing industrial sensors, IoT devices, network infrastructure, edge and cloud components, and attacker nodes. It includes benign traffic and multiple attack scenarios covering seven major attack categories, including DDoS, DoS, reconnaissance, MITM, web-based attacks, brute-force attacks, and malware. In this study, the original labels are mapped into an eight-class classification setting consisting of benign and the seven attack categories.

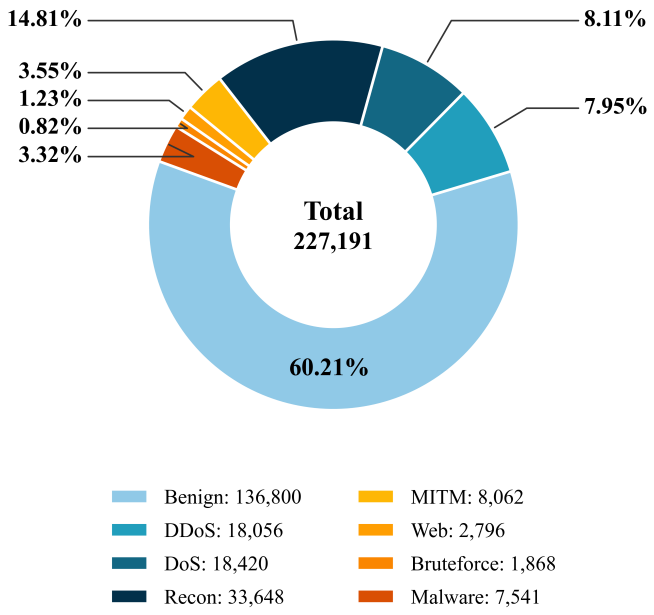


Fig. 2. DataSense class distribution.

B. Data Preprocessing and Fold Construction

Non-numeric and identifier-related columns, including device identifiers, MAC addresses, timestamps, and original label fields, are removed before model training. List-like fields such as IP lists, MAC address lists, port lists, protocol lists, and log data-type lists are also excluded to avoid sparse high-cardinality representations and identity-specific patterns. After this filtering, only numeric statistical features are retained. Missing and infinite values are replaced with zero, and feature columns are sorted to ensure consistent ordering across experimental runs.

The dataset is globally shuffled using a fixed random seed and then divided into 10 stratified folds. Each fold preserves the class distribution by distributing samples of each class across the folds. For each experimental run, one fold is used as the test set and the remaining nine folds are used for training. The same fold splits are used for all compared inference modes and router variants to ensure fair comparison.

C. Compared Inference Modes

Three inference modes are compared: Edge-only, Cloud-only, and CARE-IDS. In Edge-only mode, all test samples are classified by the lightweight edge detector without cloud invocation. In Cloud-only mode, all samples are processed by the cloud detector, which serves as the upper-performance reference but requires full cloud communication. In CARE-IDS mode, each sample is first processed by the edge detector, and the confidence-aware router decides whether to accept the edge prediction locally or offload the sample to the cloud detector.

The Edge-only and Cloud-only modes are used as baselines to evaluate the trade-off between local efficiency and detection performance. The CARE-IDS inference represents the proposed edge-cloud collaborative framework, where the goal is to approach the detection performance of Cloud-only inference while reducing the number of cloud invocations.

D. Model Configuration and Hyperparameters

The experimental environment is summarized in Table I. All models are implemented using scikit-learn. For each outer fold, a StandardScaler is fitted only on the training set and then applied to both the training and test sets. The model configurations and main hyperparameters are summarized in Table II.

