EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Reference-Assisted Learner-Aligned Distributed Stacking for Edge Intrusion Detection in the Industrial Internet of Things

Jing Li[1], Shuhao Shen[1,*], Kangrui Xu[2], Lin Cui[1]

[1]School of Information Engineering, Suzhou University, Suzhou, Anhui, China
[2]Ningbo Yunhe Technology Co., Ltd., Ningbo, Zhejiang, China

Abstract

Networked manufacturing couples sensors, programmable logic controllers, and industrial gateways to production that cannot simply be paused. An intrusion detector in this setting must control false alarms and edge-resource use as well as detect attacks. Industrial Internet of Things (IIoT) edge nodes add three practical constraints: training traffic cannot be centralized, local statistical distributions differ, and the deployed models may be structurally heterogeneous. We address this setting with Reference-Assisted Learner-Aligned Distributed Stacking (RA-LADS). It evaluates node models on mutually exclusive reference pools, groups LightGBM and XGBoost responses by learner family, and represents each family by its mean, standard deviation, and five quantiles in a fixed, permutation-invariant $7M$-dimensional vector. Across 20 matched partitioning and training seeds, the full 14-dimensional summary attained a mean F1 of 0.984514. Its gain over a global 7-dimensional summary without family semantics was 0.000756 (95% CI: 0.000559–0.000962; Holm-adjusted $p = 1.08 \times 10^{-5}$). By comparison, the difference from node-wise raw prediction concatenation was −0.000009, with an interval spanning zero. Random balanced grouping was not significantly different from true learner-family grouping after multiplicity correction; family identity is therefore a useful grouping prior, but not the only one. Mean F1 scores for a compact reference-pool LightGBM, a 41-parameter Tiny Deep Sets model, and a heterogeneous one-model-per-node summary were 0.983887, 0.983659, and 0.983865. The full method delivered small yet reproducible paired gains over all three controls. At fixed false-alarm-rate (FAR) budgets, no stable advantage over a single LightGBM appeared between 0.1% and 2% FAR; at 5% FAR, F1 increased by 0.000417 (Holm-adjusted $p = 0.017$). Five-seed retraining on two external NetFlow datasets did not show a general advantage. The claim is therefore limited to settings in which node responses carry exploitable distributional structure. Within that boundary, RA-LADS offers a lightweight interface for heterogeneous tree collaboration when labeled reference data and explicit alert operating points are available, although system cost still grows linearly.

Keywords: Industrial Internet of Things, edge intrusion detection, distributed stacking, non-IID data, reference data, model fusion

Received on 21 July 2026; accepted on 12 August 2026; published on 31 August 2026

doi:10.4108/eetsis.14107

* Corresponding author. E-mail: shenshuhao@ahszu.edu.cn

1. Introduction

The Industrial Internet links sensors, programmable logic controllers, industrial gateways, and cloud platforms across production systems that operate continuously. Within networked manufacturing, these connections span production cells, equipment generations, and line-side services, so defensive interventions must preserve continuous operation while respecting site-specific alert and maintenance capacity. The same links can carry scanning, denial-of-service, injection, credential, and ransomware activity toward control domains. A plant with several production lines and gateways cannot judge a detector by recall alone: missed attacks, false-alarm workload, storage, and inference delay all matter to production continuity. This setting motivates the design constraints used here. It does not amount to validation

validation in an operational factory, because every experiment uses a public NetFlow benchmark. Accordingly, the present evidence is manufacturing-motivated benchmark validation rather than end-to-end factory validation: traffic acquisition, line-side feature extraction, sustained gateway load, update rollout, and operator response were not jointly exercised. Public NetFlow datasets and standardized feature extraction make controlled comparisons possible[1, 2]; work on online and embedded deployment likewise shows why detection quality must be read alongside streaming performance, model size, and endpoint execution cost[3–5].

Tree models capture nonlinear structure in tabular traffic, but centralized training pools the records held by different nodes. Federated learning instead exchanges model updates. FedAvg, FedProx, SCAFFOLD, and FedNova address communication efficiency, objective heterogeneity, client drift, and unequal local updating, respectively[6–9]. Their aggregation rules usually assume that client parameters align element by element. That assumption breaks when gateways use different tree learners, such as LightGBM and XGBoost, whose topologies and parameter meanings have no shared coordinate system. Heterogeneous edge collaboration is consequently a different problem from homogeneous federated optimization[10–12].

Prediction space offers another route. Stacked generalization combines base learners through a meta-learner[13], and random forests, XGBoost, and LightGBM bring different inductive biases to tabular traffic classification[14–16]. Responses from nodes that use the same learner family form an unordered set, so changing node order should not change the set representation. Deep Sets and Set Transformer obtain this invariance through symmetric pooling and attention-based interaction[17, 18]. Here the question is deliberately narrower than learning a high-capacity set encoder. We ask whether family grouping, followed by location, dispersion, and order statistics, retains more useful information than a mean while keeping the interface fixed in dimension.

This question leads to RA-LADS. Nodes train heterogeneous tree learners locally. The coordinator uses mutually exclusive reference subsets to lock features, construct the response representation, train the meta-model, and choose a threshold; gateways then receive the frozen bundle. The contributions are:

- We establish a heterogeneous tree-response interface based on common functional coordinates, an interpretable grouping prior, and symmetric order structure. For each registered family, a weighted mean, weighted standard deviation, and five unweighted linearly interpolated quantiles form a $7M$-dimensional meta-feature. The interface is invariant to node permutations and fixed in input schema without requiring a mapping between tree parameters; changing the node count still requires retraining the meta-model and recalibrating the threshold.

- We design matched-base-model controls that remove family semantics, introduce random balanced groups, concatenate raw node responses, sort responses within families, train single models on the reference pool, or use a 41-parameter Tiny Deep Sets model. These controls distinguish gains due to structured grouping, node responses, and second-stage capacity. A one-model-per-node setting additionally evaluates genuine family heterogeneity and model absence, and is explicitly separated from the default two-family-per-node configuration.

- We jointly delimit detection gains, alert operating points, and system growth using mutually exclusive data roles, 20-seed statistics, fixed FAR budgets of 0.1%–5%, attack-prior adjustment, three forms of non-IID data, five-seed full retraining at $K = 2/5/10/20$, reference-pool size, label noise, node dropout, and five-seed reproduction on two external datasets.

1.1. Related Work and Research Positioning

Stacking, Set Representations, and Probabilistic Operating Points. Stacking learns a second-stage mapping from base-model outputs to labels on independent samples. Both base-model diversity and the separation of meta-learning data affect its validity. Three tasks must remain distinct in a probabilistic IDS: the meta-model captures complementarity, threshold selection fixes the operating point, and calibration measures agreement between predicted probabilities and observed frequencies[19]. RA-LADS therefore uses separate meta-learning and threshold-tuning pools, then reports the Brier score and expected calibration error (ECE) alongside detection metrics.

XGBoost, LightGBM, and random forests differ in tree growth, regularization, and ensemble construction. They therefore provide meaningful heterogeneity at the nodes. Within one learner family, however, node identity usually carries no semantic meaning. Direct concatenation keeps every response but binds the meta-model to an arbitrary ordering. A learnable set model removes that dependence at the price of additional capacity and training. RA-LADS uses explicit symmetric statistics instead, which lets us test complete order retention separately from the benefit of a fixed-dimensional interface.

Statistical Heterogeneity, Model Heterogeneity, and Knowledge Transfer. Non-IID data can differ in label proportion, client size, feature distribution, or local optimization trajectory. MOON, Ditto, FedBN, and FedLC respond through representation

consistency, personalization, local normalization statistics, and label-distribution calibration[20–23]. Model and system heterogeneity call for other devices: HeteroFL assigns subnetworks of different complexity from a shared network, whereas FedProto exchanges class prototypes rather than gradients[24, 25]. FLTrust anchors robust aggregation in trusted root data[26]. Shared substructures, semantic prototypes, and reference anchors can all supply coordinates for collaboration. Their assumptions about models, communication, and data nevertheless differ from the fixed statistical interface studied here for arbitrary tree-family responses.

FedDF distils outputs from heterogeneous client models with unlabeled data, while FedGen transfers user knowledge through a data-free generator[27, 28]. Ensemble knowledge distillation has also been applied to an IDS with heterogeneity in both IoT client models and data[29]. Such methods avoid parameter alignment by working in prediction or knowledge space. RA-LADS also compares functional responses on common inputs, but it does not train a shared student or send learned knowledge back to clients. The coordinator keeps the tree models and fits a lightweight meta-classifier on labeled reference data. Our question is not whether prediction-space fusion dominates federated optimization. It is whether a transparent, fixed-dimensional summary can replace node-bound raw prediction concatenation when reference labels are available.

