An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.x, No.x, pp.xx-xx (2026)
https://doi.org/10.36922/IJOCTA026270147

RESEARCH ARTICLE

LARO-IDS: Family-leakage-aware robust multi-objective optimization for model selection in IoT intrusion detection

Ameen Shaheen*, Wael Alzyadat, and Aysh Alhroob

Department of Software Engineering, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, Jordan
a.shaheen@zuj.edu.jo, wael.alzyadat@zuj.edu.jo, aysh@zuj.edu.jo

ARTICLE INFO

Article History:
Received: July 5, 2026
Revised: July 17, 2026
Accepted: August 3, 2026
Published Online: August 14, 2026

Keywords:
Multi-objective optimization
Robust model selection
Internet of Things intrusion detection
Cross-family robustness
Edge deployment

AMS Classification 2010:
90C29; 90C90

ABSTRACT

Machine learning-based intrusion detection systems (IDS) are commonly selected based on conventional validation or development metrics, although such criteria may not sufficiently reflect robustness against unseen attack families or suitability for resource-constrained Internet of Things and edge environments. This study proposes learned acquisition and reconstruction optimization (LARO)—IDS (LARO-IDS), a family-leakage-aware robust multi-objective optimization framework for model selection in Internet of Things intrusion detection. Instead of selecting the model that only maximizes conventional predictive performance, LARO-IDS jointly considers development macro-F1, mean cross-family robustness, worst-family behavior, robustness variability, and prediction latency in the candidate-selection objective, while training time and model size are retained as additional deployment-cost indicators for final comparison. Candidate models were evaluated using a model-selection evaluation subset and a leave-one-attack-family-out robustness protocol, then ranked using a weighted-sum scalarization of normalized objectives, with the results further supported by Pareto-efficiency analysis. Experiments on the CICIoT2023 dataset show that conventional score-based selection favors RF_03_regularized, which achieved the highest macro-F1. In contrast, LARO-IDS selects RF_01_fast, which preserves nearly identical predictive performance, with only a -0.0015 macro-F1 difference, while achieving slightly higher mean cross-family F1 scores across attack families. The LARO-selected model also reduces training time by 49.49%, prediction latency by 46.36%, and model size by 50.12% compared with the conventionally selected model. Sensitive analysis of objective weights further shows that RF_01_fast remains selected under balanced, performance-priority, robustness-priority, and edge-priority scenarios. These results demonstrate that robust IDS model selection should integrate family-leakage-aware robustness and latency-aware deployment cost rather than relying solely on conventional predictive performance.

1. Introduction

The rapid expansion of Internet of Things (IoT) technologies and edge-connected devices has increased the need for intrusion detection systems (IDS) that can operate reliably under heterogeneous traffic patterns, limited computational resources, and continuously evolving attack behaviors. Machine learning-based intrusion detection has therefore become a widely adopted approach for identifying malicious network flows; recent studies and surveys have shown the effectiveness of supervised learning, ensemble learning, and deep learning methods for network and IoT security applications.[1,2]

*Corresponding Author

The availability of recent IoT and Industrial IoT (IIoT) benchmark datasets has further supported the development of data-driven intrusion detection models under realistic attack scenarios. TON_IoT, for example, introduced heterogeneous telemetry, operating system, and network data sources for IoT and IIoT security evaluation.[3] Edge-IIoTset also provided a comprehensive dataset for centralized and federated learning scenarios in IoT and IIoT environments.[4] In this study, CICIoT2023 is used because it provides a large-scale IoT attack benchmark with diverse attack types collected from a topology involving real-world IoT devices, making it suitable for evaluating intrusion detection models under multiple attack categories.[5]

Although machine learning-based intrusion detection models have frequently achieved high reported performance, the way these models are usually evaluated and selected does not always provide enough evidence of their reliability in deployment environments. In many studies, model selection is mainly based on accuracy, F1-score, or validation performance obtained using random or predefined data splits. However, a model that performs well under these conditions may not continue to perform reliably when it is tested against unseen attack families, different traffic distributions, or constraints that arise during deployment. This issue is consistent with earlier observations that network intrusion detection is a difficult application area for machine learning because operational networks are open, adversarial, and non-stationary.[6]

Cross-evaluation studies have also demonstrated that intrusion detection models can suffer from considerable performance degradation when they are evaluated under conditions that are different from those used during training and validation.[7] Similarly, cross-dataset generalization studies have shown that strong performance within one dataset does not necessarily lead to reliable performance when the model is applied to another intrusion detection dataset.[8] These observations indicate that the criterion used to select the model should include robustness and generalization as part of the model-selection process itself, rather than considering them only as post-selection analyses after the model has already been chosen.

The importance of this issue becomes clearer in IoT and edge IDS, where the most suitable model is not necessarily the model with the highest validation score alone. In these settings, deployment requires a model that can maintain good predictive performance while also remaining efficient with respect to inference latency, memory footprint, and operational cost. Edge computing was proposed to bring computation closer to end devices in order to reduce latency and improve the response time of applications that require real-time decisions.[9]

Within edge-intelligence environments, the interaction between deployed models and limited edge resources makes computational efficiency an essential requirement for practical use.[10] Recent edge-intelligence architectures also show that edge deployment should take into account system-level constraints such as communication cost, inference time, and available resources.[11] As a result, a model that provides only a small improvement in validation performance may not be the most appropriate choice if this improvement comes with higher memory requirements, slower inference, or reduced stability across unseen attack families. Thus, intrusion detection model selection should be viewed as a multi-objective process in which predictive performance, robustness, worst-case behavior, and computational cost are considered together.

Multi-objective optimization represents an appropriate framework for model selection in cases where the selected model is expected to satisfy conflicting requirements. Rather than depending on one performance metric as the only basis for selection, multi-objective optimization aims to identify candidate solutions that provide different balances among several objectives and therefore make the trade-offs among these objectives more explicit.[12] This is the main reason why Pareto-based optimization methods are useful, as in many model-selection problems, no single candidate can dominate all other candidates with respect to every objective.[13]

Within machine learning, multi-objective hyperparameter optimization has been increasingly considered a suitable methodology when predictive performance has to be evaluated together with other practical factors, including prediction time, robustness, sparsity, fairness, interpretability, and resource usage.[14] Similar survey work has also noted that many traditional hyperparameter optimization methods still focus on a single performance measure, although real-world machine learning applications usually include multiple objectives that may conflict.[15] This view is in line with the wider multi-objective and data-driven optimization literature, where candidate solutions are compared by modeling the conflicts between objectives and by identifying possible trade-off alternatives instead of relying on only one scalar criterion.[16,17]

Nevertheless, in many IoT intrusion detection evaluation pipelines, model selection remains primarily driven by classification performance, while

robustness to unseen attack families and edge-deployment cost are still rarely included as part of the formal model-selection objective.

To address this gap, this study proposes learned acquisition and reconstruction optimization (LARO)–IDS (LARO-IDS), a family-leakage-aware robust multi-objective optimization framework for model selection in IoT intrusion detection. In this study, family-leakage-aware selection refers to a model-selection protocol in which robustness is evaluated under held-out attack-family conditions, reducing the risk that the final model is selected only because it performs well on attack-family-specific patterns shared between the training and testing sets. The central idea is that cross-family evaluation signals should not only be reported as diagnostic results but should directly influence the selection of the final model. LARO-IDS evaluates candidate models using conventional predictive performance, leave-one-attack-family-out robustness, worst-family performance, robustness variability, and prediction latency within the scalar selection objective. Training time and model size are retained as complementary deployment-cost indicators for final comparative analysis. These criteria are then combined into a normalized multi-objective score and complemented by Pareto-efficiency analysis to identify models that provide robust and deployment-efficient trade-offs.

The evaluation of the proposed framework is performed on the CICIoT2023 dataset through a step-by-step experimental design. Initially, conventional held-out performance is reported for several model families to establish the standard classification results. Subsequently, the leave-one-attack-family-out protocol is used to test whether the models can maintain their performance when a complete attack family is not seen during training. Next, the candidate model configurations are compared using two selection strategies: the conventional score-based criterion and the proposed LARO-based criterion. Finally, a weight-sensitivity analysis is used to examine whether changing the importance of performance, robustness, and deployment-related costs affects the selected model.

The results show that the conventional criterion selects a model mainly because of a small advantage in the macro-F1 score. However, the LARO criterion selects a more deployment-efficient model that achieves nearly the same predictive performance, provides greater average robustness across unseen attack families, and substantially reduces training time, inference latency, and model size. Moreover, the selected model remains the same under the performance-priority, robustness-priority, and edge-priority scenarios, which provides further support for the stability of the LARO selection objective.

The main contributions of this study are as follows:

(i) Leakage-aware model selection for IoT intrusion detection is formulated as a multi-objective optimization problem that jointly considers conventional predictive performance, cross-family robustness, worst-family behavior, robustness stability, and latency-aware deployment cost.

(ii) The LARO-IDS objective is introduced to integrate robustness-oriented evaluation signals and prediction latency cost into a unified model-selection criterion, while training time and model size are retained for final deployment-cost comparison.

(iii) A leave-one-attack-family-out robustness protocol is designed to assess whether candidate intrusion detection models generalize beyond attack families encountered during training.

(iv) Ranking instability and weight sensitivity are analyzed by comparing conventional score-based selection with LARO-based robust selection, showing that the preferred model changes when robustness and deployment cost are considered and remains consistent across multiple LARO weighting scenarios.

(v) The experimental evaluation on the CICIoT2023 demonstrates that LARO can select a model with almost unchanged macro-F1 score while reducing training time, prediction latency, and model size by approximately half compared with conventional score-based selection.

The remainder of this study is organized as follows. Section 2 reviews related work on machine learning-based IoT intrusion detection, leakage-aware evaluation approaches, and multi-objective model selection. Section 3 presents the problem formulation and the proposed LARO-IDS framework. Section 4 describes the dataset, preprocessing, candidate models, and experimental protocol. Section 5 reports and discusses the experimental results. Section 6 concludes the study and outlines potential future research directions.

2. Literature review

2.1. Machine learning-based intrusion detection

Machine learning-based methods have become widely adopted in intrusion detection because they provide a practical way to learn patterns from network-flow features and use these patterns to

detect malicious traffic. In IoT intrusion detection, recent studies have compared different machine learning algorithms within IoT network environments, which highlights the need for algorithm-level evaluation rather than treating all models as having the same behavior.[18] Tree-based ensemble models are among the most suitable methods for structured tabular security data as they can learn nonlinear relationships between features and often achieve strong classification results without extensive preprocessing. Within this group, random forest (RF) has been widely used as an established and reliable baseline for classification tasks.[19] Extremely randomized trees (Extra Trees) are closely related to RF, but they introduce additional randomization during the construction of trees, which can reduce variance and make the learning process more efficient in some settings.[20] Similarly, gradient-boosting methods are widely adopted in IDS pipelines, where XGBoost provides a scalable boosting framework, and LightGBM provides an efficient implementation for high-dimensional and large-scale intrusion detection datasets.[21,22]

The task of intrusion detection becomes more challenging in IoT and IIoT networks because these networks are composed of different types of devices, rely on diverse communication protocols, and may be exposed to attack behaviors that evolve over time. Previous studies on IoT intrusion detection have demonstrated that machine learning methods are capable of providing effective intrusion detection performance in IoT applications. However, these studies also showed that the obtained performance is not only related to the selected algorithm but also to the dataset used for training and testing, the representation of the extracted features, and the evaluation parameters adopted in the experiment.[23] Recent survey work has also argued that practical evaluation of IDS models in IoT environments should take into account dataset limitations, the choice of performance measures, and computational feasibility, rather than relying solely on a single aggregate measure such as overall accuracy.[24] Despite this, a large part of the existing literature continues to evaluate proposed models mainly through classification accuracy or overall F1-score. This type of reporting may make a model appear strong at the overall level, while hiding weak detection ability for rare attack types or for attack categories that were not included in the training data.

Another issue that needs to be considered when evaluating intrusion detection models is that many IDS datasets contain strong class imbalance, repeated traffic patterns, and characteristics that are specific to the dataset itself. These issues can make the performance reported within the same dataset appear stronger than the performance that may be obtained when the model is deployed under real deployment conditions. Previous survey work on network-based intrusion detection datasets indicated that the design of the dataset, the diversity of attack types, the feature extraction process, and how well the dataset reflects realistic network traffic are all important factors that affect the validity of IDS evaluation.[25] Therefore, the performance of a model should not be understood only from the reported classification scores, but should also be examined in relation to the evaluation protocol and the diversity of the attack scenarios used during testing.

2.2. Evaluation bias and generalization in intrusion detection

The difficulty of evaluating machine learning-based IDS models is mainly related to the gap between performance in controlled experimental settings and performance in real network environments. In particular, a model may achieve strong results during experimentation because it has learned dataset-specific regularities or repeated traffic patterns, instead of learning more general attack behavior that remains valid when the network environment changes. This concern is consistent with the discussion presented by Sommer *et al.*,[6] who explained that network intrusion detection should not be treated in the same way as many traditional machine learning problems, since operational environments are open, adversarial, and non-stationary. As a result, performance obtained under a closed experimental setting should be interpreted carefully, because it does not provide full assurance that the model will continue to behave reliably after deployment.