TABLE I
Experimental Environment

Item	Specification
CPU	AMD Ryzen 5 7520U, 2.80 GHz
RAM	16 GB
OS	64-bit Windows
Python	3.12.12
scikit-learn	1.8.0
Execution mode	CPU-only

For the router ablation study, different uncertainty-related inputs are evaluated to examine whether additional uncertainty

TABLE II
MODEL CONFIGURATIONS AND MAIN HYPERPARAMETERS

Component	Model / Setting	Main Parameters
Edge detector	Logistic Regression	solver = lbfgs, max_iter = 2000
Cloud detector	Random Forest	n_estimators = 200, max_depth = None, random_state = 42, n_jobs = -1
Router	Logistic Regression	solver = lbfgs, max_iter = 2000, class_weight = balanced
Feature scaling	StandardScaler	fitted on training set only
Outer evaluation	Stratified 10-fold CV	random_state = 42
Router data generation	Inner stratified CV	5 folds

information can improve routing performance beyond top-1 confidence. Specifically, the evaluated router inputs include confidence only, confidence with margin, confidence with entropy, confidence with both margin and entropy, and confidence with margin, entropy, and predicted-class information.

E. Experimental Procedure

For each outer fold, the edge detector and cloud detector are trained on the training subset. The router training data are generated from out-of-fold edge predictions using the inner stratified cross-validation procedure, and the router is then trained as a binary classifier. During testing, Edge-only, Cloud-only, and CARE-IDS inferences are evaluated on the same test subset under a per-sample streaming protocol. In CARE-IDS inference, each sample is first processed by the edge detector, and the router decides whether to keep the edge prediction locally or offload the sample to the cloud detector.

F. Evaluation Metrics

Classification performance is evaluated using accuracy, macro-precision, macro-recall, and macro-F1. Macro-averaged metrics are used because the mapped eight-class dataset is imbalanced and each class should contribute equally to the final evaluation.

System efficiency is evaluated using cloud invocation ratio, cloud-call reduction, inference time, time per sample, and speedup over Cloud-only inference. The cloud invocation ratio is defined as:

$$CR = \frac{\mathrm{N}_{cloud}}{\mathrm{N}_{test}}, \tag{8}$$

where N_{cloud} is the number of samples offloaded to the cloud and N_{test} is the total number of test samples. For Edge-only, $CR = 0$; for Cloud-only, $CR = 1$; and for CARE-IDS, CR is determined by the router decisions.

Cloud-call reduction measures the proportion of cloud invocations avoided by CARE-IDS compared with Cloud-only inference:

$$CCR = 1 - CR. \tag{9}$$

The average inference time per sample is calculated as:

$$TPS = \frac{\mathrm{T}_{total}}{\mathrm{N}_{test}}, \tag{10}$$

where T_{total} is the total inference time over the test set. The speedup over Cloud-only inference is calculated as:

$$Speedup = \frac{\mathrm{T}_{cloud}}{\mathrm{T}_{hybrid}}, \tag{11}$$

where T_{cloud} and T_{hybrid} denote the total inference time of Cloud-only and CARE-IDS inferences, respectively. The reported inference time measures model-side execution time under the experimental environment and does not include real network transmission latency.

V. RESULTS AND DISCUSSION

A. Overall Comparison of Inference Modes

Table III reports the 10-fold average performance of Edge-only, Cloud-only, and CARE-IDS. Edge-only inference eliminates cloud communication and has the lowest model-side inference time, but its Macro-F1 is clearly lower than the other two modes. This indicates that the lightweight edge detector is efficient but insufficient as a standalone detector for the eight-class DataSense task.

Cloud-only inference achieves the best detection performance, confirming the stronger classification capability of the Random Forest cloud detector. However, it requires every sample to be processed by the cloud, resulting in the highest cloud invocation ratio and model-side inference cost. Therefore, Cloud-only inference provides an upper-performance reference rather than a communication-efficient deployment choice.

The CARE-IDS inference provides a better balance between the two extremes. It substantially improves upon Edge-only inference and remains close to Cloud-only performance, while invoking the cloud for only 16.64% of test samples on average. As a result, CARE-IDS reduces cloud calls by 83.36% and achieves a 5.80× model-side speedup over Cloud-only inference. These results show that confidence-aware offloading can preserve most of the cloud detectors benefit while avoiding unnecessary cloud invocation for the majority of samples.