Federated and Edge Intrusion Detection. Federated deep models and communication-efficient anomaly detectors have been studied in industrial cyber-physical and IoT settings[30, 31]. Most train homogeneous neural networks and exchange updates over multiple rounds. Our topology instead collects models once, learns from reference data at the coordinator, and replicates the complete tree bundle. Model capacity, communication lifecycle, and server-side data assumptions are not matched across these paradigms. The FedAvg and FedProx results should thus be read as capacity-disclosed reference baselines, not as evidence that prediction-space methods are generally superior.

In RA-LADS, raw training records stay at the edge nodes while the fitted tree models are uploaded. The coordinator learns a meta-model from labeled reference data, and each gateway receives the full bundle for local inference. One collection and one bundle-distribution phase take the place of repeated parameter averaging. What distinguishes this design from parameter-federated methods is the object being shared, the need for reference data, and the communication lifecycle—not any single byte-count comparison.

To avoid cross-paradigm comparisons based on a single performance or communication metric, Table 1 contrasts representative approaches and RA-LADS in terms of raw-data movement, central reference data, client models, training communication, online communication, and deployment boundaries.

2. Problem Formulation, System Architecture, and the RA-LADS Method

2.1. Data Roles and Learning Task

Write the official training and sealed test sets as $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$. A fixed random seed divides the training data into four non-overlapping subsets, so feature selection, meta-learning, and threshold selection do not reuse the same records:

$$\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{edge}} \dot{\cup} \mathcal{D}_{\text{feature}} \dot{\cup} \mathcal{D}_{\text{meta}} \dot{\cup} \mathcal{D}_{\text{threshold}}. \quad (1)$$

The default proportions are 85%/5%/5%/5%. Here, $\mathcal{D}_{\text{feature}}$ determines the input features, $\mathcal{D}_{\text{meta}}$ trains the second-stage classifier, and $\mathcal{D}_{\text{threshold}}$ optimizes the final operating threshold. The edge pool is further partitioned among K nodes:

$$\mathcal{D}_{\text{edge}} = \dot{\bigcup}_{i=1}^{K} \mathcal{D}_i, \quad \mathcal{D}_i = \mathcal{D}_i^{\text{fit}} \dot{\cup} \mathcal{D}_i^{\text{val}}. \quad (2)$$

Each node reserves 1/17 of its edge allocation for local validation. Consequently, node fitting data account for approximately 80% of the official training set and local validation data for approximately 5%. The test set is accessed only after the features, base models, meta-representation, weights, and thresholds have been fixed.

In the intended manufacturing topology, each node represents the traffic view available to a production cell, line-side control segment, or industrial gateway, whereas the three labeled reference pools are governed centrally and remain disjoint from node fitting records. This mapping defines the deployment context rather than the provenance of the benchmark: NF-ToN-IoT-v2 provides no verified plant, production-line, or device identities. The synthetic partitions evaluated below therefore isolate statistical heterogeneity that can challenge manufacturing segments; they do not measure differences between operating factories.

For a traffic sample $x \in \mathbb{R}^d$ at node i, local learner $f_{i,m}$ outputs an attack probability, where $m \in \{1, \ldots, M\}$ indexes the learner family. The objective is to use reference data to learn the mapping

$$\mathcal{F} : \{f_{i,m}(x)\}_{i=1,m=1}^{K,M} \longrightarrow p(y = 1 \mid x), \quad (3)$$

without pooling the raw records in $\mathcal{D}_i$, and to produce a binary decision at the edge gateway using threshold τ. The central difficulty is that tree models from different families have no parameter coordinates that can be averaged directly, whereas an unstructured average over all probabilities removes information about learner family and inter-node disagreement.

Table 1. Differences in assumptions between representative collaborative-learning paradigms and the proposed method. "Online communication" denotes per-flow network exchange after deployment and excludes training and model updates

Paradigm	Raw data moved	Central reference data	Client model	Training comm.	Online comm.	Primary heterog.	Deployment/ boundary
Centralized trees	Yes	N/A	N/A	One-time data pooling	None	Unified training distribution	Central training; one model can be distributed
FedAvg/ FedProx	No	No	Same parameters	Multiple rounds	Potentially zero after deploy.	Label, quantity, system	Parameter averaging; identical topology required
SCAFFOLD/ FedNova	No	No	Same parameters	Multiple rounds; normalized control	Potentially zero after deploy.	Client drift, local steps	Optimization-level correction
Personalized or calibrated	No	Usually no	Shared core; local state	Multiple rounds	Local inference	Feature/label personaliza- tion	Clients retain personalized state
Distillation/ knowledge transfer	No	Unlabeled proxy or generator	May differ	Multiple rounds; periodic distill.	Local student inference	Model/data heterog.	Unified student or returned knowledge
RA-LADS (this work)	No	Yes; disjoint training roles	Different tree families	Model collection; bundle distribution	Zero after replication	Label, quantity, feature partition	Alignment; $K \times M$ base models per gateway

2.2. Model-Level Collaboration Architecture

The architecture has three roles: edge training nodes, a coordinator, and deployment gateways (Fig. 1). A node fits LightGBM and XGBoost locally, then uploads both serialized models and quality statistics from its validation split. At the coordinator, every base model is evaluated on identical reference inputs. The resulting predictions are grouped by learner family, converted into cross-node distribution features, and passed to a logistic-regression meta-learner. Base models, meta-model, feature list, and threshold are frozen together and replicated to the gateways for local inference.

In a networked factory, the coordinator could reside in the site security or plant-edge management domain, while the frozen bundle runs beside each monitored production segment without a per-flow round trip to that coordinator. This topology reduces online dependence on site-wide connectivity, but an approved model revision would still require controlled validation, distribution, version tracking, and rollback across the gateway fleet.

Reference samples supply a common predictive coordinate system without forcing nodes to share a tree structure. The coordinator handles functional responses to identical inputs rather than averaging parameters. It also retains learner type, which ordinary probability averaging discards, and uses distribution statistics to describe agreement across nodes. This flexibility is not free: deployment cost includes model collection, distribution of the full bundle, and storage and execution of $K \times M$ base models at every gateway.

2.3. Trust Model and Security Boundary

The experimental protocol assumes honest participants. Nodes use locked code for training and upload, the coordinator executes reference inference, meta-learning, and threshold selection as specified, and model-distribution links provide authentication and integrity. These assumptions isolate the representation mechanism; they are not a formal security guarantee. Keeping raw edge records local reduces direct data aggregation, but model parameters and prediction responses can still create privacy or integrity risks. The Discussion places those risks within the scope of the present evidence.

The threat model therefore excludes malicious clients, a compromised coordinator, poisoned reference pools, model theft, and transport-layer attacks. RA-LADS implements neither secure aggregation nor differential privacy, remote attestation, or Byzantine robustness; uploading a model is not described as privacy preserving. At minimum, deployment would add model signatures, node authentication, version control for reference data, and isolation of anomalous models. Those safeguards sit outside the response representation and need their own adversarial evaluation.