Generalization becomes particularly important when an intrusion detection model is tested against attacks that are unseen during training. If the training and testing samples share similar artifacts, traffic-generation procedures, or attack signatures, the reported evaluation results may overestimate the actual robustness of the model under practical deployment conditions. Previous cross-evaluation studies have shown that IDS models can suffer from performance degradation when they are evaluated under conditions that differ from the original training setup.[7] Cross-dataset generalization studies have also demonstrated that strong performance on one dataset does not necessarily transfer to other datasets.[8] In this context, deployment-oriented evaluation should include testing scenarios where the attack distributions, traffic condi-

tions, or attack families are different between the training and testing stages.

Many benchmark datasets have been proposed to make the evaluation of IDS models more realistic and representative of different attack conditions. The Bot-IoT dataset, for example, was designed to support IoT botnet detection and network forensic analysis using large-scale attack traffic.[26] Similarly, UNSW-NB15 introduced a comprehensive benchmark for network intrusion detection that includes modern attack categories and flow-based features.[27] The availability of these datasets, along with later IoT and IIoT benchmarks, shows that IDS models should not be evaluated only through a single closed-world split but should also be tested under broader and more diverse evaluation conditions.

The present study follows this line of evaluation by using the leave-one-attack-family-out protocol as an additional way to assess model robustness. Unlike conventional multi-class testing, this protocol removes one complete attack family from the training data and then tests whether the model is still able to recognize traffic from that excluded family as malicious traffic. The purpose of this evaluation is not to replace the conventional held-out evaluation, but to provide an additional view of how the selected model behaves when it is exposed to attack behavior that was not observed during training. In this way, the protocol helps determine whether the model is only performing well under the standard evaluation setting, or whether it can also maintain reliable detection performance when facing unseen attack-family behaviors.

2.3. Edge-oriented intrusion detection system and computational constraints

Internet-of-Things intrusion detection system is usually expected to operate close to the source of the data, such as on gateways, edge nodes, or lightweight monitoring devices. For this reason, computational cost cannot be treated as a secondary issue when selecting an IDS model for deployment. A model may achieve a slightly higher predictive score, but this does not necessarily mean that it is the most suitable model for practical deployment. If the model requires a much larger memory footprint or produces slower inference, then its practical usefulness may be limited, especially in edge-based environments where computational resources are limited. Recent IoT IDS survey work has emphasized that practical evaluation should consider computational efficiency and real-world testing conditions in addition to standard predictive performance metrics.[24] Efficiency-oriented NIDS research has also shown that detection-engine performance can be improved through parallel approximate matching, highlighting the importance of runtime cost in practical intrusion detection systems deployment.[28]

The need for efficient deployment is also supported by recent work on lightweight IDS for industrial IoT and edge environments. Lightweight optimized IDS models have been proposed to reduce computational overhead and improve real-time feasibility while maintaining high detection performance.[29] Similarly, recent edge-oriented IDS research has emphasized that many existing models are too complex for deployment on edge servers, motivating lightweight feature selection and compact detection models.[30] More broadly, model-compression research highlights the importance of reducing model size and resource requirements for deployment in mobile, edge, and IoT systems.[31] Feature-selection research in network intrusion detection has also emphasized that reducing the feature space can help control training cost, computational overhead, and resource consumption while preserving detection performance.[32]

This consideration provides the motivation for including the cost-aware component in the proposed LARO-IDS framework. In this framework, the selected model is not chosen only because it achieves strong conventional held-out performance. Instead, prediction latency is included within the candidate-selection objective so that the selection process can account for the model's inference speed. Training time and model size are also reported as additional indicators of deployment cost in the final comparison. By doing so, LARO-IDS makes the model-selection process more suitable for edge-oriented intrusion detection, where a model should not only perform well in terms of prediction, but should also be practical in terms of speed and resource requirements.

2.4. Multi-objective model selection

The selection of a suitable model is not always a matter of improving a single performance value, especially when the selected model is expected to satisfy several requirements at the same time. In machine learning, hyperparameter optimization is usually applied to improve the predictive performance of machine learning models. However, in many practical applications, the model that achieves the best predictive result may not necessarily be the most appropriate model for deployment. This is because other factors, such as latency, robustness, interpretability, fairness, and computational resource consumption, may also

affect the suitability of the selected model for deployment. Classical hyperparameter search methods, such as random search, showed that the final performance of a model can be strongly influenced by the configuration space used during the search process, and that efficient search can sometimes provide better results than exhaustive grid-based search.[33] Later optimization frameworks, such as Optuna, have supported this direction by providing practical and scalable tools for hyperparameter optimization and automated model selection.[34]

However, many model-selection and hyperparameter-optimization pipelines are still often applied with a primary predictive-performance objective. In deployment-oriented IDS, the selection process should also consider robustness under unseen attack families and computational cost. Multi-objective model selection has been studied in contexts where accuracy alone is not a sufficient criterion. For example, Pareto-based model-selection approaches have been used to balance predictive performance with training time and latency, allowing practitioners to identify candidates that represent different trade-offs rather than selecting a single optimal model.[35] More generally, model selection can be formulated as a multi-objective optimization problem in which competing criteria, such as model fit and model complexity, are balanced through Pareto-oriented reasoning.[36] Recent IDS-focused work has also formulated feature-selection and detection-pipeline design as multi-objective optimization problems, showing that predictive performance, selected-feature size, error behavior, and runtime may represent competing deployment-oriented objectives.[37]

The key distinction in the present study is that multi-objective selection is applied to family-leakage-aware and robustness-aware IDS evaluation. Instead of optimizing only accuracy or macro-F1, LARO-IDS combines conventional performance, mean cross-family robustness, worst-family behavior, robustness variability, and latency-aware deployment costs. This provides a more deployment-oriented selection criterion than conventional score-based model ranking.

2.5. Positioning of the present study

Existing work has established that machine learning can provide strong intrusion detection performance, that IDS datasets strongly influence evaluation validity, and that deployment conditions can differ from those in closed-world experimental settings.[23–27] Prior evaluation studies have also emphasized that operational reliability cannot be inferred from within-dataset performance alone.[6–8] At the same time, optimization and model-selection studies show that practical machine learning deployment often requires balancing predictive performance with additional objectives such as latency, robustness, stability, and resource consumption.[31–37]

The present study builds on these insights but shifts the main focus from performance evaluation to optimization-driven model selection. The objective is not merely to show that model rankings change under different evaluation settings. Instead, LARO-IDS incorporates robustness and cost signals into the model-selection objective itself. This positioning distinguishes the proposed framework from conventional IDS benchmarking studies, where robustness and computational cost are often reported after model selection rather than being integrated into the selection rule. By combining conventional predictive performance, cross-family robustness, worst-case behavior, stability, and latency-aware deployment cost, LARO-IDS provides a deployment-oriented multi-objective model-selection framework for IoT intrusion detection (**Figure 1**).

3. Problem formulation and LARO-IDS framework

This section presents the proposed LARO-IDS framework for family-leakage-aware robust model selection in IoT intrusion detection. In this study, family-leakage-aware selection refers to evaluating candidate models under held-out attack-family conditions so that the final model is not selected solely based on performance patterns shared between training and testing sets containing similar attack families. The framework shifts the model-selection process from conventional single-metric selection toward a multi-objective decision process that jointly considers predictive performance, cross-family robustness, worst-case behavior, robustness stability, and latency-aware deployment costs. Multi-objective optimization is suitable for this setting because no single intrusion detection model is expected to dominate all alternatives across accuracy, robustness, latency, and resource requirements.[12] Pareto-efficient analysis further provides a principled way to identify candidate models that represent different robustness-cost trade-offs.[13]

3.1. Problem definition

Let the intrusion detection dataset be defined as:

$$D = \{(x_i, y_i)\}_{i=1}^{N} \tag{1}$$

where $x_i \in \mathbb{R}^d$ denotes the d-dimensional network-flow feature vector of the i-th sample, $y_i \in \mathcal{Y}$ denotes its class label, $\mathcal{Y}$ denotes the set of benign and attack labels, and N is the total number of samples.

Since individual attack labels may belong to broader behavioral categories, each label is mapped to an attack family using the following taxonomy function:

$$g: \mathcal{Y} \rightarrow \mathcal{F} \tag{2}$$

where $g(\cdot)$ denotes the label-to-family mapping function, and $\mathcal{F} = \{f_1, f_2, \ldots, f_L\}$ denotes the set of L attack families. This mapping enables robustness evaluation at the attack-family level rather than only at the individual class-label level.

The goal of LARO-IDS is to select a final model from a candidate pool:

$$\mathcal{M} = \{m_1, m_2, \ldots, m_K\} \tag{3}$$

where $\mathcal{M}$ denotes the set of candidate IDS models, m_i denotes the i-th candidate model, and K is the number of candidates. Each candidate corresponds to a specific learning algorithm and associated hyperparameter configuration. Conventional model selection usually chooses the model with the highest validation or development performance score. In contrast, LARO-IDS selects the model that provides the best trade-off among conventional predictive performance, cross-family robustness, worst-family reliability, robustness stability, and deployment cost.

The framework starts from the CICIoT2023 dataset and constructs an attack-family taxonomy to support family-leakage-aware robustness evaluation. A pool of candidate IDS models is then generated from different model families and hyperparameter configurations. Each candidate is evaluated under conventional held-out performance, leave-one-attack-family-out robustness, and deployment-cost assessment. The LARO score integrates development macro-F1, mean cross-family robustness, worst-family behavior, robustness variability, and prediction latency cost. Training time and model size are retained as complementary deployment-cost indicators for the final comparison between the LARO-selected and conventionally selected candidates. The resulting scores are complemented by Pareto-efficiency analysis to select a robust and deployment-efficient IDS candidate.

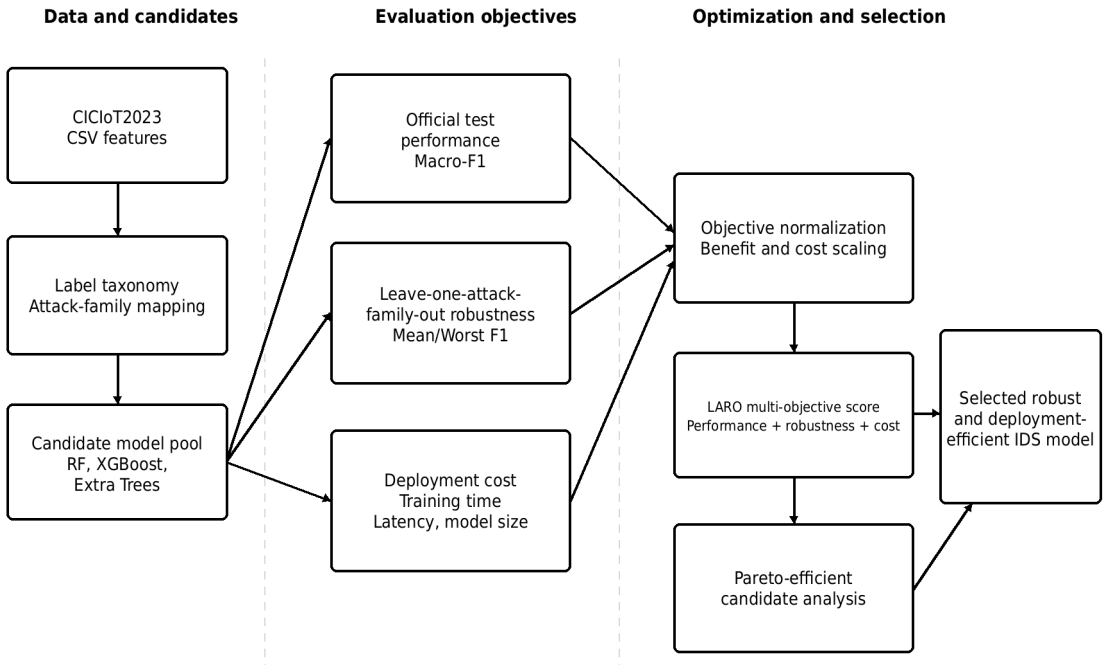

Figure 1. Overall architecture of the proposed LARO-IDS framework
Abbreviations: Extra Trees: Extremely randomized trees; IDS: Intrusion detection system; LARO: Learned acquisition and reconstruction optimization; RF: Random forest.

3.2. Candidate model pool

Each candidate model is represented as:

$$m_i = A_i(\theta_i) \tag{4}$$

where A_i denotes the learning algorithm used by the i-th candidate model and θ_i denotes its hyperparameter configuration. The candidate pool may include different model families and multiple hyperparameter configurations within the same family. This allows LARO-IDS to compare not only predictive performance but also the effect of model complexity on robustness and deployment costs.

In this study, the candidate pool includes ensemble-based and boosting-based configurations because these models are effective for structured tabular intrusion detection datasets. The purpose of the candidate pool is not only to identify the highest-scoring model under conventional evaluation but also to expose trade-offs among conventional performance, cross-family robustness, worst-case behavior, and computational cost.

3.3. Development predictive performance objective

For each candidate model, the development predictive performance is evaluated on the held-out model-selection subset. The main development performance metric used in the LARO objective is macro-F1, defined as:

$$F_{\text{off}}(m_i) = \frac{1}{|\mathcal{Y}|} \sum\nolimits_{c \in \mathcal{Y}} F1_c(m_i) \tag{5}$$

where the development macro-F1 score denotes the predictive performance of a candidate model on the model-selection subset, the number of class labels defines the averaging scope, and the class-wise F1-score is computed independently for each class. Macro-F1 is used because it gives equal importance to all classes and is more informative than accuracy under highly imbalanced class distributions.