B. Performance-Offloading Trade-off of Confidence-Aware Routing

To evaluate whether confidence-aware routing can provide a controllable balance between detection performance and cloud communication cost, we compare the learnable router with a series of fixed-threshold routing policies. In fixed-threshold routing, samples with edge confidence lower than a predefined threshold are offloaded to the cloud, while the remaining samples are handled locally by the edge detector. Therefore, changing the threshold produces different operating points between Edge-only and Cloud-only inference.

As shown in Fig. 3, fixed-threshold routing forms a clear performance-communication trade-off curve. When the thresh-

TABLE III
DETECTION PERFORMANCE OF DIFFERENT INFERENCE MODES UNDER 10-FOLD CROSS-VALIDATION

Method	Accuracy (%)	Macro-F1 (%)	Cloud Invocation Ratio (%)	Time per Sample (ms)
Edge-only	88.31 ± 0.17	77.69 ± 0.53	0.00	0.071 ± 0.000
Cloud-only	93.47 ± 0.21	91.09 ± 0.35	100.00	38.047 ± 0.319
CARE-IDS	92.57 ± 0.21	89.65 ± 0.32	16.64 ± 0.25	6.559 ± 0.109

old is low, only a small number of samples are offloaded, resulting in low communication cost but limited performance improvement. As the threshold increases, more uncertain samples are processed by the cloud detector, and Macro-F1 improves accordingly. However, the performance gain becomes smaller when the cloud invocation ratio becomes high, indicating diminishing returns from excessive cloud invocation.

The learnable router is located near the favorable region of the trade-off curve, close to the fixed-threshold setting around 0.85. This result shows that the learnable router can automatically identify a practical operating point that achieves competitive Macro-F1 while keeping the cloud invocation ratio low. Instead of requiring manual threshold tuning, the learnable router provides a data-driven way to select an effective routing boundary for edge-cloud collaborative intrusion detection.

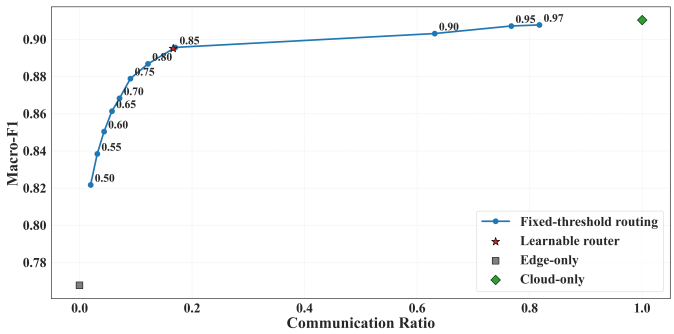


Fig. 3. Macro-F1 versus cloud invocation ratio.

C. Class-wise Detection Performance

To further examine the class-level behavior of the proposed framework, Fig. 4 compares the per-class F1-scores of Edge-only, Cloud-only, and CARE-IDS. The Edge-only detector achieves reasonable performance on several classes, but its weakness becomes clear on difficult attack categories such as bruteforce and mitm. In particular, a noticeable number of attack samples are misclassified as benign, indicating that the lightweight edge model has limited ability to separate some minority or ambiguous classes.

Cloud-only inference provides stronger class discrimination, as reflected by the higher per-class F1-scores. This confirms that the cloud detector is more reliable for difficult samples, but this benefit comes with full cloud invocation. In contrast, CARE-IDS improves the weak classes of Edge-only inference and achieves a per-class F1 pattern closer to Cloud-only inference, while still offloading only a limited portion of samples.

Overall, these class-wise results show that the proposed framework does not merely improve aggregate metrics. Instead, confidence-aware offloading helps recover part of the class-level performance lost by the lightweight edge detector, especially for categories where Edge-only inference is less reliable. For completeness, the normalized confusion matrices of Edge-only, CARE-IDS, and Cloud-only inferences are provided in the Appendix.