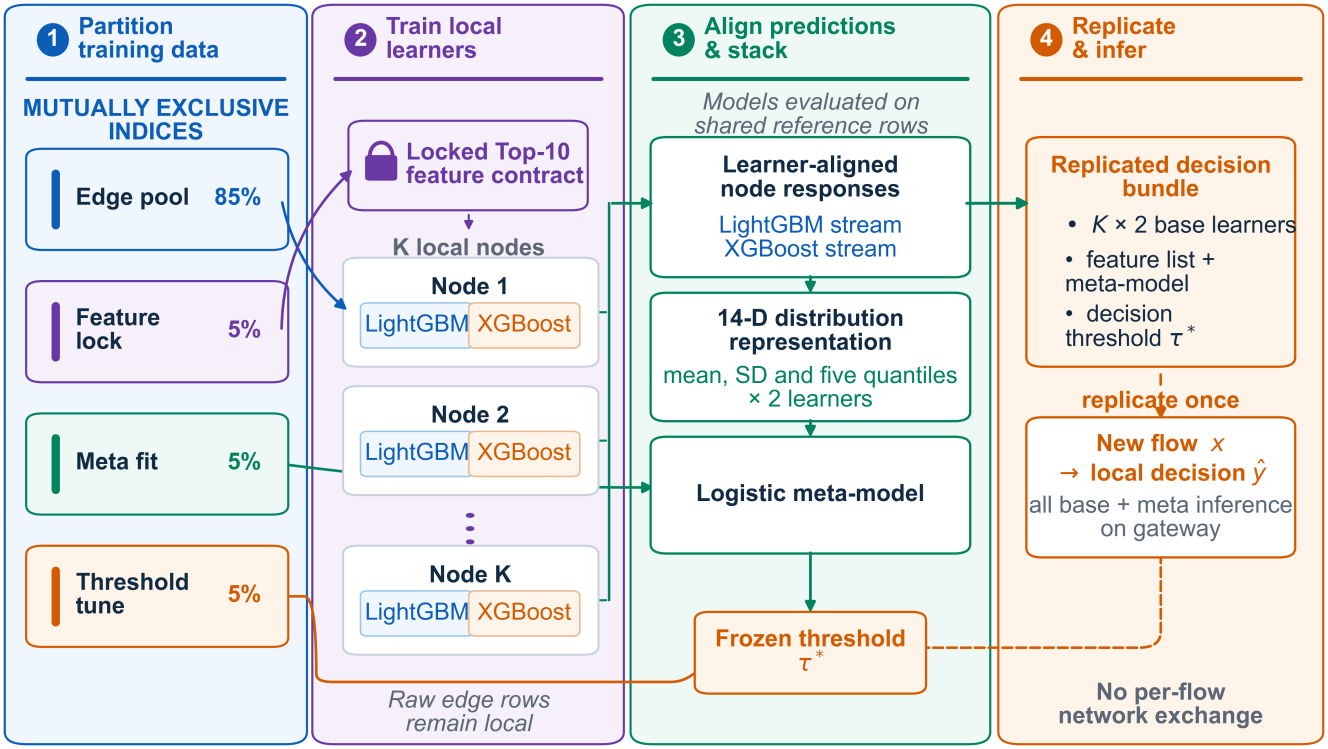

Figure 1. Training and deployment workflow of RA-LADS. Mutually exclusive reference pools support feature locking, meta-learning, and threshold optimization. Node models are aligned by learner family to form a cross-node distribution representation, and the complete model bundle is deployed to gateways for local inference

2.4. Reference-Based Feature Locking and Node-Level Base Learning

Removing source and destination IPv4 addresses leaves 41 numerical flow features. We apply a deterministic signed $\log(1+|x|)$ transform to two extreme per-second rate fields and retain the original scale elsewhere. Only $\mathcal{D}_{\text{feature}}$ enters feature selection. From it, 400,000 stratified rows are ranked by LightGBM split-count importance and evaluated at $k \in \{10, 15, 20, 30, 41\}$. We choose the smallest candidate whose validation F1 lies within 0.003 of the optimum. The selected variables are L7 protocol, destination port, flow duration, longest and shortest flow packets, maximum inbound TCP window, TCP flags, the number of packets no larger than 128 bytes, source port, and client TCP flags. The rule trades a prespecified amount of detection performance for lower node-model complexity.

Each node trains two compact tree-model families in the same 10-dimensional feature space. LightGBM uses 48 trees, 15 leaves, a maximum depth of 5, a learning rate of 0.08, a minimum of 100 samples per leaf, subsampling and column-sampling rates of 0.85, $L_2 = 1$, and balanced class weights. XGBoost uses 40 trees, a maximum depth of 4, a learning rate of 0.08, a minimum child weight of 20, subsampling and column-sampling rates of 0.85, $L_2 = 1$, the histogram algorithm, and 63 histogram bins. Each model uses at most two threads. If a node fitting set contains only one class, the corresponding constant-probability model is used to preserve a consistent inference interface.

2.5. Learner-Family-Aligned Distribution Representation

Let $f_{i,m}(x) \in [0,1]$ denote the attack probability assigned to sample x by learner family m at node i. Family alignment first organizes models with the same inductive bias into the set $\{f_{i,m}\}_{i=1}^{K}$. The simplest representation is the weighted within-family mean:

$$z_m^{\text{mean}}(x) = \sum_{i=1}^{K} w_i f_{i,m}(x), \quad \sum_i w_i = 1, \quad w_i \geq 0. \quad (4)$$

Family identity is retained, but all responses within that family collapse to one location statistic. Equal means can arise from stable consensus or from predictions that disagree and offset one another. The mean alone cannot tell these cases apart.

To characterize the cross-node response structure, RA-LADS jointly computes, within each family, the weighted mean μ_m, weighted population standard deviation σ_m, minimum, 25th percentile, median, 75th percentile, and

maximum:

$$\mu_m(x) = \sum_i w_i f_{i,m}(x), \tag{5}$$

$$\sigma_m(x) = \left[\sum_i w_i \{f_{i,m}(x) - \mu_m(x)\}^2\right]^{1/2}, \tag{6}$$

$$\boldsymbol{z}_m(x) = [\mu_m, \sigma_m, q_{0,m}, q_{.25,m}, q_{.50,m}, q_{.75,m}, q_{1,m}]. \tag{7}$$

Normalized node weights enter the mean and standard deviation; the five quantiles weight all available models in a family equally. We use NumPy's linear interpolation, which combines the neighboring sorted values around index $(n-1)q$. Concatenating the family blocks gives $\boldsymbol{z}(x) = [\boldsymbol{z}_1(x), \ldots, \boldsymbol{z}_M(x)] \in \mathbb{R}^{7M}$. With $M = 2$, each flow is encoded by 14 values. Adding a node changes their estimates, not the meta-learner's input schema. The summary is permutation invariant within a family, unlike node-wise raw concatenation, and its dimension does not grow with K, unlike sorted within-family concatenation. One special case matters: at $K = 5$, $q_0, q_{.25}, q_{.50}, q_{.75}, q_1$ are exactly the five sorted responses. Quantiles-only and within-family sorting are therefore identical in the default configuration.

Two matched controls separate family alignment from dimensionality. The first pools all KM predictions into one 7-dimensional summary and discards family semantics. The second fixes a label-agnostic, balanced two-group partition once per seed, then computes the same 7 statistics for each group. Genuine model heterogeneity is examined by retaining one prespecified learner per node and normalizing weights over the family members that remain. A 14-dimensional input is still available while every registered family has at least one model. If a family disappears, however, its empirical 7-dimensional block is undefined. We do not fill it with zeros. Instead, the system switches to the corresponding single-family 7-dimensional mode and refits both meta-model and threshold. Temporary node dropout likewise requires re-estimation over the surviving models, so these tests do not establish calibration-free hot-plug tolerance.

The logistic-regression meta-learner g_θ is fitted on $\mathcal{D}_{\text{meta}}$ using balanced class weights, a fixed random seed, and at most 1000 iterations:

$$p(x) = \sigma(\theta_0 + \boldsymbol{\theta}^\top \boldsymbol{z}(x)), \qquad \hat{y}(x) = \mathbb{I}[p(x) \geq \tau]. \tag{8}$$

A linear meta-model keeps representation gains separate from second-stage classifier capacity. The comparison includes the 2-dimensional family mean, a 4-dimensional mean-plus-standard-deviation vector, 10 quantiles, the full 14-dimensional summary, node-bound $K \times M$ raw concatenation, and sorted within-family concatenation. Every variant reuses the same base learners and the same meta-learning and threshold-tuning pools. Tiny Deep Sets tests whether the linear stage is too restrictive. Its input pairs each attack probability with a one-hot learner-family code; an 8-unit shared tanh encoder, set-mean pooling, and a logistic output layer yield 41 trainable parameters. Training lasts 6 epochs with batches of 8192. Neither architecture nor hyperparameters are chosen on the threshold pool or test set.

2.6. Node-Reliability Weighting

For node i, the local validation set yields sample size n_i, F1, recall R_i, false alarm rate FAR_i, log loss, and class coverage $C_i \in \{0.5, 1\}$. To examine the effect of node reliability on within-family aggregation, we compare equal weighting, sample-size weighting, validation-F1 weighting, inverse-log-loss weighting, and composite-quality weighting. The composite score is

$$s_i = 0.40F1_i + 0.25R_i + 0.15\frac{\log(1+n_i)}{\max_j \log(1+n_j)} + 0.15C_i - 0.20FAR_i, \tag{9}$$

and is normalized as $w_i = \max(s_i, 10^{-6}) / \sum_j \max(s_j, 10^{-6})$. These rules access only node validation statistics, and all coefficients are fixed before the sealed test is evaluated. The equally weighted distribution representation is the primary configuration; the remaining rules test whether explicit reliability weighting contributes information beyond meta-learning.

2.7. Threshold Optimization and Probability-Quality Evaluation

All probability-output models share $\mathcal{D}_{\text{threshold}}$ and the candidate set $\{0, 0.001, \ldots, 1\}$. The optimal threshold is defined as

$$\tau^* = \arg\max_\tau F1(\tau). \tag{10}$$

When multiple thresholds yield the same F1, the threshold with the lower FAR is selected, followed by the numerically larger threshold if a tie remains. The implementation uses sorting and prefix counts, with time complexity $O(n \log n + B \log n)$, where $B = 1001$ is the number of candidates. In addition to F1, precision, recall, and FAR, evaluation includes ROC-AUC, AUPRC, FAR@TPR=95%, the Brier score, and 15-bin ECE, covering operating-point performance, ranking ability, and probability quality.