The development performance objective is treated as a maximization objective:

$$\max_{m_i \in \mathcal{M}} F_{\text{off}}(m_i) \tag{6}$$

where maximizing this objective favors candidate models with stronger predictive performance on the development set. However, this objective alone is not sufficient for deployment-oriented IDS selection because a model with high development performance may still perform poorly on unseen attack families or require excessive computational resources.

3.4. Leave-one-attack-family-out robustness objective

To evaluate robustness under unseen attack-family behaviors, LARO-IDS uses a leave-one-attack-family-out protocol. For each attack family $f_j \in \mathcal{F}$, all samples belonging to f_j are excluded from the training data. The trained model is then evaluated on a test set containing benign traffic and samples from the held-out attack family f_j. This setting approximates a deployment scenario in which the IDS encounters an attack family that was not observed during training. This protocol is family-leakage-aware because the held-out attack family does not contribute training samples, reducing the risk that model selection is driven by attack-family-specific artifacts shared across training and testing sets.

For candidate model m_i and held-out family f_j, the cross-family attack F1-score is defined as:

$$F_{\text{cf}}(m_i, f_j) = F1_{\text{attack}}(m_i \mid f_j) \tag{7}$$

where $F_{\text{cf}}(m_i, f_j)$ denotes the attack F1-score achieved by candidate m_i when attack family f_j is excluded from training and used as the held-out attack family during testing. The term $F1_{\text{attack}}(m_i \mid f_j)$ denotes the binary attack-class F1-score under the corresponding held-out attack-family condition.

The mean cross-family robustness of candidate m_i is computed as:

$$\bar{F}_{\text{cf}}(m_i) = \frac{1}{L} \sum\nolimits_{j=1}^{L} F_{cf}(m_i, f_j) \tag{8}$$

where $\bar{F}_{\text{cf}}(m_i)$ denotes the average robustness of the candidate model m_i across all held-out attack families, and L denotes the number of evaluated attack families. This objective measures the average ability of a model to detect traffic from unseen attack families.

The mean cross-family robustness objective is defined as:

$$\max_{m_i \in \mathcal{M}} \bar{F}_{\text{cf}}(m_i) \tag{9}$$

where maximizing $\bar{F}_{\text{cf}}(m_i)$ favors candidates that generalize better across unseen attack-family scenarios.

3.5. Worst-family robustness and stability

Average robustness may hide severe degradation within a specific attack family. Therefore, LARO-IDS also considers the worst-case attack-family robustness:

$$F_{\text{worst}}(m_i) = \min_{f_j \in \mathcal{F}} F_{\text{cf}}(m_i, f_j) \tag{10}$$

where $F_{\text{worst}}(m_i)$ denotes the lowest cross-family attack F1-score achieved by candidate model m_i across all held-out families. This metric captures the weakest generalization condition for the candidate model.

The worst-family robustness objective is defined as:

$$\max_{m_i \in \mathcal{M}} F_{\text{worst}}(m_i) \tag{11}$$

where maximizing $F_{\text{worst}}(m_i)$ encourages the selection of models that avoid severe robustness degradation under difficult unseen attack-family conditions.

Robustness stability is measured using the standard deviation of the cross-family attack F1-scores:

$$\sigma_{\text{cf}}(m_i) = \sqrt{\frac{1}{L}\sum_{j=1}^{L}\left(F_{\text{cf}}(m_i, f_j) - \bar{F}_{\text{cf}}(m_i)\right)^2} \tag{12}$$

where $\sigma_{\text{cf}}(m_i)$ denotes the variability of candidate model m_i's cross-family robustness across held-out attack families. A lower value indicates more stable robustness performance across unseen attack-family conditions.

Robustness variability is treated as a minimization objective:

$$\min_{m_i \in \mathcal{M}} \sigma_{cf}(m_i) \tag{13}$$

where minimizing $\sigma_{cf}(m_i)$ favors candidates with more consistent cross-family robustness performance.

3.6. Deployment-cost objectives

For IoT and edge-oriented intrusion detection, model deployment cost is an essential part of deployment-oriented model selection. In this study, three practical cost indicators are considered: training time, prediction latency, and model size. Prediction latency is included in the scalar LARO selection objective because it is available consistently across all retained candidates and directly reflects model inference efficiency. Training time and model size are retained as complementary deployment-cost indicators for the final comparison between the LARO-selected and conventionally selected models:

$$C_{\text{train}}(m_i) = T_{\text{train}}(m_i) \tag{14}$$

$$C_{\text{pred}}(m_i) = \frac{T_{\text{pred}}(m_i)}{N_{eval}} \tag{15}$$

$$C_{\text{size}}(m_i) = S_{\text{model}}(m_i) \tag{16}$$

where $C_{\text{train}}(m_i)$ denotes the training-time cost of candidate m_i, $T_{\text{train}}(m_i)$ denotes the total training time, $C_{\text{pred}}(m_i)$ denotes the average prediction latency per sample, $T_{\text{pred}}(m_i)$ denotes the total prediction time on the evaluation set, N_{eval} denotes the number of evaluated samples, and $S_{\text{model}}(m_i)$ denotes the stored model size.

These cost indicators are treated as minimization objectives as follows:

$$\min_{m_i \in \mathcal{M}} C_{\text{train}}(m_i), \min_{m_i \in \mathcal{M}} C_{\text{pred}}(m_i), \min_{m_i \in \mathcal{M}} C_{\text{size}}(m_i) \tag{17}$$

where minimizing these objectives favors models that are more suitable for resource-constrained IoT and edge deployment environments.

3.7. Objective normalization

Because the LARO objective combines metrics with different scales and optimization directions, all objectives are normalized before the final aggregation step. For a benefit objective $B(m_i)$, the normalized value is computed as:

$$\widehat{B}(m_i) = \frac{B(m_i) - \min_{m \in \mathcal{M}} B(m)}{\max_{m \in \mathcal{M}} B(m) - \min_{m \in \mathcal{M}} B(m)} \tag{18}$$

where $B(m_i)$ denotes a benefit objective for candidate m_i, and $\widehat{B}(m_i)$ denotes its normalized value in the range $[0, 1]$. Higher values indicate better performance after normalization.

For a cost objective $C(m_i)$, the normalized value is inverted so that higher values remain preferable:

$$\widehat{C}(m_i) = \frac{\max_{m \in \mathcal{M}} C(m) - C(m_i)}{\max_{m \in \mathcal{M}} C(m) - \min_{m \in \mathcal{M}} C(m)} \tag{19}$$

where $C(m_i)$ denotes a cost objective for candidate m_i, and $\widehat{C}(m_i)$ denotes its normalized cost score. This transformation assigns higher values to candidates with lower deployment costs.

3.8. Learned acquisition and reconstruction optimization multi-objective score

The proposed LARO score combines conventional predictive performance, mean cross-family robustness, worst-family robustness, robustness stability, and prediction-latency costs. The scalar LARO score uses the objectives available for all retained candidates in the optimization process. Training time and model size are retained as additional deployment-cost indicators for the final comparison between the LARO-selected and conventionally selected candidates. The score for a candidate

is defined as:

$$S_{\mathrm{LARO}}(m_i) = w_1 \widehat{F}_{\mathrm{off}}(m_i) + w_2 \widehat{\bar{F}}_{\mathrm{cf}}(m_i) + w_3 \widehat{F}_{\mathrm{worst}}(m_i) + w_4 \widehat{\sigma}_{\mathrm{cf}}(m_i) + w_5 \widehat{C}_{\mathrm{pred}}(m_i) \quad (20)$$

where denotes the final LARO score of the candidate, denotes normalized development macro-F1, denotes normalized mean cross-family robustness, denotes normalized worst-family robustness, denotes the normalized inverted robustness-variability cost, and denotes normalized inverted prediction-latency cost. The coefficients represent the relative importance of the objectives.

The weights satisfy:

$$w_r \geq 0, \quad \sum\nolimits_{r=1}^{5} w_r = 1 \quad (21)$$

where w_r denotes the weight assigned to the r-th objective. This constraint ensures that the LARO score is a convex weighted aggregation of normalized objectives, following the common weighted-sum scalarization principle for multi-objective optimization.[38]

In the experimental implementation, the balanced LARO weights were selected to prioritize robust deployment-oriented model selection while still preserving conventional predictive performance. Higher weights were assigned to mean cross-family robustness and worst-family robustness because the objective of LARO-IDS is to avoid selecting models that perform well only under conventional held-out testing. The latency objective was assigned a nonzero weight to ensure that deployment efficiency contributes to the final decision. **Table 1** summarizes the objective weights used in the balanced LARO score.

Table 1. Objective weights used in the balanced learned acquisition and reconstruction optimization score

Objective	Symbol	Weight
Development macro-F1	w_1	0.20
Mean cross-family attack F1-score	w_2	0.25
Worst-family attack F1-score	w_3	0.25
Robustness stability	w_4	0.15
Prediction latency	w_5	0.15

The selected model is the candidate model with the maximum LARO score:

$$m^* = \arg\max_{m_i \in \mathcal{M}} S_{\mathrm{LARO}}(m_i) \quad (22)$$

where m^* denotes the final IDS model selected by the proposed LARO-IDS framework. This selection rule prioritizes candidates that provide strong robustness and deployment efficiency while preserving conventional predictive performance.

3.9. Pareto-efficient selection

In addition to the scalar LARO score, the framework identifies Pareto-efficient candidates. Let the multi-objective vector of candidate m_i be defined as:

$$z(m_i) = \begin{bmatrix} F_{off}(m_i), \bar{F}_{cf}(m_i), F_{worst}(m_i), \\ -\sigma_{cf}(m_i), -C_{pred}(m_i) \end{bmatrix} \quad (23)$$

where $z(m_i)$ denotes the objective vector of candidate m_i. Benefit objectives are included directly, while cost objectives are multiplied by -1 so that all dimensions can be interpreted as maximization objectives.

A candidate m_a dominates another candidate m_b if:

$$z_q(m_a) \geq z_q(m_b), \quad \forall q, \\ \text{and } \exists q : z_q(m_a) > z_q(m_b) \quad (24)$$

where $z_q(m_i)$ denotes the q-th component of the objective vector $z(m_i)$. This condition means that m_a is at least as good as m_b in all objectives and strictly better in at least one objective.

The Pareto-efficient set is defined as:

$$\mathcal{P} = \left\{ \begin{matrix} m_i \in \mathcal{M} \mid \nexists m_k \\ \in \mathcal{M} \text{ such that } m_k \succ m_i \end{matrix} \right\} \quad (25)$$

where denotes the set of Pareto-efficient candidates and indicates that candidate dominates candidate. Pareto analysis complements the LARO score by showing whether the selected model belongs to the set of non-dominated alternatives.[13,16]

The full LARO-IDS model-selection procedure is summarized in **Algorithm 1**. The algorithm first constructs the attack-family taxonomy, generates candidate models, and evaluates each candidate under development and leave-one-attack-family-out settings. It then computes robustness and cost indicators, normalizes all objectives, calculates the LARO score, identifies Pareto-efficient candidates, and returns the selected IDS model. In the candidate-pool optimization stage, the scalar LARO score uses the objectives available for all retained candidates. Additional deployment-cost indicators, including training time and model size, are used in the final comparison between the LARO-selected and conventionally selected candidates.

This formulation makes the proposed framework distinct from conventional IDS benchmarking. Instead of selecting the model that only maximizes conventional predictive performance, LARO-IDS selects a candidate that offers a stronger balance among predictive accuracy, robustness

against unseen attack families, worst-case reliability, and latency-aware deployment efficiency. Training time and model size are then used to verify the practical deployment-cost advantage of the selected model.

Algorithm 1. LARO-IDS: Family-leakage-aware robust multi-objective model-selection procedure

Input: Dataset D, attack-family set F, candidate model set M, objective weights w_1, ..., w_5.
Output: Selected robust IDS model m^*.
1. Construct the attack-family mapping g: $Y \rightarrow F$.
2. Generate the candidate model pool $M = \{m_1, m_2, ..., m_K\}$.
3. For each candidate model $m_i \in M$ do:
   3.1 Train m_i on the capped training subset.
   3.2 Evaluate development predictive performance $F_{dev}(m_i)$.
   3.3 For each attack family $f_j \in F$ do:
      3.3.1 Exclude samples from f_j during training.
      3.3.2 Train m_i using the remaining attack families and benign samples.
      3.3.3 Test the trained model on benign samples and samples from the held-out family f_j.
      3.3.4 Compute $F_{cf}(m_i, f_j)$.
   3.4 Compute mean cross-family robustness $\bar{F}_{cf}(m_i)$.
   3.5 Compute worst-family robustness $F_{worst}(m_i)$.
   3.6 Compute robustness variability $\sigma_{cf}(m_i)$.
   3.7 Measure prediction latency $C_{pred}(m_i)$.
   3.8 Record training time and model size as complementary deployment-cost indicators.
4. Normalize all benefit and cost objectives across the candidate pool.
5. Compute $S_{LARO}(m_i)$ for each candidate using the balanced LARO weights.
6. Identify the Pareto-efficient set P.
7. Select the final model:
$m^* = arg\ max_{mi \in M}\ S_{LARO}(m_i)$
8. Return m^*.

4. Dataset and experimental protocol

This section describes the dataset, preprocessing steps, candidate model pool, evaluation settings, and performance metrics used to validate the proposed LARO-IDS framework. The experimental design supports both conventional intrusion detection evaluation and family-leakage-aware model selection. Therefore, the protocol includes conventional held-out evaluation, leave-one-attack-family-out robustness evaluation, multi-candidate model selection, weight-based LARO selection, Pareto-efficiency analysis, and deployment-cost comparison.