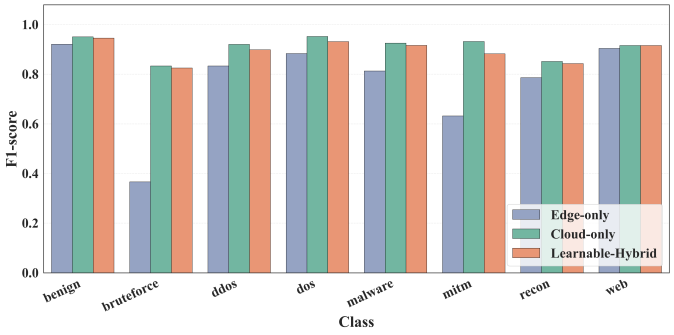


Fig. 4. Per-class F1-score comparison of Edge-only, Cloud-only, and CARE-IDS.

D. Confidence Distribution and Routing Interpretability

Fig. 5 visualizes the edge confidence distributions of correctly and incorrectly classified samples. The purpose of this analysis is to examine whether edge confidence can serve as a meaningful reliability signal for routing decisions. As shown in the figure, correctly classified samples are highly concentrated in the high-confidence region, whereas incorrectly classified samples are more widely distributed and appear more frequently in lower-confidence ranges.

This pattern supports the use of confidence-aware offloading. Low-confidence predictions are more likely to be unreliable and therefore are suitable candidates for cloud inspection, while high-confidence predictions can often be accepted locally to avoid unnecessary cloud invocation. The distribution also explains the trade-off observed in Fig. 3: increasing the confidence threshold offloads more uncertain samples and improves detection performance, but also increases the cloud invocation ratio.

The CARE-IDS router benefits from the same reliability signal. Instead of manually selecting a fixed confidence threshold, it learns a routing boundary from training data and automatically places the operating point near the effective confidence region. Therefore, the confidence distribution provides

TABLE IV
FEATURE ABLATION OF THE CARE-IDS ROUTER

Router Input	Accuracy (%)	Macro-F1 (%)	Cloud Invoc. Ratio (%)	Cloud Call Reduction (%)	Speedup
Confidence	92.57 ± 0.21	89.65 ± 0.32	16.64 ± 0.25	83.36 ± 0.25	5.80× ± 0.09×
Confidence + Margin	92.57 ± 0.22	89.65 ± 0.34	16.73 ± 0.27	83.27 ± 0.27	5.77× ± 0.09×
Confidence + Entropy	92.63 ± 0.20	89.63 ± 0.31	16.80 ± 0.29	83.20 ± 0.29	5.74× ± 0.09×
Confidence + Margin + Entropy	92.64 ± 0.21	89.72 ± 0.33	52.84 ± 0.49	47.16 ± 0.49	1.87× ± 0.02×
Confidence + Margin + Entropy + Pred. Class	92.61 ± 0.20	89.63 ± 0.30	52.65 ± 0.47	47.35 ± 0.47	1.87× ± 0.01×

an interpretable explanation for both fixed-threshold routing and the proposed learnable routing strategy.

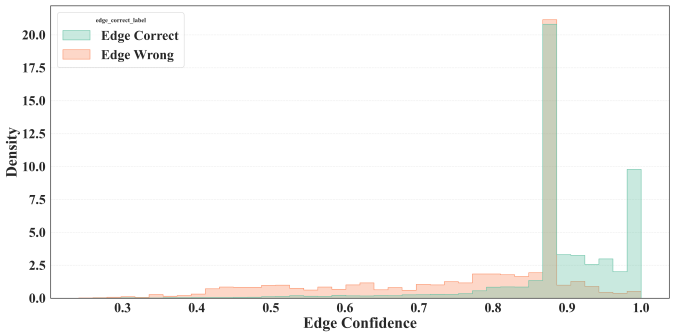


Fig. 5. Confidence distribution by edge correctness.