To prevent the maximum-F1 threshold from obscuring alert costs, we additionally define five FAR budgets, $\beta \in \{0.001, 0.005, 0.01, 0.02, 0.05\}$, and use only $\mathcal{D}_{\text{threshold}}$ to select among 10,001 evenly spaced candidates:

$$\tau_\beta = \arg\max_{\tau:\, FAR(\tau) \leq \beta} Recall(\tau). \tag{11}$$

If recall is tied, the threshold with higher F1 is selected, followed by the numerically larger threshold. The test

set reports only the actual FAR, recall, and F1 under the frozen threshold. Because the training-side threshold pool and sealed-test distribution are not identical, the test FAR may deviate slightly from the target budget.

In a manufacturing security operations workflow, β is therefore a capacity constraint rather than only a statistical tuning target: it must be chosen against benign traffic volume, the available triage and engineering capacity, and the operational cost of investigating a production asset. The benchmark analysis exposes that trade-off but does not prescribe an acceptable plant threshold.

2.8. Complexity and Lifecycle Communication

Let $S_{i,m}$ be the serialized size of a base model and S_g the size of the meta-model. The complete deployment-bundle size at each gateway is

$$S_{\text{pkg}} = \sum_{i=1}^{K} \sum_{m=1}^{M} S_{i,m} + S_g. \tag{12}$$

Although the representation has only $7M$ entries, computation, base-model storage, and inference count all scale linearly with K. A fixed input schema is not a node-count-independent system. Model collection uploads approximately $\sum_{i,m} S_{i,m}$, and copying the bundle to K gateways costs approximately KS_{pkg}. Five float64 quality fields add 40 bytes per node. Once deployed, a gateway predicts locally for each flow. Over U model updates, collection and distribution traffic each accumulate in proportion to U.

Table 2 gives the complete workflow from in-training data-role assignment to sealed-test evaluation, ensuring that the feature set, meta-model, and thresholds are fixed before test access.

3. Experimental Design

3.1. Datasets, Integrity, and Preprocessing

The primary experiments use the official NF-ToN-IoT-v2 files released by the University of Queensland. The training file has 13,552,396 rows (2,127,627,131 bytes; SHA-256 `d33bb12d...8e004`), of which 4,880,476 are benign and 8,671,920 are attacks. The test file has 3,388,100 rows (531,923,548 bytes; SHA-256 `806b685c...15f2a9`), comprising 1,218,993 benign and 2,169,107 attack flows. We convert the CSV data by chunks into float32 feature memmaps and int8 label memmaps. After the two signed-log transforms, no non-finite values remain and no row is removed.

NF-UNSW-NB15-v2 and NF-BoT-IoT-v2 provide the extended experiments. Neither complete CSV release contains a separate train/test split, so we hash each full row deterministically: $h(\text{row}) \bmod 5 = 0$ assigns the row to test, and every other row to training. Identical raw rows cannot cross the partition boundary. NF-BoT-IoT-v2 is large and strongly imbalanced. Its training budget is therefore fixed at 2,000,000 rows, retaining every minority example and filling the remainder with majority examples drawn under a fixed seed. Evaluation still covers all 7,550,710 test flows. Feature locking, model fitting, and threshold optimization are repeated from the training partition of each dataset.

Dataset sizes, train/test partitions, and experimental access rules are summarized in Table 3. The external datasets are used only for per-dataset retraining and reproduction, not as evidence of zero-shot transfer.

3.2. Client Heterogeneity and Test Protocols

The default setting has $K = 5$ and label Dirichlet parameter $\alpha = 0.35$. For each class, client shares are drawn independently from Dirichlet($\alpha \mathbf{1}_K$) and limited to 1%–80% per client. We vary label skew with $\alpha \in \{0.05, 0.1, 0.35, 1, 10\}$. Quantity skew instead draws client sizes from a log-normal distribution while holding each client's class ratio fixed. To create feature heterogeneity without using labels, records are sorted on the first locked feature, `L7_PROTO`, and contiguous ranges are assigned to nodes. These three constructions alter class priors, sample counts, and covariates. Under seed 42, node allocations span 0.773–5.268 million edge samples and attack shares span 2.86%–99.23%. Every per-seed size and proportion is retained in the manifest.

Two protocols assess local models. In the unified global protocol, every node model predicts the full official test set and node metrics are macro-averaged. The distribution-matched protocol deterministically stratifies test samples to reproduce the benign/attack ratio of each node's training data. This stratification is used only at final evaluation. Comparisons among the primary method, probability averaging, centralized models, and federated baselines all use the same official test set; the local protocols supply supplementary, node-specific measurements.

3.3. Comparators and Fair Hyperparameter Selection

Comparators include: (1) learner-family mean representations with equal, sample-size, validation-F1, inverse-log-loss, or composite-quality weighting; (2) the full 14-dimensional distribution summary with equally weighted nodes; (3) a global 7-dimensional summary without family semantics and a 14-dimensional random balanced-group summary; (4) node-wise raw prediction concatenation, sorted within-family concatenation, and the mean of a single LightGBM family; (5) a heterogeneous one-model-per-node summary and 41-parameter Tiny Deep Sets; (6) logistic regression and compact LightGBM trained only on labeled $\mathcal{D}_{\text{meta}}$ samples, separating the contribution

Table 2. Training and sealed-test workflow for RA-LADS

Step	Operation	Data and model state
1	Partition official training indices into four mutually exclusive data roles	Record the random seed, subset sizes, and data hashes
2	Compare candidate dimensions on the feature pool and lock the top 10	Freeze and share the feature list across all nodes
3	Construct clients with label, quantity, or feature heterogeneity and split node fitting/validation sets	Keep node partitions disjoint; use a constant-probability model for a single-class node
4	Train LightGBM/XGBoost at each node and compute validation statistics	Upload models and quality fields; retain raw records locally
5	Construct raw, sorted, mean, dispersion, or quantile representations on the meta-learning pool and train logistic regression	Reuse the same base models and data roles across variants
6	Search the threshold pool for the maximum-F1 threshold and fixed-FAR operating points	Produce model-specific alert operating points
7	Evaluate detection quality, probability quality, and system cost on the sealed test	Output metrics, model bundles, and experiment records

Table 3. Dataset sizes, partitions, and experimental access rules

Dataset	Released/raw rows	Available training rows	Sealed/hashed test rows	Training protocol
NF-ToN-IoT-v2	16,940,496	13,552,396	3,388,100	Official train/test; default 85/5/5/5 training roles
NF-UNSW-NB15-v2	2,390,275	1,911,225	479,050	Deterministic full-row hash partition; full training group
NF-BoT-IoT-v2	37,763,497	30,212,787	7,550,710	Hash partition; 2,000,000-row training budget; full test group

of reference labels from that of node responses; and (7) a global average of all base-model probabilities, local protocols, centralized tree models, and FedAvg/FedProx MLPs. Centralized models access the pooled edge training data and serve as pooled-data performance references. Reference-pool baselines access only the 5% meta-learning pool and not the edge pool. Because these paradigms differ in capacity and inductive bias, their results do not support claims of universal superiority for any collaboration paradigm.

The structural ablation begins with five nodes, two learners, and seed 42. From one fixed base-model bundle, it fits the family mean, mean plus standard deviation, quantiles alone, node-wise raw concatenation, within-family sorting, and the full summary. The broader 20-seed study then compares family alignment with no-family pooling, random grouping, raw and sorted concatenation, a single family, genuine heterogeneity, two reference-pool baselines, and Tiny Deep Sets. Within a seed, every representation sees exactly the same independently trained base-model responses. Its fixed-FAR operating points are frozen separately on the common 10,001-point grid.

Node-count and reference-ratio experiments both use the full 14-dimensional method and five independent seeds. The former retrains the complete pipeline from client partitioning at $K = 2/5/10/20$. The latter fixes the feature pool at 5%, sets the combined meta-learning and threshold-pool proportion to 0.5%/1%/2%/5%/10%, and splits the allocation equally between the two pools. External datasets are retrained independently with five seeds. Reference robustness is evaluated conditionally on the fixed base-model responses from seed 42 by uniformly flipping 1%/5%/10%/20% of meta-learning labels (20 repetitions per level) and by separately simulating node dropout, complete family absence, and different attack priors in the threshold pool. These are conditional sensitivity analyses; repeated label flips are not described as independent base-model training runs.