4.1. Dataset description

The experiments were conducted using CICIoT2023, a large-scale IoT intrusion detection dataset designed to evaluate machine learning-based security models under diverse IoT attack scenarios.[5] The dataset contains benign traffic and multiple attack categories collected from an IoT network environment involving real devices.

In this study, the CSV feature files were used because they provide structured network-flow features suitable for machine learning-based intrusion detection.

In this study, the dataset includes 34 class labels, with one label corresponding to benign traffic and 33 labels corresponding to attack traffic. To make the evaluation more suitable for cross-family robustness analysis, the original attack labels were not treated solely as isolated class labels. Instead, these labels were reorganized into eight broader traffic and attack families, namely Benign, Distributed Denial of Service (DDoS), Denial of Service (DoS), Mirai, Recon, Spoofing, BruteForce, and Web_Malware, as presented in **Table 2**, which also reports the original training, validation, and testing counts for each label.

The reason for using this grouping is that a label-based evaluation alone may show whether the model can classify the attacks that it has already seen during training, but it does not clearly show whether the model can generalize to a broader attack family that was completely removed from the training. Therefore, this family-level mapping allows the study to evaluate whether the candidate models can generalize beyond the original attack

labels and still detect traffic from a held-out attack family as malicious. The reason for using this grouping is that a label-based evaluation alone may show whether the model can classify the attacks that it has already seen during training, but it does not clearly show whether the model can generalize to a broader attack family that was completely removed from the training. Therefore, this family-level mapping allows the study to evaluate whether the candidate models can generalize beyond the original attack labels and still detect traffic from a held-out attack family as malicious.

Table 2. CICIoT2023 label taxonomy and attack-family mapping

Label	Attack family	Training	Validation	Testing	Total
BenignTraffic	Benign	129,538	27,519	27,709	184,766
DictionaryBruteForce	BruteForce	1,541	290	319	2,150
DDoS-ICMP_Flood	DDoS	848,088	182,011	180,447	1,210,546
DDoS-UDP_Flood	DDoS	637,558	136,466	136,717	910,741
DDoS-TCP_Flood	DDoS	528,499	113,888	113,735	756,122
DDoS-PSHACK_Flood	DDoS	481,254	102,985	103,326	687,565
DDoS-SYN_Flood	DDoS	478,653	102,644	102,208	683,505
DDoS-RSTFINFlood	DDoS	475,441	101,563	101,819	678,823
DDoS-SynonymousIP_Flood	DDoS	422,083	90,795	90,480	603,358
DDoS-ICMP_Fragmentation	DDoS	53,046	11,430	11,402	75,878
DDoS-UDP_Fragmentation	DDoS	34,169	7,221	7,224	48,614
DDoS-ACK_Fragmentation	DDoS	33,581	7,194	7,292	48,067
DDoS-HTTP_Flood	DDoS	3,371	706	709	4,786
DDoS-SlowLoris	DDoS	2,757	620	622	3,999
DoS-UDP_Flood	DoS	390,422	83,446	83,627	557,495
DoS-TCP_Flood	DoS	314,174	67,056	67,697	448,927
DoS-SYN_Flood	DoS	237,573	51,115	51,116	339,804
DoS-HTTP_Flood	DoS	8,487	1,833	1,805	12,125
Mirai-greeth_flood	Mirai	116,133	24,910	24,934	165,977
Mirai-udpplain	Mirai	104,814	22,314	22,536	149,664
Mirai-greip_flood	Mirai	88,821	18,867	19,279	126,967
Recon-HostDiscovery	Recon	15,737	3,482	3,331	22,550
Recon-OSScan	Recon	11,587	2,500	2,433	16,520
Recon-PortScan	Recon	9,648	2,083	2,082	13,813
VulnerabilityScan	Recon	4,396	905	913	6,214
Recon-PingSweep	Recon	249	45	53	347
MITM-ArpSpoofing	Spoofing	36,316	7,741	7,840	51,897
DNS_Spoofing	Spoofing	21,214	4,565	4,570	30,349
BrowserHijacking	Web_Malware	665	158	134	957
SqlInjection	Web_Malware	590	148	148	886
CommandInjection	Web_Malware	620	132	119	871
XSS	Web_Malware	414	91	103	608
Backdoor_Malware	Web_Malware	392	93	89	574
Uploading_Attack	Web_Malware	140	35	33	208

Abbreviations: DoS: Distributed Denial of Service; DDoS: Denial of Service.

The distribution of samples in the dataset was not balanced across the attack classes. High-frequency attacks, including DDoS-ICMP_Flood and DDoS-UDP_Flood, contained hundreds of thousands of training samples, while low-frequency attacks, including Uploading_Attack, Recon-PingSweep, and Backdoor_Malware, had substantially fewer training samples. Such a distribution can make overall accuracy less informative, since the final accuracy score may mostly reflect the ability of the model to classify the classes that occur most often. This does not necessarily mean that the model is also performing well on rare or underrepresented attacks. Accordingly, macro-F1,

balanced accuracy, and attack-family-level robustness were used in addition to accuracy to support a fairer interpretation of the model's performance.

4.2. Preprocessing and sampling strategy

The original CSV feature files were used without packet capture-level processing. The class label column was encoded for model training, while all remaining columns were treated as numerical network-flow features. No feature engineering based on the test labels was introduced, and the attack-family mapping was used only for evaluation and robustness splitting.

To make the candidate-pool experiments more computationally feasible and to reduce the dominance of highly frequent attack labels, a capped balanced training subset was constructed from the original training file. A maximum of 20,000 samples was selected per label when available, while all samples were retained for labels with fewer than 20,000 training instances. This produced a development and model-selection training subset of 440,594 samples across all 34 labels.

For conventional held-out evaluation, a capped model-selection evaluation subset was constructed from the held-out evaluation split using a maximum of 5,000 samples per label when available, while retaining all samples for rare labels with fewer than 5,000 instances. This produced a model-selection evaluation subset of 107,463 samples across all 34 labels. The capped held-out subset was used as the development-stage model-selection evaluation subset for comparing candidate models, not as a completely untouched final external test set. Therefore, the reported LARO ranking should be interpreted as a model-selection decision within the experimental protocol rather than as an independent final deployment estimate. The LARO objective weights were predefined before candidate ranking and were not learned from the held-out subset. Evaluation using a fully untouched external test set, cross-dataset validation, and temporal or device-level evaluation are left for future work. The purpose of this sampling strategy was to preserve all rare attack labels while maintaining a computationally manageable and less frequency-dominated evaluation setting. **Table 3** summarizes the capped training subset used for candidate model selection.

Table 3 shows that the sampling strategy reduced the dominance of highly frequent attack labels while preserving all available low-frequency attack classes. This is important because the proposed framework evaluates both conventional classification performance and robustness under difficult attack-family conditions.

Table 3. Distribution of the capped training subset used for candidate model selection

Label	Count	Percentage (%)
DDoS-UDP_Flood	20,000	4.54
MITM-ArpSpoofing	20,000	4.54
DDoS-SYN_Flood	20,000	4.54
BenignTraffic	20,000	4.54
DoS-SYN_Flood	20,000	4.54
DNS_Spoofing	20,000	4.54
Mirai-greip_flood	20,000	4.54
DDoS-RSTFINFlood	20,000	4.54
Mirai-udpplain	20,000	4.54
DDoS-ICMP_Flood	20,000	4.54
DDoS-ACK_Fragmentation	20,000	4.54
DDoS-SynonymousIP_Flood	20,000	4.54
DoS-TCP_Flood	20,000	4.54
DDoS-TCP_Flood	20,000	4.54
DoS-UDP_Flood	20,000	4.54
DDoS-UDP_Fragmentation	20,000	4.54
DDoS-ICMP_Fragmentation	20,000	4.54
DDoS-PSHACK_Flood	20,000	4.54
Mirai-greeth_flood	20,000	4.54
Recon-HostDiscovery	15,737	3.57
Recon-OSScan	11,587	2.63
Recon-PortScan	9,648	2.19
DoS-HTTP_Flood	8,487	1.93
VulnerabilityScan	4,396	1.00
DDoS-HTTP_Flood	3,371	0.77
DDoS-SlowLoris	2,757	0.63
DictionaryBruteForce	1,541	0.35
BrowserHijacking	665	0.15
CommandInjection	620	0.14
SqlInjection	590	0.13
XSS	414	0.09
Backdoor_Malware	392	0.09
Recon-PingSweep	249	0.06
Uploading_Attack	140	0.03

4.3. Candidate model pool

The LARO-IDS framework was evaluated using a pool of candidate machine learning models generated using three model families: RF, Extra Trees, and XGBoost. RF was included as a strong ensemble-learning baseline for structured classification tasks.[19] Extra Trees was included as a more randomized tree-ensemble alternative that can reduce variance and provide different robustness–computational-cost trade-offs.[20] XGBoost was included as a scalable gradient-boosting method that is effective for structured tabular data and large-scale classification problems.[21] These model families were selected because they are suitable for structured intrusion detection features and provide different trade-offs between predictive performance, robustness, and computational cost.

The candidate pool included multiple hyperparameter configurations for each model family. XGBoost candidates varied in the number of estimators, maximum tree depth, learning rate, subsampling ratio, column-sampling ratio, and regularization settings. RF and Extra Trees candidates varied in the number of trees, maximum depth, minimum leaf size, and feature-sampling strategy. This design allowed the study to compare both inter-family and intra-family trade-offs instead of treating each model family as a single fixed baseline.

The final optimization candidate pool used in the LARO analysis consisted of 10 candidates after excluding clearly underperforming Extra Trees configurations from the robustness sweep. The retained pool included four XGBoost candidates, four RF candidates, and two Extra Trees candidates. Each candidate was evaluated using the same conventional held-out and cross-family robustness protocol, ensuring that the LARO score was computed from comparable performance, robustness, stability, and latency indicators. The retained candidate pool should therefore be understood as the optimization search space that was actually evaluated in this study, not as a complete list of all possible IDS algorithms and hyperparameter settings. The Extra Trees configurations that were excluded were removed before the final LARO normalization and Pareto analysis because their development results showed that they were not sufficiently competitive. Keeping these configurations in the full robustness analysis would have increased the computational cost without materially affecting the comparison among the stronger candidate models. Therefore, the min-max normalization, Pareto membership, and LARO ranking were calculated only using the retained candidate pool. Accordingly, the selected model should be interpreted as the best trade-off among the evaluated candidates, not as a universal optimal model across all possible IDS models.

4.4. Conventional held-out evaluation

The first evaluation setting was used to assess the conventional predictive performance of each candidate model on the held-out model-selection subset. In this setting, each candidate model was trained using the capped training subset and then evaluated using the capped held-out subset prepared for model selection. Macro-F1 was used as the main conventional performance metric because it assigns equal importance to each class, making it more suitable for imbalanced multi-class intrusion detection problems. In addition to macro-F1, weighted-F1, accuracy, and balanced accuracy were also reported to provide a broader view of the classification performance of the candidate models.

The purpose of this evaluation setting is to represent the conventional model-selection approach, where the selection decision is typically based on the highest score obtained during the development or validation stage. Therefore, in this study, the candidate model with the highest development macro-F1 score was treated as the conventional score-based selection baseline. This baseline serves as the reference case in which model selection depends mainly on predictive performance measured on the held-out model-selection

4.5. Leave-one-attack-family-out robustness evaluation

The second evaluation setting measured robustness against unseen attack-family behavior. For each attack family, all samples belonging to that family were removed from the training data. The model was then trained on the remaining data and tested on a binary detection task consisting of benign traffic and the held-out attack family. The target labels for this setting were "Benign" and "Attack."

This protocol supports family-leakage-aware model selection because the held-out attack family is excluded from training, reducing the possibility that robustness estimates are influenced by attack-family–specific patterns shared between training and testing. In this study, the term "family-leakage-aware" is used in this specific attack-family-level sense. It means that samples from the held-out attack family do not contribute to model training during the robustness evaluation. It does not imply that all possible leakage sources are eliminated, such as device-level, session-level, timestamp-level, source-address-level, duplicated-flow, or data-generation-artifact leakage. Addressing these additional leakage sources would require group-, device-, session-, or time-aware splitting strategies, which fall beyond the current experimental scope and are considered future work.

This protocol was repeated for the following held-out attack families: DDoS, DoS, Mirai, Recon, Spoofing, Web_Malware, and BruteForce. Benign traffic was not used as a held-out attack family because it represents the normal class rather than an attack family. For each held-out family, attack precision, attack recall, attack F1-score, balanced accuracy, true positives, false positives, true negatives, and false negatives were recorded.

The leave-one-attack-family-out protocol was used to estimate how well a candidate model can generalize to detecting attack behavior that

was not observed during training. Unlike conventional multi-class evaluation, this setting focuses on deployment-relevant robustness against unseen attack-family behavior.

4.6. Learned acquisition and reconstruction optimization candidate selection protocol

After development and cross-family evaluations, each candidate model was summarized according to the objectives defined in Section 3. The candidate-pool LARO score was computed using the objectives that were available for all retained candidates: development macro-F1, mean cross-family attack F1, worst-family attack F1, standard deviation of cross-family attack F1, and prediction latency.

The LARO score was computed by normalizing benefit and cost objectives and aggregating them using the balanced objective weights reported in **Table 1**. Higher LARO scores indicate better trade-offs between predictive performance, cross-family robustness, robustness stability, and latency-aware deployment efficiency. Pareto-efficiency analysis was also applied to identify non-dominated candidates across performance, robustness, and deployment-cost objectives.