E. Router Feature Ablation

Table IV compares different input features for the CARE-IDS router. The confidence-only router already achieves a strong performance-efficiency balance, with Macro-F1 close to the best variant while maintaining the lowest cloud invocation ratio and the highest speedup. Adding margin or entropy produces only negligible changes in detection performance and communication cost, suggesting that top-1 confidence already provides sufficient reliability information for routing.

Using confidence, margin, and entropy together slightly improves Macro-F1, but this improvement comes at the cost of substantially more cloud invocations. Its cloud invocation ratio increases to more than 52%, and the speedup drops to about 1.87×. Adding predicted-class information shows a similar pattern and does not improve the overall trade-off. Therefore, the confidence-only router is used as the default configuration because it provides the best balance between detection performance, cloud-call reduction, and model-side inference efficiency.

F. Discussion and Limitations

The results show that confidence-aware offloading provides a practical middle ground between Edge-only and Cloud-only intrusion detection. Edge-only inference is efficient and communication-free, but it has limited capability for difficult or minority classes. Cloud-only inference achieves stronger detection performance, but it requires every sample to be processed by the cloud. The CARE-IDS inference uses the edge detector as a first-stage filter and invokes the cloud only for samples with higher estimated uncertainty, thereby improving detection performance while reducing unnecessary cloud calls.

The fixed-threshold analysis further shows that edge confidence forms a useful performance-communication trade-off. Different confidence thresholds correspond to different operating points, allowing the system to be adjusted according to deployment requirements. The CARE-IDS router provides an automatic alternative to manual threshold selection and falls near a favorable region of this trade-off curve. The router ablation also indicates that top-1 confidence alone is already a strong routing signal, while adding more uncertainty features brings little performance benefit and may increase cloud invocation.

This study still has several limitations. First, the experiments are conducted on a single dataset, so the generalizability of the framework should be further verified on additional IoT and IIoT intrusion detection datasets. Second, the evaluation is performed offline, and the reported speedup measures model-side inference time only. Real network transmission delay, cloud queuing, bandwidth fluctuation, and edge-device resource contention are not included. Third, the model combinations are limited to Logistic Regression for the edge detector and router, and Random Forest for the cloud detector. Future work will evaluate more datasets, deploy the framework in realistic edge-cloud environments, and examine additional model and routing configurations.

VI. CONCLUSION

This paper presented CARE-IDS, a confidence-aware edge-cloud collaborative intrusion detection framework for IoT network traffic. The framework uses a lightweight edge detector to process all samples first, a stronger cloud detector to handle uncertain samples, and a learnable confidence-aware router to decide whether each edge prediction should be accepted locally or offloaded to the cloud. Experiments on the DataSense dataset show that the proposed framework improves Macro-F1 from 0.7769 under Edge-only inference to 0.8965, while maintaining performance close to the Cloud-only Macro-F1 of 0.9109. At the same time, CARE-IDS reduces cloud invocations by 83.36% and achieves a 5.80× model-side speedup compared with Cloud-only inference. These results show that confidence-aware routing can preserve much of the cloud detectors detection capability while avoiding unnecessary cloud usage, suggesting its potential

as a lightweight selective offloading strategy for resource-constrained IoT intrusion detection.

Code Availability

The implementation of CARE-IDS is publicly available at: https://github.com/C1auvios1eepyhead/CARE-IDS-Confidence-Aware-Routing-for-Edge-Cloud-Collaborative-Intrusion-Detection-in-IoT-Networks.