All models that produce unified probabilities share $\mathcal{D}_{\text{threshold}}$ and the same threshold function; local models select node-specific thresholds from their own validation sets. Federated baselines use a single-hidden-layer ReLU MLP with 16 or 64 candidate hidden units, $\mu \in \{0.001, 0.01, 0.1\}$ for FedProx, and $\mu = 0$ for FedAvg. In each round, every client traverses its local data sequentially in a batch of 65,536 rows, with learning

rate 0.02 and $L_2 = 10^{-4}$. Local loss is weighted by client class frequency, and the server averages client parameters by sample size. Network width, μ, and round count are selected using a stratified in-training subset of at most 1,000,000 rows, after which the selected configuration is retrained on the complete edge pool.

3.4. Metrics, Statistical Units, and Reproducibility Environment

The primary endpoint is attack-class F1; precision, recall, FAR, ROC-AUC, and AUPRC are reported jointly. Let FP and TN be the numbers of false positives and true negatives among benign flows, respectively, so that $FAR = FP/(FP + TN)$. Fixed-FAR analyses additionally report false alarms per million benign flows, $10^6 FAR$. To avoid a misleading deployment interpretation of precision caused by the 63.99% attack prevalence in the test set, a prior-adjusted positive predictive value is calculated from test sensitivity and specificity for an assumed attack prior π:

$$PPV_\pi = \frac{Recall \cdot \pi}{Recall \cdot \pi + FAR \cdot (1 - \pi)}, \tag{13}$$

We also report alerts per million total flows. This conversion is a conditional scenario analysis and is not equivalent to measuring the true online prior. Resource endpoints include uncompressed model size, compressed deployment-bundle size, per-gateway and fleet-wide distribution volume, loading time, resident-set-size (RSS) increase, inference quantiles, throughput, and single-host multiprocess wall-clock time.

Weighting, non-IID, core-structure, and fixed-FAR experiments use 20 independent partitioning and training seeds as statistical units. Node-count, reference-ratio, and external-dataset experiments use five seeds. Methods are summarized by across-seed means, standard deviations, and bootstrap 95% intervals. Paired differences additionally report Cohen's d_z, paired t tests, and Holm correction for multiple comparisons. These intervals quantify partitioning and training randomness on a fixed public test set; they do not cover dataset sampling, site variation, or temporal drift. Model-dropout, label-noise, and threshold-prior experiments are conditional on the seed-42 base models. Single-host process-scaling experiments are repeated three times for each worker count.

All experiments ran on an Apple M3 arm64 host with eight CPU cores, a 10-core integrated GPU, and 24 GB unified memory. Tree models did not use the GPU. The `uv.lock` file fixes Python 3.12.10, NumPy 2.3.5, pandas 2.2.3, scikit-learn 1.9.0, LightGBM 4.6.0, and XGBoost 3.3.0. Raw outputs are stored as CSV; JSON manifests record data and run metadata. The 21 unit tests all passed. They cover the core protocol, fixed-FAR thresholds, grouped summaries, permutation properties, Tiny Deep Sets, partition integrity, constant-model fallback, FedProx at $\mu = 0$, and related checks.

4. Results

The results are organized into four evidence blocks: contribution attribution (Sections 4.1–4.2), operating-point and heterogeneity tests (Sections 4.3–4.5), resource and scaling measurements (Sections 4.6–4.8), and external and applicability checks (Sections 4.9–4.10). Throughout these blocks, statistical reliability is interpreted separately from engineering magnitude.

4.1. Primary Sealed-Test Comparison: Collaboration Gains and the Pareto Boundary

Table 4 presents the primary comparison of detection performance, probability quality, and model size for NF-ToN-IoT-v2 under the unified sealed-test protocol.

Figure 2 further illustrates the Pareto relationships among detection quality, false-alarm rate, and model storage.

Among the distributed variants, the primary configuration achieved the highest F1, AUPRC, and probability-quality scores, but not the lowest FAR. Its maximum-F1 improvement therefore arose mainly from higher attack recall rather than from a general reduction in false alarms; the ablations below identify which response structure contributed to that result.

The centralized random forest produced the strongest detection results at approximately 148 MB, whereas compact centralized LightGBM occupied 89.4 KB and reached F1 of 0.9839. The 819.7 KB complete RA-LADS bundle occupies an intermediate region and exchanges additional storage for a heterogeneous-model interface without pooling raw edge records.

4.2. Attribution Controls for Family Semantics, Representation Capacity, and Reference Labels

The 20-seed matched-base-model attribution and matched-budget controls are summarized in Table 5.

Mean F1 was 0.984514 for the full summary and 0.983758 after family semantics were removed. The paired difference was 0.000756 (95% CI: 0.000559–0.000962; $d_z = 1.618$; Holm-adjusted $p = 1.08 \times 10^{-5}$). Random balanced grouping reached 0.984218. Its deficit from true family grouping was 0.000296, but the Holm-adjusted p value was 0.077. Structured groups are therefore better supported than complete pooling; the data do not show that learner labels are the only grouping rule that works.

Node-wise raw concatenation produced mean F1 of 0.984523, only −0.000009 from the full summary (95% CI: −0.000222–0.000221). Sorted within-family concatenation was also indistinguishable from the summary. The 14-dimensional vector does not exceed the accuracy obtained by retaining every response. Its advantage is a

Table 4. Primary sealed-test comparison on NF-ToN-IoT-v2. Except for local protocols marked with †, all methods use the same official global test set and in-training threshold budget. The RF occupies approximately 148.0 MB

Model	Test protocol	F1	AUPRC	FAR	FAR@ 95%TPR	Brier	ECE	Size/KB
Distribution-summary stacking–equal (primary)	Official global	0.9854	0.9987	0.0407	0.0099	0.0155	0.0091	819.7
Learner-mean stacking–equal	Official global	0.9825	0.9985	0.0431	0.0119	0.0179	0.0095	819.7
Single-LightGBM collaborative variant	Official global	0.9837	0.9981	0.0355	0.0096	0.0175	0.0115	440.4
Global probability average	Official global	0.9834	0.9985	0.0381	0.0099	0.0194	0.0508	819.0
Centralized LightGBM	Pooled upper ref.	0.9839	0.9981	0.0323	0.0092	0.0165	0.0270	89.4
Centralized XGBoost	Pooled upper ref.	0.9824	0.9977	0.0317	0.0134	0.0186	0.0355	77.7
Centralized random forest	Pooled upper ref.	0.9965	0.9998	0.0079	0.0027	0.0040	0.0043	151,539.0
Local distribution-matched†	Simulated client strata	0.9914	0.9993	0.0134	0.0046	0.0114	0.0270	165.6/ node
Local unified-global macro mean†	Client macro mean	0.9710	0.9982	0.0351	0.0102	0.0233	0.0532	163.8/ node

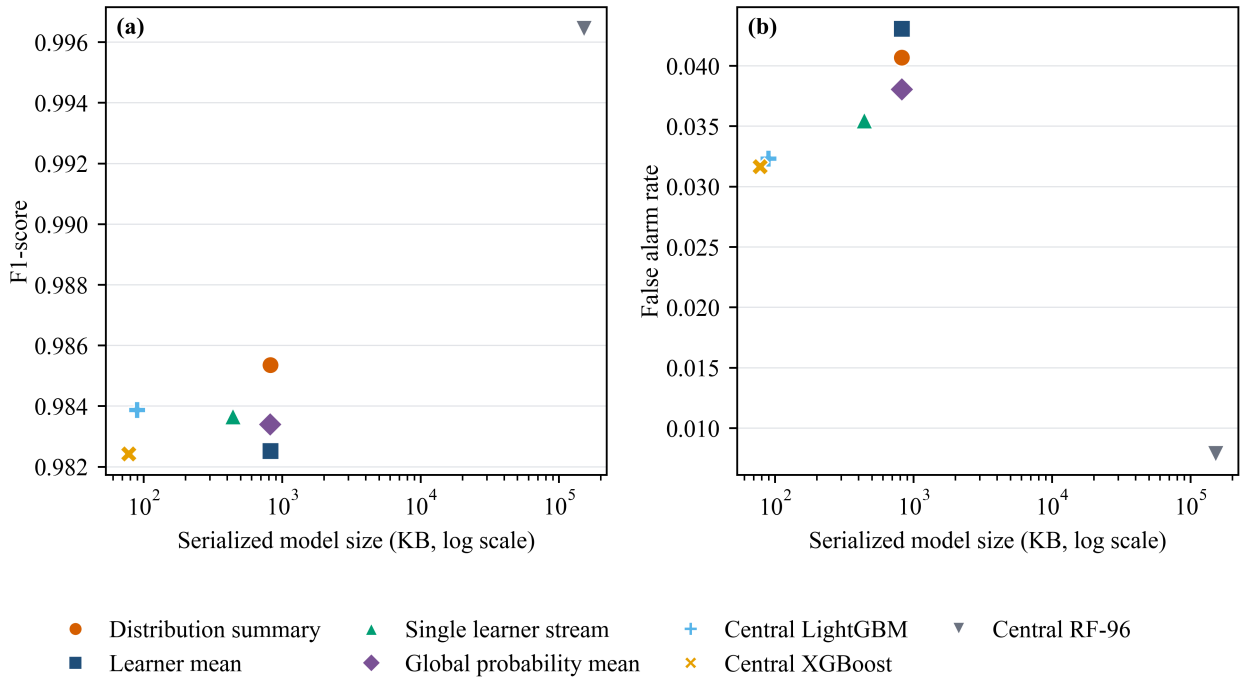

Figure 2. Pareto relationship between sealed-test detection quality and model storage. The left panel compares F1 and FAR; the right panel shows the F1–model-size trade-off on a logarithmic storage axis. Centralized models access pooled training data

permutation-invariant interface whose width stays fixed as K changes. At seed 42, F1 was 0.985278 for quantiles alone, 0.985361 for the full summary, and 0.985330 for raw concatenation, against 0.982546 for mean plus standard deviation. This decomposition points to within-family order structure as the main increment.