Training time and model size were retained as complementary deployment-cost indicators and were used in the final comparison between the LARO-selected candidate and the conventionally selected candidate. This separation ensures that the LARO score is computed consistently across the full candidate pool using objectives available for all retained candidates, while the final deployment-cost analysis also considers training cost and model footprint for the selected models.

The weighted-sum LARO score was used as a transparent preference-based decision rule rather than as a learned optimization model. The objective weights were predefined to reflect a deployment-oriented preference for robustness and inference efficiency while preserving conventional predictive performance. Because minimum–maximum normalization is computed over the retained candidate pool, the resulting scores are relative to the evaluated candidates and may change if a substantially different candidate pool or normalization strategy is used. For this reason, the weight-sensitivity analysis was included to examine whether the selected candidate remains stable under different performance-, robustness-, and edge-priority preferences.

The candidate selected by conventional development macro-F1 was then compared with the candidate selected by the LARO score. This comparison was used to determine whether the proposed robust multi-objective selection framework could identify a model with comparable predictive performance, stronger robustness–cost trade-off, and improved deployment suitability.

4.7. Evaluation metrics

For conventional multi-class evaluation, the following metrics were used:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{26}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. In the multi-class case, accuracy represents the proportion of correctly classified samples.

The class-wise F1-score is defined as:

$$F1_c = \frac{2 \cdot P_c \cdot R_c}{P_c + R_c} \tag{27}$$

where P_c denotes precision for class c, R_c denotes recall for class c, and $F1_c$ denotes the F1-score of class c.

Macro-F1 is computed as:

$$\text{Macro-F1} = \frac{1}{|\mathcal{Y}|} \sum\nolimits_{c \in \mathcal{Y}} F1_c \tag{28}$$

where |Y| denotes the number of class labels. Macro-F1 gives equal importance to all classes and is therefore appropriate for evaluating imbalanced intrusion detection datasets.

Weighted-F1 is computed as:

$$\text{Weighted-F1} = \sum\nolimits_{c \in \mathcal{Y}} \frac{n_c}{N} F1_c \tag{29}$$

where n_c denotes the number of samples belonging to class c, and N denotes the total number of samples. Weighted-F1 accounts for class frequency and is useful for understanding performance under the original sample distribution.

Balanced accuracy is defined as:

$$\text{Balanced accuracy} = \frac{1}{|\mathcal{Y}|} \sum\nolimits_{c \in \mathcal{Y}} R_c \tag{30}$$

where R_c is the recall of class c. This metric is useful when class frequencies are highly imbalanced.

For the leave-one-attack-family-out binary evaluation, attack precision, attack recall, and attack F1-score were used as the main robustness metrics. Attack precision measures the proportion of predicted attack samples that are truly attacks, while

attack recall measures the proportion of held-out attack samples that are detected as attacks. Attack F1-score summarizes the trade-off between these two values and is used as the main cross-family robustness indicator.

4.8. Computational environment and cost measurement

All experiments were conducted on a Windows-based workstation equipped with an NVIDIA GeForce RTX 4050 laptop GPU with 6 GB of graphics processing unit (GPU) memory. XGBoost models were trained using GPU acceleration, while RF and Extra Trees models were trained using CPU parallelism. The implementation used Python with scikit-learn for classical machine learning models, preprocessing utilities, and evaluation metrics,[39] and the XGBoost library for GPU-accelerated gradient-boosted tree training.[21]

Training time was measured as the elapsed wall-clock time required to fit each model. Prediction latency was measured as the total prediction time divided by the number of evaluated samples. Model size was measured by saving the trained model to disk and reporting its compressed storage size in megabytes. These measurements were used as deployment-cost indicators in the LARO analysis and in the final comparison between conventional and LARO-based model selection.

4.9. Weight-sensitivity analysis

To examine whether the LARO decision depends on a specific weighting configuration, a weight-sensitivity analysis was conducted using four scenarios: balanced-LARO, performance-priority, robustness-priority, and edge-priority scenarios. The balanced-LARO scenario corresponds to the objective weights reported in **Table 1**. The performance-priority scenario assigns greater importance to development macro-F1, the robustness-priority scenario assigns greater importance to mean and worst-family attack robustness, and the edge-priority scenario assigns greater importance to prediction latency.

This analysis was conducted using the same normalized candidate-pool objectives used by the LARO score: development macro-F1, mean cross-family attack F1, worst-family attack F1, robustness stability, and prediction latency. The purpose of the sensitivity analysis was to determine whether the selected candidate remains stable under different optimization preferences.

4.10. Experimental outputs

The experimental pipeline produced several result tables used in the analysis. These include the dataset label taxonomy, capped training distribution, conventional held-out baseline results, leave-one-attack-family-out robustness results, candidate-pool evaluation results, LARO model-selection summary, conventional-versus-LARO comparison, weight-sensitivity analysis, ranking-instability analysis, and final cost-performance comparison. Together, these outputs provide the quantitative basis for evaluating whether LARO-IDS improves deployment-oriented model selection compared with conventional score-based selection.

5. Results and discussion

This section evaluates LARO-IDS as a robust, deployment-aware model-selection framework. The analysis proceeds from conventional held-out evaluation to cross-family robustness, candidate-pool optimization, LARO-based selection, weight-sensitivity analysis, ranking instability, and deployment-cost comparison. The objective is not merely to identify the classifier with the highest development performance score, but to determine whether the proposed multi-objective selection criterion identifies a more suitable IDS model for practical IoT and edge deployment.

5.1. Conventional held-out performance

The first step establishes a conventional performance baseline using the held-out evaluation setting. This evaluation reflects the standard model-selection practice in which candidate models are compared using metrics such as accuracy, macro-F1, weighted-F1, and balanced accuracy. Although this setting is useful, it does not directly measure robustness against unseen attack families or deployment cost.

Table 4 summarizes the conventional held-out performance of the baseline models on the capped CICIoT2023 held-out evaluation subset. The results show that RF achieved the highest development macro-F1 score, with a value of 0.8444, followed by XGBoost with 0.8176 and Extra Trees with 0.8099. XGBoost achieved the highest weighted-F1 score, with a value of 0.9606, indicating strong performance on the high-frequency classes. Logistic regression and LightGBM performed substantially worse under this setting and were therefore not retained for the later robustness-focused candidate analysis.

A key observation from **Table 4** is the gap between macro-F1 and weighted-F1. Although the

best weighted-F1 values exceeded 0.95 for RF and XGBoost, the macro-F1 values were lower. This indicates that performance on rare labels remains more difficult than performance on high-frequency attack classes. Therefore, aggregate accuracy or weighted-F1 alone is not sufficient for selecting a deployment-ready IDS model. This observation supports the use of macro-F1 and robustness-oriented metrics in the proposed LARO-IDS framework.

Table 4. Conventional held-out performance of baseline models

Model	Accuracy	Macro-F1	Weighted-F1	Balanced accuracy	Train time (s)	Latency (ms)
Random forest	0.958	0.844	0.957	0.812	172.64	0.0289
XGBoost	0.962	0.818	0.961	0.800	61.31	0.0125
Extremely randomized trees	0.947	0.810	0.946	0.788	102.04	0.0432
Logistic regression	0.665	0.480	0.652	0.534	1676.83	0.0009
LightGBM	0.145	0.073	0.102	0.124	243.09	0.1907

5.2. Robustness under unseen attack families

The conventional held-out results provide a useful performance baseline, but they do not answer whether the models can detect attack families that were not observed during training. This question is central to deployment-oriented IDS selection because IoT networks may encounter new attack behaviors after the model has been trained. The leave-one-attack-family-out protocol was therefore used to evaluate robustness under unseen attack-family conditions. **Table 5** summarizes the leave-one-attack-family-out binary detection results for the main baseline models.

The results show that robustness varies substantially across attack families. DDoS, DoS, and Mirai were detected with very high attack recall by all evaluated models. For example, XGBoost reached attack recall values close to 1.0000 for DDoS, DoS, and Mirai. However, BruteForce was consistently the most difficult held-out family. For the BruteForce attack family, XGBoost achieved an attack F1-score of 0.3144, while RF achieved a higher value of 0.3883. This difference provides further evidence that strong results in the conventional held-out setting do not guarantee consistent robust generalization when the model is tested against unseen attack families.

Table 5 also indicates that each model provides a different trade-off between robustness and computational cost. RF was generally more consistent across attack families and performed better on difficult cases such as BruteForce and Web_Malware attacks. XGBoost, however, provided a clear advantage in training and prediction latency, even though its robustness was weaker in some family-level tests. Extra Trees produced competitive recall in some attack-family settings, but its lower precision and higher false-positive behavior made its overall behavior less stable. Therefore, these findings show that model selection should not be based only on the highest single metric, but should consider the joint effect of predictive performance, robustness, and computational efficiency.

5.3. Optimization candidate pool

The baseline comparison demonstrates that model families differ in both conventional performance and robustness. However, robust model selection should not be limited to one configuration per model family. A model family may include configurations with different levels of complexity, latency, robustness, and model footprint. For this reason, an optimization candidate pool was constructed from multiple hyperparameter configurations of RF, XGBoost, and Extra Trees. **Table 6** summarizes the conventional held-out evaluation results of the optimization candidate pool.

Among the evaluated candidates, RF_03_regularized achieved the highest development macro-F1 score of 0.8522, followed closely by RF_01_fast with 0.8507. The difference between these two candidates was only 0.0015 in macro-F1. Although RF_03_regularized was the best candidate under conventional development macro-F1–based selection, RF_01_fast provided lower training time and prediction latency while

maintaining nearly identical predictive performance.

The results in **Table 6** suggest that the conventional best model may not be the most deployment-efficient candidate. However, conventional held-out performance alone cannot determine whether the lower-cost candidate is also robust to unseen attack families. Therefore, the same candidate pool was further evaluated using the leave-one-attack-family-out robustness protocol.

Table 5. Leave-one-attack-family-out robustness performance of baseline model families

Model	Family	Attack precision	Attack recall	F1	Balanced accuracy	Latency (ms)
Random forest	DDoS	0.981	1.000	0.990	0.901	0.0072
Random forest	DoS	0.944	1.000	0.971	0.900	0.0116
Random forest	Mirai	0.938	1.000	0.968	0.901	0.0121
Random forest	Recon	0.931	0.883	0.906	0.884	0.0187
Random forest	Spoofing	0.935	0.845	0.888	0.866	0.0204
Random forest	Web_Malware	0.390	0.998	0.561	0.902	0.0309
Random forest	BruteForce	0.242	0.981	0.388	0.893	0.0356
Extremely randomized trees	DDoS	0.975	1.000	0.987	0.867	0.0090
Extremely randomized trees	DoS	0.929	1.000	0.963	0.872	0.0141
Extremely randomized trees	Mirai	0.920	1.000	0.958	0.870	0.0159
Extremely randomized trees	Recon	0.917	0.849	0.882	0.857	0.0286
Extremely randomized trees	Spoofing	0.892	0.624	0.734	0.740	0.0283
Extremely randomized trees	Web_Malware	0.342	0.995	0.509	0.878	0.0439
Extremely randomized trees	BruteForce	0.200	0.981	0.333	0.866	0.0465
XGBoost	DDoS	0.983	1.000	0.992	0.913	0.0016
XGBoost	DoS	0.951	1.000	0.975	0.913	0.0025
XGBoost	Mirai	0.948	1.000	0.973	0.918	0.0027
XGBoost	Recon	0.937	0.761	0.840	0.835	0.0037
XGBoost	Spoofing	0.937	0.758	0.838	0.830	0.0035
XGBoost	Web_Malware	0.408	0.954	0.572	0.890	0.0078
XGBoost	BruteForce	0.205	0.677	0.314	0.755	0.0076

Abbreviations: DoS: Distributed Denial of Service; DDoS: Denial of Service.

5.4. Development performance versus cross-family robustness

The candidate-pool results raise an important model-selection question: do candidates with similar conventional performance also provide similar cross-family robustness? If not, conventional score-based ranking alone is insufficient for deployment-oriented IDS selection.

Figure 2 illustrates the relationship between development macro-F1 and mean cross-family attack F1 across the candidate IDS models. **Figure 2** shows that RF_01_fast and RF_03_regularized achieved very close development macro-F1 values. However, RF_01_fast achieved a slightly higher mean cross-family attack F1. This indicates that the candidate selected by conventional score-based selection is not necessarily the most robust candidate model under cross-family evaluation. This observation directly supports the need for a model-selection criterion that integrates both conventional performance and robustness signals.

5.5. Learned acquisition and reconstruction optimization-based robust model selection

After evaluating the candidate pool under both development and cross-family settings, the next step is to determine whether the proposed LARO objective leads to a different model-selection outcome than conventional score-based ranking. This analysis is central to the proposed framework because it treats candidate IDS models as optimization candidates that must balance predictive performance, cross-family robustness, worst-family performance, robustness stability, and deployment cost. **Table 7** summarizes the LARO-based model-selection results for the retained candidate pool.

The highest LARO score was achieved by RF_01_fast, with a score of 0.9687 under the balanced-LARO objective. RF_03_regularized ranked second, with a LARO score of 0.9355. Although RF_03_regularized had the highest development macro-F1, RF_01_fast achieved a slightly higher mean cross-family attack F1-score, with 0.8709 compared with 0.8663 for RF_03_regularized. RF_01_fast also provided lower prediction latency and computational cost.