References

[1] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2214212619305046

[2] A. Firouzi, S. Dadkhah, S. A. Maret, and A. A. Ghorbani, "DataSense: a real-time sensor-based benchmark dataset for attack analysis in IIoT with multi-objective feature selection," *Electronics*, vol. 14, no. 20, p. 4095, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/20/4095

[3] T. Sowmya and E. Mary Anita, "A comprehensive review of AI based intrusion detection system," *Measurement: Sensors*, vol. 28, p. 100827, 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2665917423001630

[4] R. Cordova-Cardenas, D. Amor, and . Gutirrez, "Edge AI in practice: A survey and deployment framework for neural networks on embedded systems," *Electronics*, vol. 14, no. 24, p. 4877, 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/24/4877

[5] T. Gong, L. Zhu, F. R. Yu, and T. Tang, "Edge intelligence in intelligent transportation systems: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 9, pp. 8919–8944, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10133894/

[6] Z. Zhang and J. Li, "A review of artificial intelligence in embedded systems," *Micromachines*, vol. 14, no. 5, p. 897, 2023. [Online]. Available: https://www.mdpi.com/2072-666X/14/5/897

[7] Md Ashraf Uddin, N. H. Chu, R. Rafeh, and M. Barika, "A scalable hierarchical intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 12, no. 20, pp. 41 591–41 607, 2025. [Online]. Available: http://arxiv.org/abs/2505.16215

[8] M. Verkerken, L. Dhooge, D. Sudyana, Y.-D. Lin, T. Wauters, B. Volckaert, and F. De Turck, "A novel multi-stage approach for hierarchical intrusion detection," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3915–3929, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10077796/

[9] L. Yang, A. Moubayed, and A. Shami, "MTH-IDS: a multitiered hybrid intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 616–632, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9443234/

[10] K. Albulayhi, Q. Abu Al-Haija, S. A. Alsuhibany, A. A. Jillepalli, M. Ashrafuzzaman, and F. T. Sheldon, "IoT intrusion detection using machine learning with a novel high performing feature selection method," *Applied Sciences*, vol. 12, no. 10, p. 5015, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/10/5015

[11] J. Li, M. S. Othman, H. Chen, and L. M. Yusuf, "Optimizing IoT intrusion detection system: feature selection versus feature extraction in machine learning," *Journal of Big DATA*, vol. 11, no. 1, p. 36, 2024. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-024-00892-y

[12] M. B. Musthafa, S. Huda, Y. Kodera, M. A. Ali, S. Araki, J. Mwaura, and Y. Nogami, "Optimizing IoT intrusion detection using balanced class distribution, feature selection, and ensemble machine learning techniques," *Sensors*, vol. 24, no. 13, p. 4293, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/13/4293

[13] S. Walling and S. Lodh, "Enhancing IoT intrusion detection through machine learning with AN-SFS: a novel approach to high performing adaptive feature selection," *Discover Internet of Things*, vol. 4, no. 1, pp. 16–41, 2024. [Online]. Available: https://link.springer.com/10.1007/s43926-024-00074-5

[14] F. Dong, H. Wang, D. Shen, Z. Huang, Q. He, J. Zhang, L. Wen, and T. Zhang, "Multi-exit DNN inference acceleration based on multi-dimensional optimization for edge intelligence," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9769868/

[15] T. J. Mohammed, A. Alnoor, F. Abdelfattah, X. Chew, and K. W. Khaw, "Lightweight principal component analysis-driven ensemble framework for real-time intrusion detection in industrial IoT networks," *Cybersecurity*, vol. 9, no. 1, pp. 46–61, 2026. [Online]. Available: https://link.springer.com/10.1186/s42400-025-00445-7

[16] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020. [Online]. Available: https://www.nature.com/articles/s42256-020-00257-z

[17] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big DATA*, vol. 6, no. 1, p. 27, 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0192-5

[18] M. Jordahn and P. M. Olmos, "Decoupling feature extraction and classification layers for calibrated neural networks," *arXiv preprint arXiv:2405.01196*, pp. 1–20, 2023.

Appendix

Confusion Matrices

This appendix provides the normalized confusion matrices corresponding to the Edge-only, CARE-IDS, and Cloud-only inferences.