Reference-pool logistic regression achieved F1 of only 0.798022, whereas compact reference-pool LightGBM reached 0.983887. Thus, the 5% labeled reference

Table 5. Twenty-seed attribution and matched-budget controls under default label skew. Brackets denote bootstrap 95% CIs; model size is the across-seed mean of uncompressed deployment content

Method	Meta-dim.	Permutation invariant	Mean F1 [95% CI]	Mean FAR [95% CI]	Size/KB
RA-LADS family-aligned summary	14	Yes	0.984514 [0.984329, 0.984686]	0.035876 [0.033734, 0.038120]	798.3
No-family global summary	7	Yes	0.983758 [0.983544, 0.983958]	0.037328 [0.036085, 0.038587]	798.3
Random balanced-group summary	14	Yes	0.984218 [0.984027, 0.984409]	0.036803 [0.034403, 0.039247]	798.3
Node-wise raw prediction concatenation	10	No	0.984523 [0.984267, 0.984742]	0.036402 [0.034351, 0.038422]	798.3
Sorted within-family concatenation	10	Yes	0.984449 [0.984265, 0.984625]	0.036833 [0.035171, 0.038666]	798.3
Heterogeneous one-model-per-node summary	14	Yes	0.983865 [0.983687, 0.984057]	0.037938 [0.036106, 0.039809]	403.9
Single-LightGBM family mean	1	Yes	0.983755 [0.983622, 0.983895]	0.036257 [0.034864, 0.037804]	438.2
Compact reference-pool LightGBM	–	–	0.983887 [0.983790, 0.983970]	0.036493 [0.034440, 0.038762]	87.7
Tiny Deep Sets (41 parameters)	Set	Yes	0.983659 [0.983534, 0.983783]	0.036058 [0.034796, 0.037256]	798.6

pool alone can train a strong tree model and is a necessary control. The full summary improved over reference-pool LightGBM by 0.000626 in paired F1 (95% CI: 0.000398–0.000846; Holm-adjusted $p = 4.35 \times 10^{-4}$). Node responses therefore contribute small but reproducible information, although far less than the gap to the weak linear reference baseline. Tiny Deep Sets, despite 41 second-stage parameters, remained 0.000855 below the full summary (Holm-adjusted $p = 1.26 \times 10^{-6}$); the present data do not justify greater meta-layer capacity. The genuinely heterogeneous one-model-per-node setting approximately halved bundle size but reduced F1 by 0.000649 relative to the complete two-model configuration (Holm-adjusted $p = 3.67 \times 10^{-4}$), quantifying the capacity–performance trade-off under model heterogeneity.

Statistical reproducibility and engineering importance should therefore be separated. The 0.000626 paired improvement over compact reference-pool LightGBM is 0.0626 percentage points in absolute F1, whereas the difference from raw concatenation is essentially zero. Moreover, no multiplicity-adjusted advantage over a single LightGBM appears at 0.1%–2% FAR. The practical case for the complete method rests on its fixed, permutation-invariant, auditable interface; deployment is justified only when that interface benefit and a site-specific operating-point gain outweigh the additional labeling, storage, and inference costs.

4.3. Alert Operating Points Under Fixed False-Alarm-Rate Budgets

The maximum-F1 threshold does not represent an industrial alert budget; the 20-seed operating points under fixed FAR budgets from 0.1% to 5% are reported in Table 6.

At the 0.1% budget, actual FAR was 0.000989—about 989 false alarms per million benign flows—and recall was 0.657968. Raising the budget to 0.5% increased recall to 0.908999, but also implied roughly 5,014 false alarms per million benign flows. After Holm correction, the full summary and a single LightGBM did not differ in paired F1 at 0.1%, 0.5%, 1%, or 2% FAR. A significant advantage appeared only at the 5% budget: F1 increased by 0.000417 (95% CI: 0.000188–0.000649; Holm-adjusted $p = 0.017$) and recall by 0.000986 (Holm-adjusted $p = 0.0016$). These results do not support describing RA-LADS as a strictly low-false-alarm detector. Between 0.1% and 2% FAR, a single LightGBM is at least as competitive.

Deployment priors further change the operational meaning of alerts. Holding the above sensitivity and specificity fixed, an actual attack prior of 0.1% gives prior-adjusted $PPV_\pi = 0.3999$ at the 0.1% FAR operating point and approximately 1,646 alerts per million total flows. At the 0.5% and 1% FAR operating points, PPV_π falls to 0.1538 and 0.0874, respectively. If the attack prior is 1%, the corresponding values are 0.8701, 0.6469, and 0.4914. Raw precision from a sealed test with a high attack prevalence cannot therefore be extrapolated

Table 6. Twenty-seed sealed-test operating points under fixed FAR budgets. Brackets denote bootstrap 95% CIs and are placed below point estimates

Budget	RA-LADS F1 [95% CI]	Single LGB F1 [95% CI]	RA-LADS Recall [95% CI]	Single LGB Recall [95% CI]	RA-LADS FAR [95% CI]	False alarms per million benign flows
0.1%	0.792071 [0.773955, 0.809360]	0.775064 [0.736081, 0.799874]	0.657968 [0.633265, 0.681866]	0.638670 [0.595775, 0.667234]	0.000989 [0.000953, 0.001023]	989
0.5%	0.950806 [0.945135, 0.955214]	0.955721 [0.954132, 0.956961]	0.908999 [0.898864, 0.916923]	0.917832 [0.914941, 0.920086]	0.005014 [0.004912, 0.005098]	5,014
1%	0.971449 [0.970349, 0.972523]	0.971870 [0.971577, 0.972142]	0.949766 [0.947671, 0.951819]	0.950538 [0.949990, 0.951052]	0.009931 [0.009847, 0.010014]	9,931
2%	0.981562 [0.981271, 0.981847]	0.981708 [0.981608, 0.981812]	0.974563 [0.974033, 0.975094]	0.974796 [0.974566, 0.975037]	0.019885 [0.019727, 0.020032]	19,885
5%	0.983128 [0.982876, 0.983359]	0.982711 [0.982522, 0.982904]	0.993653 [0.993166, 0.994117]	0.992668 [0.992396, 0.992946]	0.049393 [0.049173, 0.049589]	49,393

directly to low-base-rate production networks; thresholds must be refrozen using site-specific priors and alert-handling capacity.

4.4. Ablation of Node-Reliability Weighting

Figure 3 shows the paired ablation distributions for the node-weighting rules over 20 independent partitioning seeds.

Paired effects relative to equally weighted learner-mean stacking and the corresponding multiplicity-adjusted comparisons are summarized in Table 7.

Equal weighting yielded F1 of 0.983238 ± 0.000443; composite-quality weighting yielded 0.983280 ± 0.000450. Their paired difference was only 4.16×10^{-5}, with bootstrap interval $[3 \times 10^{-6}, 9.5 \times 10^{-5}]$ and Holm-adjusted $p = 0.324$. Weighting by sample size added 0.000204 over the equal rule (Holm-adjusted $p = 0.030$), whereas using a single LightGBM added 0.000517 (Holm-adjusted $p < 0.001$). Combining several validation statistics into one quality score did not produce a stable gain. Family alignment and meta-learning appear to have absorbed most of the reliability information available here.

Inverse-log-loss weighting reduced mean F1 by 0.000089 but lowered mean FAR from 0.03833 to 0.03716, revealing opposing effects on detection and false-alarm operating points. Considering both effect sizes and multiplicity-adjusted comparisons, subsequent experiments use equal node weights as the default distribution representation and treat explicit reliability weighting as a scenario-dependent option.