Figure 3 presents the LARO candidate pool from a Pareto-oriented robustness–cost perspective. **Figure 3** illustrates how the retained candidates differ in mean cross-family robustness, prediction latency, and worst-family performance. The results in **Table 7** and **Figure 3** represent the core findings of the proposed framework. Conventional score-based selection chooses RF_03_regularized because it has the highest development macro-F1. In contrast, LARO selects RF_01_fast because it offers a better overall trade-off among conventional performance, cross-family robustness, and deployment cost. Thus, LARO does not simply reproduce the conventional ranking; instead, it modifies the model-selection decision based on robustness-aware and cost-aware objectives. This result also clarifies the role of the family-leakage-aware evaluation mechanism in the multi-objective selection process. The leave-one-attack-family-out protocol does not serve merely as a separate robustness test; it generates the cross-family robustness indicators that are included directly in the LARO objective, including mean cross-family F1-score, worst-family F1-score, and robustness variability. As a result, the family-level holdout evaluation affects the final optimization ranking and contributes to selecting RF_01_fast over RF_03_regularized, despite the latter having a slightly higher development macro-F1.

Table 6. Conventional held-out performance evaluation of the optimization candidate pool

Candidate model	Family	Accuracy	Macro-F1	Weighted-F1	Balanced accuracy	Training time (s)	Predicted (ms)
RF_03_regularized	Random forest	0.955	0.852	0.954	0.834	164.90	0.0259
RF_01_fast	Random forest	0.955	0.851	0.955	0.826	82.60	0.0135
RF_04_large	Random forest	0.958	0.845	0.957	0.812	258.49	0.0442
RF_02_balanced	Random forest	0.958	0.844	0.957	0.812	171.70	0.0302
XGB_04_regularized	XGBoost	0.961	0.819	0.960	0.801	65.25	0.0133
XGB_03_deep	XGBoost	0.962	0.819	0.961	0.800	96.10	0.0179
XGB_02_balanced	XGBoost	0.962	0.818	0.961	0.800	59.89	0.0124
XGB_01_fast	XGBoost	0.959	0.814	0.958	0.796	33.48	0.0083
ET_04_large	Extra Trees	0.948	0.811	0.947	0.789	148.82	0.0626
ET_02_balanced	Extra Trees	0.947	0.810	0.946	0.788	100.24	0.0412
ET_01_fast	ExtraTrees	0.848	0.704	0.850	0.753	33.80	0.0141
ET_03_regularized	Extra Trees	0.848	0.696	0.852	0.752	60.74	0.0246

Abbreviation: Extra Trees: Extremely randomized trees.

5.6. Direct comparison between conventional and learned acquisition and reconstruction optimization selection

The previous subsection shows that LARO and conventional score-based selection choose different candidates. The next step is to quantify whether the LARO-selected model sacrifices predictive performance or instead provides a more efficient robustness–cost trade-off. **Table 8** compares the conventionally selected candidate and the LARO-selected candidate.

The development macro-F1 of RF_03_regularized was 0.8522, while RF_01_fast achieved 0.8507. The difference was only -0.0015 in macro-F1, corresponding to a relative decrease of -0.18%. In contrast, RF_01_fast achieved a slightly higher mean cross-family attack F1-score, with an improvement of 0.0046. This indicates that the

LARO-selected model preserved nearly identical conventional predictive performance while slightly improving mean cross-family robustness.

Table 7. Summary of learned acquisition and reconstruction optimization-based candidate model selection

Candidate model	Machine learning model	Development macro-F1 score	Mean cross-family F1-score	Worst-family F1-score	Standard deviation of cross-family F1-score	Latency (ms)	LARO	Pareto-efficient status
RF_01_fast	RF	0.851	0.871	0.591	0.148	0.0135	0.968	Yes
RF_03_regularized	RF	0.852	0.866	0.599	0.146	0.0259	0.936	Yes
RF_02_balanced	RF	0.844	0.810	0.388	0.238	0.0302	0.467	No
RF_04_large	RF	0.845	0.809	0.387	0.238	0.0442	0.430	No
XGB_02_balanced	XGB	0.818	0.786	0.314	0.254	0.0124	0.244	Yes
XGB_03_deep	XGB	0.819	0.786	0.310	0.255	0.0179	0.237	Yes
XGB_04_regularized	XGB	0.819	0.784	0.314	0.256	0.0133	0.236	Yes
XGB_01_fast	XGB	0.814	0.779	0.288	0.264	0.0083	0.179	Yes
ET_02_balanced	ET	0.810	0.767	0.333	0.256	0.0412	0.086	No
ET_04_large	ET	0.811	0.766	0.331	0.256	0.0626	0.044	No

Abbreviations: ET: Extremely randomized trees; LARO: Learned acquisition and reconstruction optimization; RF: Random forest; XGB: XGBoost.

The deployment-cost differences in **Table 8** were more substantial. RF_01_fast reduced training time by 49.49%, prediction latency by 46.36%, and model size by 50.12% compared with RF_03_regularized. These reductions are important for IoT and edge deployment, where memory footprint and inference latency can affect deployment feasibility. Therefore, the LARO-selected model provides a more deployment-efficient alternative without sacrificing meaningful predictive performance.

5.7. Weight-sensitivity analysis of learned acquisition and reconstruction optimization selection

To examine whether the LARO decision is sensitive to a single weighting configuration, a weight-sensitivity analysis was conducted using four scenarios: balanced-LARO, performance-priority, robustness-priority, and edge-priority. **Table 9** summarizes the selected candidate under each scenario.

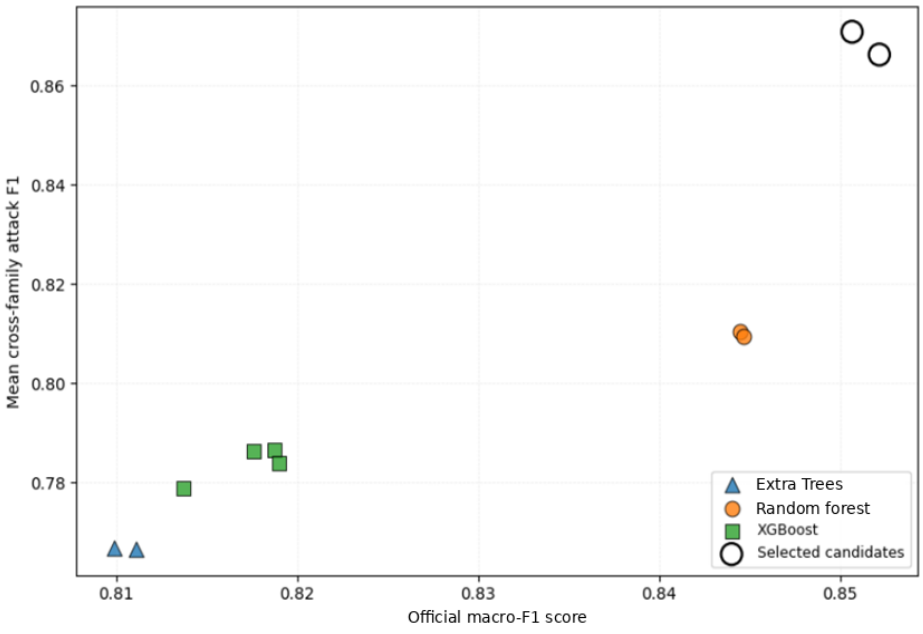

Figure 2. Relationship between development macro-F1 and mean cross-family attack F1-score across candidate intrusion detection system models
Abbreviation: Extra Trees: Extremely randomized trees.

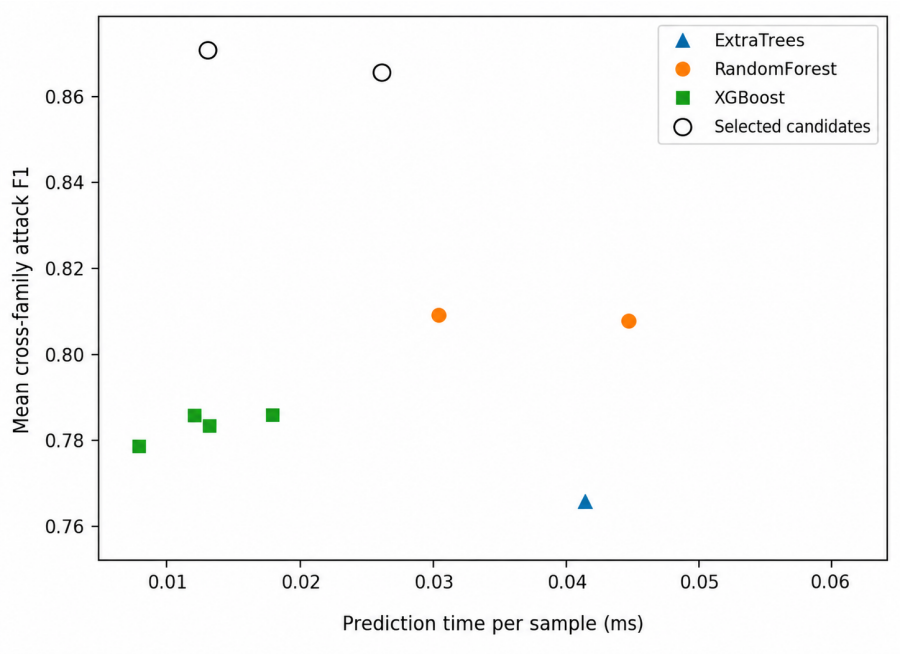

Figure 3. Pareto-oriented robustness–cost map of the learned acquisition and reconstruction optimization–intrusion detection system candidate pool
Abbreviation: Extra Trees: Extremely randomized trees.

Table 8. Conventional score-based selection versus LARO robust multi-objective selection

Strategy	Model	Development macro-F1 score	Cross-family F1-score	Worst-family F1-score	Training time (s)	Prediction latency (ms)	Size	LARO
Conventional score-based	RF_03_regularized	0.852	0.866	0.599	164.90	0.0259	233.91	0.936
LARO robust multi-objective	RF_01_fast	0.851	0.871	0.591	82.60	0.0135	116.68	0.968

Abbreviation: LARO: Learned acquisition and reconstruction optimization.

Table 9. Weight-sensitivity analysis of the LARO model-selection objective

Scenario	Model	Score	Development macro-F1 score	Cross-family F1-score	Worst-family F1-score	Standard deviation of cross-family F1-score	Latency (ms)
Balanced-LARO	RF_01_fast	0.969	0.851	0.871	0.591	0.148	0.0135
Performance-priority	RF_01_fast	0.968	0.851	0.871	0.591	0.148	0.0135
Robustness-priority	RF_01_fast	0.979	0.851	0.871	0.591	0.148	0.0135
Edge-priority	RF_01_fast	0.950	0.851	0.871	0.591	0.148	0.0135

Abbreviation: LARO: Learned acquisition and reconstruction optimization.

The results of the weight-sensitivity analysis showed that RF_01_fast was selected under all four weighting scenarios. This finding is important because it indicates that the LARO-based selection does not depend on one specific setting of the objective weights. Even when the optimization preferences were changed to give more emphasis to conventional performance, robustness, or edge-deployment cost, the selected model remained the same. Therefore, the selection of RF_01_fast can be interpreted as a stable outcome of the proposed selection objective, rather than being an artifact of a single weight configuration. This suggests that RF_01_fast provides a consistent trade-off among predictive performance, cross-family robustness, stability, and latency.

5.8. Deployment-cost gains of learned acquisition and reconstruction optimization selection

The main practical advantage of LARO is that it does not select the model solely based on a single predictive score. Instead, it attempts to identify a candidate model that has a more suitable profile for deployment. This is important in IoT and edge-based IDS applications because the selected model may be difficult to use in practice if it has high inference latency, high training cost, or a large model footprint. Therefore, the usefulness of the selected model should be judged not only by its predictive performance, but also by whether it can satisfy the operational requirements of edge-oriented deployment. **Table 10** presents the final cost-performance differences between the candidate selected by LARO and the candidate selected using the conventional score-based approach.

The LARO-selected model, RF_01_fast, provided clear reductions in several deployment-cost indicators when compared with the conventionally selected candidate. Specifically, it reduced the training time by 81.54 s, the prediction latency by 0.0118 ms per sample, and the model size by 117.22 MB. The compressed model size was also reduced from 233.91 MB for RF_03_regularized to 116.68 MB for RF_01_fast. These reductions show that the LARO-selected model not only maintains a competitive predictive profile, but also provides a lighter and more efficient deployment option. **Figure 4** illustrates the relative deployment-cost reductions achieved by the LARO-selected model.

Figure 4 summarizes the reductions in training time, prediction latency, and model size compared with the model selected by conventional score-based ranking. These results show that the main advantage of LARO is not a large increase in conventional predictive performance. Rather, LARO identifies a candidate that preserves almost the same predictive quality while substantially reducing deployment cost. This is consistent with the purpose of the proposed framework: selecting models that are robust and efficient, not merely models that maximize a single development score.

Table 10. Final cost-performance comparison between LARO and conventional selection

Metric	LARO vs. conventional
Development macro-F1 change	−0.0015
Development macro-F1 change (%)	−0.18
Weighted-F1 score change	+0.0002
Balanced accuracy change	−0.0083
Training time reduction (s)	81.54
Training time reduction (%)	49.49
Prediction latency reduction (ms/sample)	0.0118
Prediction latency reduction (%)	46.36
Model size reduction (MB)	117.22
Model size reduction (%)	50.12

Abbreviation: LARO: Learned acquisition and reconstruction optimization.