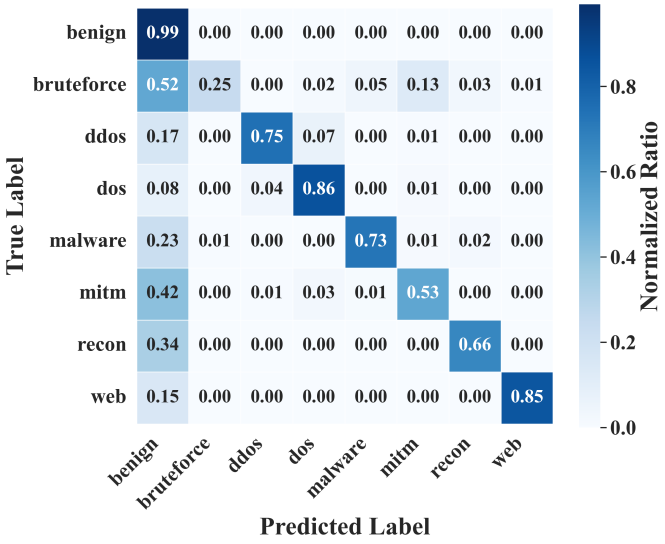


Fig. 6. Normalized confusion matrix of Edge-only inference.

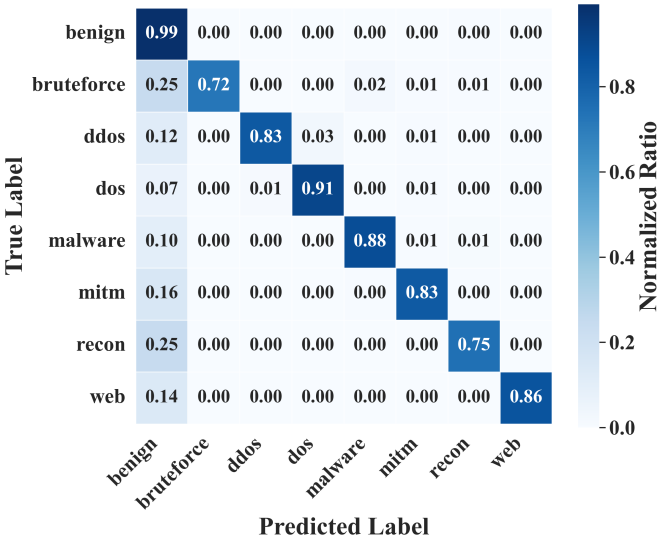


Fig. 7. Normalized confusion matrix of CARE-IDS inference.

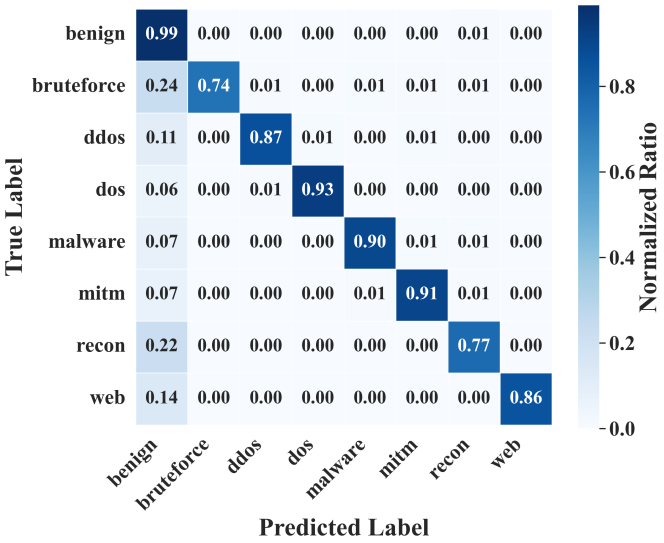


Fig. 8. Normalized confusion matrix of Cloud-only inference.