4.5. Stress Tests Under Multiple Forms of Non-IID Data

Label skew uses Dirichlet $\alpha \in \{0.05, 0.10, 0.35, 1, 10\}$, quantity skew changes client sample sizes, and feature partitioning deterministically sorts and splits data by the first training-locked feature, `L7_PROTO`, without accessing labels. Each of the three scenarios uses 20 partitioning and training seeds, producing 700 complete evaluation records.

The 20-seed numerical results comparing the distribution summary with the learner-family mean across label-skew strengths are summarized in Table 8.

Strong label skew at $\alpha = 0.05$ produced the widest across-seed interval. The distribution representation exceeded the 2-dimensional family mean in F1 at all five α levels, with paired differences from 0.000369 to 0.001276. The largest gain occurred at $\alpha = 0.35$, with bootstrap 95% CI $[0.001081, 0.001512]$ and Holm-adjusted $p = 4.42 \times 10^{-9}$. FAR-difference intervals crossed zero under all five label-skew conditions, indicating that the F1 gain was reproducible whereas FAR improvement depended on the operating point.

F1 and false-alarm-rate trends with bootstrap intervals across Dirichlet label-skew strengths are shown in Fig. 4.

The paired numerical effects under quantity skew and feature-sorted partitioning are summarized in Table 9.

Under quantity skew, the distribution representation and family mean achieved F1 scores of 0.983747 and 0.983728, respectively. The paired difference was 0.000019 and its confidence interval crossed zero; the FAR difference was likewise near zero. Under feature partitioning, in contrast, the distribution representation increased F1 from 0.975733 to 0.981293, a mean gain of 0.005560, while reducing FAR from 0.064427 to 0.028947, a mean reduction of 0.035481. The Holm-adjusted p values for the two effects were 4.91×10^{-11} and 4.12×10^{-10}, respectively. Dispersion and quantiles therefore contributed additional discriminative information primarily when the shape of the node-response distribution changed; sample-size variation alone was insufficient to activate this advantage.

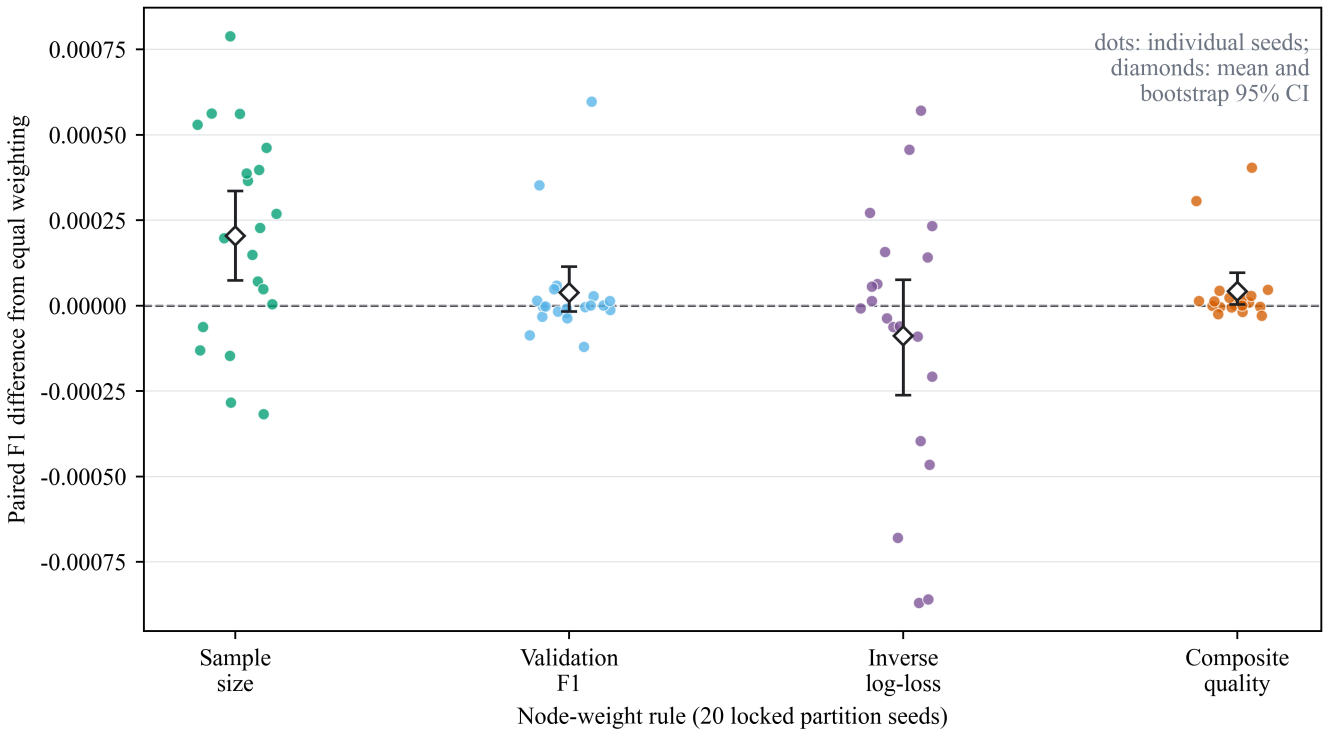

Figure 3. Twenty-seed paired ablation of node-weighting rules. Each point represents a locked partitioning and training seed; thick lines and intervals denote means and bootstrap 95% CIs. The gain from composite weighting over equal weighting was minimal and not significant after Holm correction

Table 7. Paired F1 effects relative to equally weighted learner-mean stacking ($n = 20$)

Method	Mean diff.	95% CI	d_z	Raw p	Holm p
Composite quality	0.000042	[0.000003, 0.000095]	0.377	0.108	0.324
Validation F1	0.000038	[-0.000018, 0.000115]	0.240	0.297	0.595
Sample size	0.000204	[0.000074, 0.000339]	0.667	0.0076	0.030
Inverse log loss	-0.000089	[-0.000266, 0.000071]	-0.225	0.326	0.595
Single LightGBM	0.000517	[0.000324, 0.000721]	1.125	0.000074	0.00037

Table 8. Twenty-seed results for the distribution summary relative to the equally weighted learner-family mean under label skew. Brackets denote bootstrap 95% CIs; differences are computed as distribution summary minus learner mean

α	Distribution-summary F1 [95% CI]	Learner-mean F1	ΔF1 [95% CI]	ΔFAR [95% CI]
0.05	0.981905 [0.978458, 0.983916]	0.981536	0.000369 [0.000142, 0.000625]	-0.000726 [-0.002559, 0.000980]
0.10	0.984049 [0.983823, 0.984271]	0.983112	0.000937 [0.000635, 0.001242]	-0.000677 [-0.002943, 0.001478]
0.35	0.984514 [0.984329, 0.984686]	0.983238	0.001276 [0.001081, 0.001512]	-0.002458 [-0.005104, 0.000295]
1.00	0.984333 [0.984130, 0.984523]	0.983569	0.000764 [0.000558, 0.000958]	0.000250 [-0.001940, 0.002404]
10.00	0.984101 [0.983954, 0.984241]	0.983691	0.000410 [0.000245, 0.000568]	-0.000380 [-0.002888, 0.002137]

The seed-wise effect distributions of the distribution summary relative to the learner-family mean under the three heterogeneity scenarios are shown in Fig. 5.