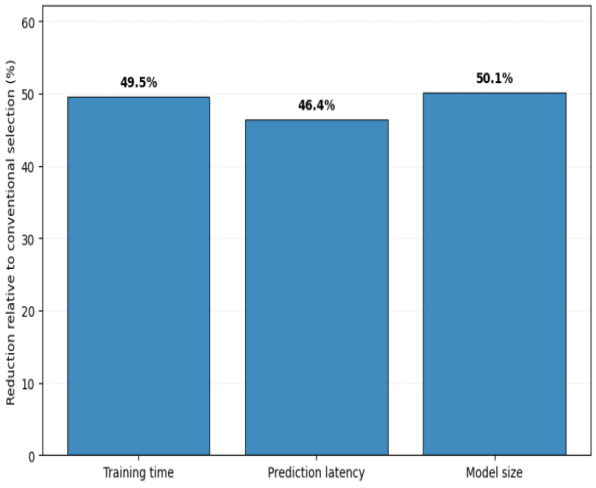

Figure 4. Relative deployment-cost reduction of the learned acquisition and reconstruction optimization–intrusion detection system-selected model

5.9. Ranking instability and the need for multi-objective selection

If the candidate rankings remain unchanged across all criteria, then a multi-objective selection framework would provide limited value. Therefore, the ranking-instability analysis was conducted to examine whether the preferred candidate changes

when models are ranked by conventional performance, cross-family robustness, worst-family robustness, prediction cost, and LARO score.

Table 11 summarizes the candidate ranking under conventional and LARO-based selection criteria.

The ranking analysis presented in **Table 11** confirms that the selected model depends on the criterion used. RF_03_regularized ranked first under development macro-F1, while RF_01_fast ranked first under the LARO score. RF_01_fast also ranked first in the mean cross-family F1 and second in worst-family F1. This indicates that the LARO selection was not arbitrary; it resulted from a consistent trade-off between robustness and cost. **Figure 5** illustrates how the ranking changes across selection criteria.

Figure 5 illustrates that the model ranking changes when moving from conventional performance-based selection to robustness-aware and cost-aware selection. XGBoost candidates ranked highly with respect to prediction cost because of their low latency, but their worst-family robustness values were lower than those achieved by the strongest RF candidates. Extra Trees candidates achieved lower LARO scores because they did not provide a superior trade-off in either robustness or cost. Therefore, the ranking-instability analysis demonstrates why single-metric model selection is insufficient for deployment-oriented IDS model selection.

5.10. Per-family behavior of the selected models

Although the aggregate LARO score is useful for model selection, a per-family analysis is needed to understand where the LARO-selected candidate improves and where it slightly loses. This analysis helps prevent overclaiming and shows that LARO should be interpreted as a trade-off mechanism rather than a guarantee of improvement for every individual attack family. **Table 12** compares RF_01_fast and RF_03_regularized across each held-out attack family.

RF_01_fast improved attack F1 on Spoofing, Recon, Web_Malware, and Mirai. The largest improvement occurred for Spoofing, where the attack F1 increased by 0.0254. For Recon, the attack F1 increased by 0.0116. These improvements are important because the Recon and Spoofing families were more difficult than DDoS, DoS, and Mirai under the leave-one-family-out setting.

RF_03_regularized performed slightly better on BruteForce, DDoS, and DoS. However, the differences for DDoS and DoS were nearly negligible, while the BruteForce advantage was 0.0084 attack F1 points. In contrast, RF_01_fast was faster across every held-out family. This finding confirms that the LARO-selected candidate provides a practical robustness-cost trade-off rather than a universal improvement on every individual family.

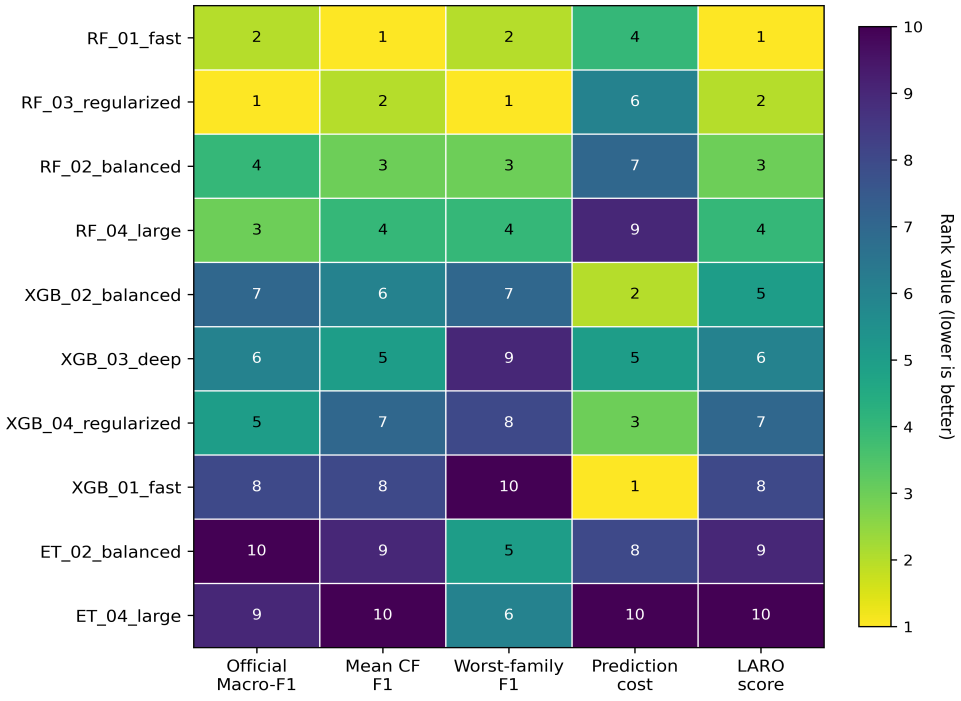

Figure 5. Ranking instability across conventional performance, robustness, cost, and the learned acquisition and reconstruction optimization (LARO) score

Table 11. Candidate ranking under conventional and LARO-based selection criteria

Candidate model	Family	Development macro-F1 score	Mean cross-family F1-score	Worst-family F1-score	Prediction latency cost	LARO	Performance shift
RF_01_fast	RF	2	1	2	4	1	+1
RF_03_regularized	RF	1	2	1	6	2	−1
RF_02_balanced	RF	4	3	3	7	3	+1
RF_04_large	RF	3	4	4	9	4	−1
XGB_02_balanced	XGBoost	7	6	7	2	5	+2
XGB_03_deep	XGBoost	6	5	9	5	6	0
XGB_04_regularized	XGBoost	5	7	8	3	7	−2
XGB_01_fast	XGBoost	8	8	10	1	8	0
ET_02_balanced	Extra Trees	10	9	5	8	9	+1
ET_04_large	Extra Trees	9	10	6	10	10	−1

Abbreviations: Extra Trees: Extremely randomized trees; LARO: Learned acquisition and reconstruction optimization; RF: Random forest.

Table 12. Per-family comparison between the LARO-selected and conventionally selected models

Attack family	F1-score of the LARO-selected model	F1-score of the conventionally selected model	Precision of the LARO-selected model	Precision of the conventionally selected model	Recall of the LARO-selected model	Recall of the conventionally selected model	Balanced accuracy of the LARO-selected model	Balanced accuracy of the conventionally selected model	Prediction latency of the LARO-selected model	Prediction latency of the conventionally selected model	Difference in F1-score between LARO-selected and conventionally selected models
Spoofing	0.876	0.850	0.963	0.966	0.803	0.760	0.872	0.854	0.0104	0.019	+0.025
Recon	0.892	0.880	0.964	0.965	0.829	0.808	0.887	0.879	0.0091	0.016	+0.012
Web_Malware	0.768	0.764	0.632	0.622	0.979	0.990	0.954	0.958	0.0227	0.042	+0.004
Mirai	0.987	0.987	0.974	0.974	1.000	1.000	0.960	0.960	0.0063	0.012	+0.000
DDoS	0.995	0.995	0.991	0.991	1.000	1.000	0.951	0.953	0.0038	0.006	−0.000
DoS	0.988	0.988	0.976	0.977	1.000	1.000	0.959	0.960	0.0066	0.012	−0.000
BruteForce	0.591	0.599	0.434	0.439	0.925	0.944	0.924	0.933	0.0222	0.045	−0.008

Abbreviations: DoS: Denial of Service; DDoS: Distributed Denial of Service; LARO: Learned acquisition and reconstruction optimization.

5.11. Discussion

The results support four main findings. First, conventional held-out performance does not fully characterize deployment-oriented IDS reliability. Models with strong development Macro-F1 may still show substantial variation across unseen attack families. This is especially evident for BruteForce and Web_Malware, which exhibited lower robustness scores than DDoS, DoS, and Mirai. Therefore, IoT intrusion detection evaluation should include robustness-oriented criteria rather than relying only on within-dataset predictive performance.

Second, model selection changes when robustness and deployment cost are included. Conventional score-based selection chose RF_03_regularized because it achieved the highest development macro-F1. LARO selected RF_01_fast because it provided nearly identical conventional performance, slightly stronger mean cross-family robustness, and substantially lower computational cost. This demonstrates that LARO is not simply an evaluation metric; it changes the decision rule used to select the final IDS model.

Third, the selected model should be interpreted as a trade-off solution. RF_01_fast did not dominate RF_03_regularized on every individual family. Instead, it provided a more balanced deployment-oriented profile: comparable predictive performance, lower training time, lower inference latency, smaller model size, and competitive cross-family robustness. This is precisely the type of decision that multi-objective optimization frameworks are designed to support. More specifically, RF_01_fast should not be interpreted as uniformly better than RF_03_regularized across all robustness dimensions. RF_03_regularized achieved a slightly higher worst-family F1, whereas RF_01_fast achieved slightly higher mean cross-family F1 and substantially lower deployment cost. Therefore, the LARO-selected model represents a robustness-cost trade-off under the predefined objective preferences, not a claim of dominance over the conventional candidate in every metric or for every attack family.

Fourth, the weight-sensitivity analysis strengthens the optimization interpretation of the proposed framework. RF_01_fast remained the selected candidate under balanced, performance-priority, robustness-priority, and edge-priority weighting scenarios. This indicates that the LARO decision was not an artifact of a single manually selected weight configuration but rather a stable outcome across different optimization preferences.

The comparison between RF_01_fast and RF_03_regularized is especially important because the difference in development macro-F1 was negligible, whereas the deployment-cost differences were substantial. A conventional selection rule would favor RF_03_regularized for a marginal gain in macro-F1 of approximately 0.0015. In contrast, LARO accepted this negligible performance difference in exchange for approximately 49.49% lower training time, 46.36% lower prediction latency, and 50.12% lower model size. This result highlights the practical value of robust multi-objective model selection in edge-oriented IDS deployment.

The per-family analysis also shows why the proposed framework should not be interpreted as a simple predictive-performance improvement method. LARO does not guarantee that the selected model will outperform the conventional candidate for every attack family. Instead, it selects the candidate that provides the most favorable overall balance across conventional performance, mean cross-family robustness, worst-case behavior, stability, and deployment cost. This distinction is important because real-world IDS deployment often requires selecting a model that is reliable and efficient under multiple constraints, rather than maximizing a single metric.

These findings are consistent with prior IDS optimization studies showing that intrusion detection design often involves competing objectives rather than a single predictive target. Earlier multi-objective IDS work used Pareto-oriented feature selection to balance feature quality and detection performance,[40] whereas recent IDS-focused multi-objective feature-selection studies further emphasized trade-offs among accuracy, feature-set size, error behavior, and runtime.[37] However, the present study differs from these works by applying multi-objective reasoning to the final model-selection decision under family-leakage-aware robustness evaluation conditions, rather than only to feature selection or detection-pipeline construction.

Despite the findings obtained in this study, several limitations remain that need to be considered. First, the experimental evaluation was conducted using the CICIoT2023 dataset only. Therefore, although the results provide useful evidence about the behavior of the proposed LARO-IDS framework, they should be further examined using other IoT and IIoT intrusion detection datasets to determine whether similar model-selection behavior can be observed under different data characteristics, traffic distributions, and attack scenarios. In particular, future validation on datasets such as TON_IoT and Edge-IIoTset would help examine

whether the LARO-IDS protocol generalizes to heterogeneous telemetry, IIoT, and edge-oriented traffic settings beyond CICIoT2023. Such evaluation would also support stronger cross-dataset generalization claims and clarify whether the selected robustness-cost trade-off remains stable across different benchmark designs. Second, the proposed LARO score is based on a weighted aggregation of normalized objectives. Although the weight-sensitivity analysis showed that the selected model remained stable across different weighting scenarios, the use of other weighting mechanisms may still be useful in future work, especially preference-learning approaches or adaptive weighting strategies. Third, the current implementation was limited to classical ensemble and boosting models applied to structured network-flow data. Future studies may therefore extend the framework to additional intrusion detection settings, including deep learning, online learning, and federated IDS environments.

Additional limitations are also related to sampling, uncertainty analysis, and deployment-cost measurement. The per-label capping strategy was used to reduce the dominance of high-frequency attack classes and to keep the experiments computationally feasible; however, this sampling choice may affect the observed performance and deployment-cost behavior compared with the original full class distribution. Future work should therefore compare capped and full-distribution settings to examine the sensitivity of LARO-IDS to the sampling strategy. In addition, BruteForce and Web_Malware remained among the most challenging held-out families, and a deeper error analysis using confusion matrices, false-positive rates, precision-recall curves, and representative failure cases would provide a more detailed explanation of these performance limitations. The current deployment-cost measurements were obtained on the stated workstation environment and should be interpreted as relative cost indicators rather than direct evidence of performance on real IoT or edge hardware. Future evaluation should include repeated timing measurements, warm-up runs, throughput, memory usage, energy consumption, and testing on representative edge devices. Finally, repeated training runs, confidence intervals, statistical significance analysis, temporal splits, cross-dataset transfer, open-set or zero-day attack evaluation, and adversarial robustness testing would further strengthen the empirical validation of the proposed framework.

Overall, the experimental results support the main premise of LARO-IDS. Robust IDS model selection should not be based solely on conventional development or held-out predictive performance. By integrating cross-family robustness, worst-family behavior, robustness stability, and latency-aware deployment cost into a unified selection objective, while using training time and model size for final deployment-cost comparison, LARO-IDS identifies models that are more suitable for practical IoT and edge intrusion detection deployment.

6. Conclusion

This study presented LARO-IDS, a family-leakage-aware robust multi-objective optimization framework for model selection in IoT intrusion detection. Unlike conventional IDS evaluation pipelines that select models mainly according to conventional held-out performance, the proposed framework integrates predictive performance, cross-family robustness, worst-family behavior, robustness stability, and latency-aware deployment cost into a unified model-selection process. This formulation shifts IDS model selection from a single-metric ranking problem to a deployment-oriented multi-objective decision problem. The experimental evaluation on the CICIoT2023 dataset demonstrated that conventional score-based selection and LARO-based selection can lead to different model choices. While the conventional criterion selected RF_03_regularized because it achieved the highest development macro-F1, LARO selected RF_01_fast due to its more favorable overall robustness–cost trade-off. The LARO-selected model preserved nearly identical conventional predictive performance, with only a 0.0015 decrease in macro-F1 difference, while achieving slightly higher mean cross-family attack F1 scores. More importantly, it reduced training time by 49.49%, prediction latency by 46.36%, and model size by 50.12% compared with the conventionally selected model.

The results also showed that candidate rankings vary across selection criteria. Models that perform best under development macro-F1 are not necessarily the best under cross-family robustness or latency-aware deployment cost. Similarly, the fastest models are not always the most robust. The weight-sensitivity analysis further showed that RF_01_fast remained the selected candidate under balanced, performance-priority, robustness-priority, and edge-priority weighting scenarios despite changes in objective preferences. This supports the stability of the proposed LARO decision rule and strengthens the multi-objective optimization-based interpretation of the framework. The per-family analysis further confirmed that LARO should be interpreted as a trade-off

selection framework rather than a method that guarantees improvement for every attack family.

The LARO-selected candidate improved attack F1 performance on several held-out families, including Spoofing, Recon, Web_Malware, and Mirai, while showing a small reduction for BruteForce. However, it remained substantially more efficient across all evaluated deployment-cost indicators. This behavior is consistent with the goal of multi-objective optimization, where the final model is selected based on the most favorable overall balance among competing objectives. Overall, the findings indicate that robust IDS model selection should not rely solely on conventional development-set performance. By incorporating family-leakage-aware cross-family robustness and latency-aware deployment cost into the unified scalar selection objective, while using training time and model size for final deployment-cost comparison, LARO-IDS identifies models that are more suitable for practical IoT and edge security environments.

Acknowledgments

None.

Funding

None.

Conflict of interest

The authors declare that they have no conflict of interest.

Author contributions

Conceptualization: Ameen Shaheen
Data curation: Ameen Shaheen
Formal analysis: Ameen Shaheen
Investigation: All authors
Methodology: Ameen Shaheen
Software: Ameen Shaheen
Validation: All authors
Writing - original draft: Ameen Shaheen
Writing - review & editing: All authors

Availability of data

The dataset (CICIoT2023) used in this study is publicly available and can be obtained from the Canadian Institute for Cybersecurity dataset repository.

AI tools statement

The authors used Grammarly and ChatGPT to improve the manuscript's grammar, clarity, and readability. These tools were used for language editing only and did not generate scientific content, results, interpretations, or conclusions. All authors confirm that AI tools were used only for language editing and that all final decisions and responsibility for the manuscript content remain with the authors.

References

1. Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans Emerg Telecommun Technol.* 2020;32(1). https://doi.org/10.1002/ett.4150

2. Gyamfi E, Jurcut A. Intrusion Detection in Internet of Things Systems: A Review on Design Approaches Leveraging Multi-Access Edge Computing, Machine Learning, and Datasets. *Sensors.* 2022;22(10):3744. https://doi.org/10.3390/s22103744

3. Alsaedi A, Moustafa N, Tari Z, Mahmood A, Anwar A. TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access.* 2020;8:165130-165150. https://doi.org/10.1109/ACCESS.2020.3022862

4. Ferrag MA, Friha O, Hamouda D, Maglaras L, Janicke H. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. *IEEE Access.* 2022;10:40281-40306. https://doi.org/10.1109/ACCESS.2022.3165809

5. Neto ECP, Dadkhah S, Ferreira R, Zohourian A, Lu R, Ghorbani AA. CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors.* 2023;23(13):5941. https://doi.org/10.3390/s23135941

6. Sommer R, Paxson V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In: *2010 IEEE Symposium on Security and Privacy.* IEEE; 2010:305-316. https://doi.org/10.1109/SP.2010.25

7. Apruzzese G, Pajola L, Conti M. The Cross-Evaluation of Machine Learning-Based Network Intrusion Detection Systems. *IEEE Trans Netw Serv Manag.* 2022;19(4):5152-5169. https://doi.org/10.1109/TNSM.2022.3157344

8. Cantone M, Marrocco C, Bria A. Machine Learning in Network Intrusion Detection: A Cross-Dataset Generalization Study. *IEEE Access.* 2024;12:144489-144508. https://doi.org/10.1109/ACCESS.2024.3472907

9. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* 2016;3(5):637-646. https://doi.org/10.1109/JIOT.2016.2579198

10. Deng S, Zhao H, Fang W, Yin J, Dustdar S, Zomaya AY. Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence. *IEEE Internet Things J.* 2020;7(8):7457-7469. https://doi.org/10.1109/JIOT.2020.2984887

11. Xu D, Li T, Li Y, *et al.* Edge Intelligence: Architectures, Challenges, and Applications. *arXiv.* Published online 2020. https://doi.org/10.48550/ARXIV.2003.12172

12. Deb K. Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction. In: *Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing.* Springer London; 2011:3-34. https://doi.org/10.1007/978-0-85729-652-8_1

13. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans Evol Comput.* 2002;6(2):182-197. https://doi.org/10.1109/4235.996017

14. Karl F, Pielok T, Moosbauer J, *et al.* Multi-Objective Hyperparameter Optimization in Machine Learning—An Overview. *ACM Trans Evol Learn Optim.* 2023;3(4):1-50. https://doi.org/10.1145/3610536

15. Morales-Hernández A, Van Nieuwenhuyse I, Rojas Gonzalez S. A survey on multi-objective hyperparameter optimization algorithms for machine learning. *Artif Intell Rev.* 2022;56(8):8043-8093. https://doi.org/10.1007/s10462-022-10359-2

16. Emmerich MTM, Deutz AH. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Nat Comput.* 2018;17(3):585-609. https://doi.org/10.1007/s11047-018-9685-y

17. Jin Y, Wang H, Chugh T, Guo D, Miettinen K. Data-Driven Evolutionary Optimization: An Overview and Case Studies. *IEEE Trans Evol Comput.* 2019;23(3):442-458. https://doi.org/10.1109/TEVC.2018.2869001

18. AbdulJawad M. Comparative Analysis of Machine Learning Algorithms for Intrusion Detection in IoT Networks. *IJASCA.* 2025. https://doi.org/10.15849/ijasca.251130.15

19. Breiman L. Random Forests. *Mach Learn.* 2001;45(1):5-32. https://doi.org/10.1023/A:1010933404324

20. Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. *Mach Learn.* 2006;63(1):3-42. https://doi.org/10.1007/s10994-006-6226-1

21. Chen T, Guestrin C. XGBoost. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM; 2016:785-794. https://doi.org/10.1145/2939672.2939785

22. Ke G, Meng Q, Finley T, *et al.* LightGBM: A highly efficient gradient boosting decision tree. *Adv Neural Inf Process Syst.* 2017;30.

23. Verma A, Ranga V. Machine Learning Based Intrusion Detection Systems for IoT Applications. *Wireless Pers Commun.* 2019;111(4):2287-2310. https://doi.org/10.1007/s11277-019-06986-8

24. Rahman MM, Shakil SA, Mustakim MR. A survey on intrusion detection system in IoT networks. *Cyber Secur Appl.* 2025;3:100082. https://doi.org/10.1016/j.csa.2024.100082

25. Ring M, Wunderlich S, Scheuring D, Landes D, Hotho A. A survey of network-based intrusion detection data sets. *Comput Secur.* 2019;86:147-167. https://doi.org/10.1016/j.cose.2019.06.005

26. Koroniotis N, Moustafa N, Sitnikova E, Turnbull B. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Gener Comput Syst.* 2019;100:779-796. https://doi.org/10.1016/j.future.2019.05.041

27. Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: *2015 Military Communications and Information Systems Conference (MilCIS).* IEEE; 2015:1-6. https://doi.org/10.1109/MilCIS.2015.7348942

28. Hnaif A, Jaber KM, Alia MA, Daghbosheh M. Parallel scalable approximate matching algorithm for network intrusion detection systems. *Int Arab J Inf Technol.* 2021;18(1):77-84. https://doi.org/10.34028/iajit/18/1/9

29. Tiwari RS, Lakshmi D, Das TK, Tripathy AK, Li KC. A lightweight optimized intrusion detection system using machine learning for edge-based IIoT security. *Telecommun Syst.* 2024;87(3):605-624. https://doi.org/10.1007/s11235-024-01200-y

30. Gao W, Wang M, Pei Y, Li F, Wang C. A Lightweight Multi-Classification Intrusion Detection Model for Edge IoT Networks. *Electronics.* 2026;15(5):938. https://doi.org/10.3390/electronics15050938

31. Choudhary T, Mishra V, Goswami A, Sarangapani J. A comprehensive survey on model compression and acceleration. *Artif Intell Rev.* 2020;53(7):5113-5155. https://doi.org/10.1007/s10462-020-09816-7

32. Di Mauro M, Galatro G, Fortino G, Liotta A. Supervised feature selection techniques in network intrusion detection: A critical review. *Eng Appl Artif Intell.* 2021;101:104216. https://doi.org/10.1016/j.engappai.2021.104216

33. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. *J Mach Learn Res.* 2012;13(10):281–305.

34. Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM; 2019:2623-2631. https://doi.org/10.1145/3292500.3330701

35. Borchert O, Salinas D, Flunkert V, Januschowski T, Günnemann S. Multi-Objective Model Selection for Time Series Forecasting. *arXiv*. Published online 2022. https://doi.org/10.48550/ARXIV.2202.08485

36. Williams P, Kendall W, Hooten M. Model Selection using Multi-Objective Optimization. *arXiv*. Published online 2018. https://doi.org/10.48550/arXiv.1810.10669

37. Sezgin A, Ulaş M, Boyacı A. Multi-Objective Feature Selection for Intrusion Detection Systems: A Comparative Analysis of Bio-Inspired Optimization Algorithms. *Sensors*. 2025;25(19):6099. https://doi.org/10.3390/s25196099

38. Marler RT, Arora JS. The weighted sum method for multi-objective optimization: new insights. *Struct Multidisc Optim*. 2009;41(6):853-862. https://doi.org/10.1007/s00158-009-0460-7

39. Pedregosa F, Varoquaux G, Gramfort A, *et al.* Scikit-learn: Machine learning in Python. *J Mach Learn Res*. 2011;12:825–2830.

40. Kamalov F, Moussa S, Zgheib R, Mashaal O. Feature selection for intrusion detection systems. In: *2020 13th International Symposium on Computational Intelligence and Design (ISCID)*. IEEE; 2020:265-269. https://doi.org/10.1109/ISCID51228.2020.00065

Ameen Shaheen *received his Ph.D. degree in Computer Science from the University of Jordan in 2018. He is affiliated with the Department of Software Engineering, Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan, Amman, Jordan. His research activities include machine learning, optimization algorithms, software engineering, artificial intelligence, and their applications in different computational and engineering problems. He has contributed to a number of research studies in these areas and has participated in collaborative research involving intelligent and data-driven systems.*
https://orcid.org/0009-0003-3120-1892

Wael Alzyadat *is an Associate Professor and Head of the Department of Software Engineering at Al-Zaytoonah University of Jordan, Amman, Jordan. His research interests include software engineering, intelligent systems, big data, streaming data, machine learning, and related data-driven applications. He has published research in several areas involving software analysis, artificial intelligence, data processing, and intelligent computing systems, and has participated in research addressing both theoretical and applied computing problems.*
https://orcid.org/0000-0002-0068-1526

Aysh Alhroob *is a Full Professor of Software Engineering and Dean of the Faculty of Science and Information Technology at Al-Zaytoonah University of Jordan, Amman, Jordan. He received his Ph.D. degree in Software Engineering from the University of Bradford, United Kingdom, in 2010. His research interests include software testing, software requirements, natural language processing, big data, data science, data analysis, computer algorithms, and artificial intelligence techniques. He has extensive academic, research, supervision, and administrative experience in software engineering and information technology.*
https://orcid.org/0000-0002-4653-7386

An International Journal of Optimization and Control: Theories and Applications (https://accscience.com/journal/ijocta)