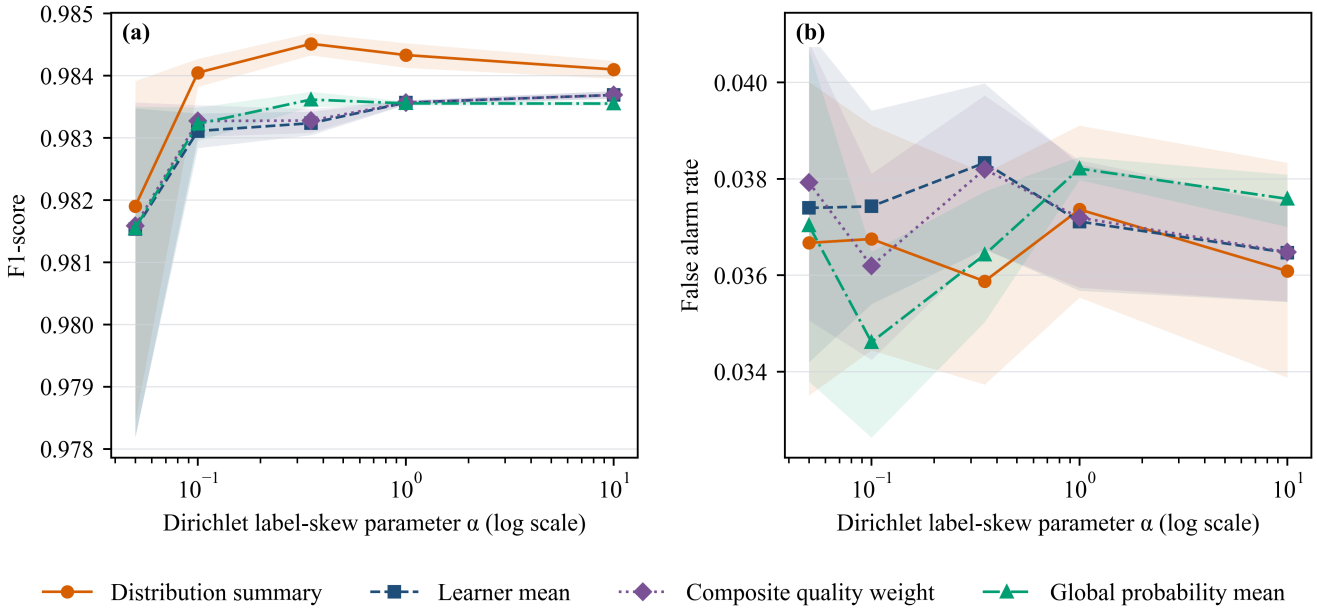

Figure 4. Label-skew intensity and sealed-test operating points. All four curves use the same official global test protocol; shaded regions denote bootstrap 95% CIs over 20 locked seeds. Distribution-matched local tests are excluded because their protocol differs. (a) F1; (b) false alarm rate

Table 9. Paired effects for representative settings of three heterogeneity types ($n = 20$). Holm p values correspond to F1 comparisons

Scenario	Distribution-summary F1	Learner-mean F1	ΔF1 [95% CI]	ΔFAR [95% CI]	Holm p
Label skew ($\alpha = 0.35$)	0.984514	0.983238	0.001276 [0.001081, 0.001512]	-0.002458 [-0.005104, 0.000295]	4.42×10^{-9}
Quantity skew	0.983747	0.983728	0.000019 [-0.000019, 0.000065]	0.000082 [-0.000401, 0.000680]	0.403
Feature-sorted partitioning	0.981293	0.975733	0.005560 [0.004856, 0.006298]	-0.035481 [-0.040753, -0.030233]	4.91×10^{-11}

4.6. Engineering Sensitivity to Node Count and Reference-Data Ratio

Table 10 reports a five-seed node-count scan for the full 14-dimensional method. At $K = 2/5/10/20$, mean F1 ranged from 0.983741 to 0.984482, with overlapping confidence intervals and no monotonic improvement with node count. Mean uncompressed bundle size increased from 319.8 KB to 3,045.1 KB, and batch-amortized per-flow inference time over the complete sealed test increased from 0.001152 ms to 0.010910 ms. These results verify that the meta-interface remains 14-dimensional, not that system cost is independent of K; each gateway must still store and execute all available base models.

For manufacturing deployment planning, these measurements separate interface standardization from fleet cost. A site can keep the same 14 meta-features across production-segment gateways, but storage, bundle distribution, and inference headroom must still be sized for the participating node models.

Applying the measured uncompressed bundle sizes to the lifecycle formula gives approximate node-to-coordinator collection payloads of 0.32, 0.80, 1.56, and 3.05 MB for $K = 2, 5, 10, 20$, respectively, and full-fleet distribution payloads of 0.64, 4.01, 15.60, and 60.90 MB per rollout (using 1 MB = 1000 KB). These are analytical payload estimates from serialized model size, not measured network traffic; transport headers, signatures, rollback copies, retransmission, and update frequency are excluded. Energy consumption, sustained-load thermals, and end-to-end latency likewise remain unmeasured and must be profiled on the target industrial gateway.

Reference-ratio sensitivity with the feature pool fixed at 5% is reported in Table 11.

This experiment varies only the combined proportion of the meta-learning and threshold-tuning pools and

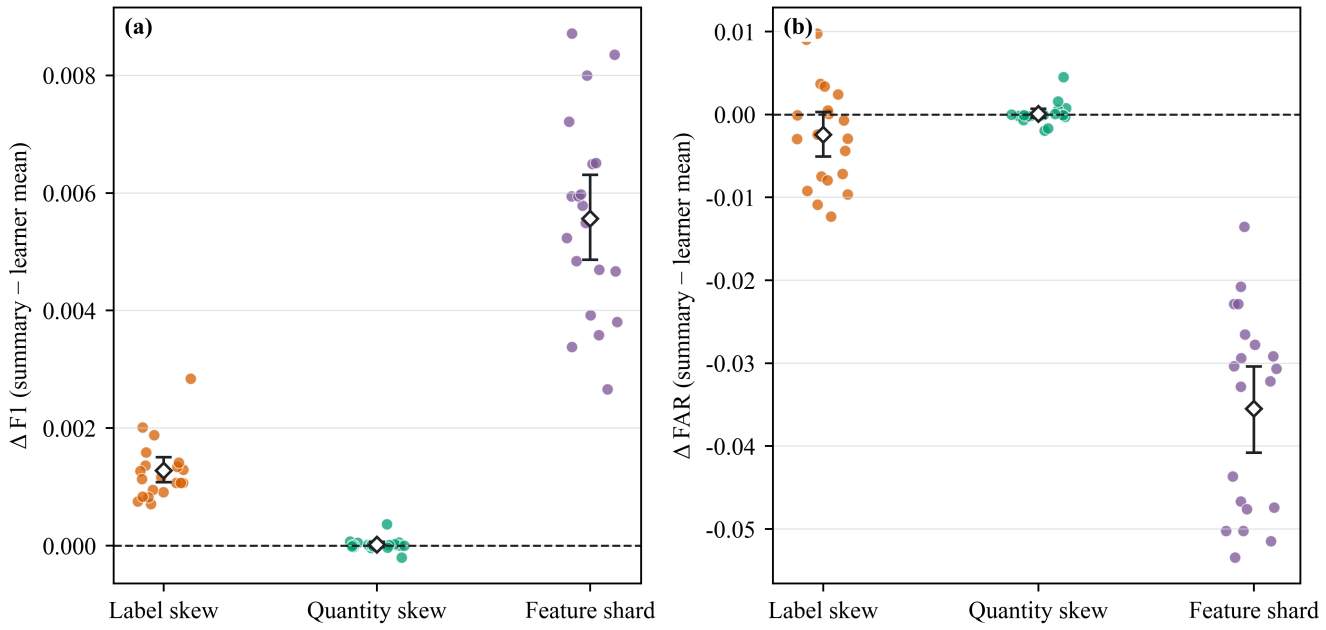

Figure 5. Twenty-seed paired effects of the distribution summary relative to the equally weighted learner-family mean. Colored points are seed-level differences; diamonds and error bars denote means and bootstrap 95% CIs; dashed lines denote zero effect. (a) F1; (b) false alarm rate. Label skew uses $\alpha = 0.35$

Table 10. Five-seed full-retraining results for the complete 14-dimensional summary at different node counts. Brackets denote bootstrap 95% CIs

K	Mean F1 [95% CI]	Mean FAR [95% CI]	Bundle/ gateway (KB)	Inference/ flow (ms)	Full-test inference time (s)	Meta-dim.
2	0.983741 [0.983384, 0.984010]	0.037570 [0.034208, 0.041201]	319.8	0.001152	3.905	14
5	0.984252 [0.983725, 0.984899]	0.039404 [0.035981, 0.043784]	802.4	0.002759	9.346	14
10	0.984482 [0.984269, 0.984694]	0.038002 [0.035086, 0.040973]	1,560.2	0.005735	19.431	14
20	0.984158 [0.983770, 0.984681]	0.037598 [0.032073, 0.043592]	3,045.1	0.010910	36.964	14

fully retrains five seeds for each configuration. At a combined ratio of 0.5%, each pool occupies only 0.25% of the training set but still contains 33,881 labeled flows; mean F1 was 0.984180. Increasing the combined ratio to 10% yielded mean F1 of 0.984252. The largest difference among the five mean F1 values was only 0.000699, all FAR intervals overlapped, and no monotonic scale benefit emerged. This conclusion depends on the current large dataset: even the smallest ratio corresponds to tens of thousands of labeled flows and does not support a claim that the method requires very few reference labels.

For manufacturing data governance, the relevant requirement is therefore the absolute volume and representativeness of reviewed reference labels rather than their nominal percentage alone. These results do not establish that RA-LADS can be commissioned from a small plant incident archive.

4.7. Federated Baselines With In-Training Capacity Selection and Unified Thresholds

Table 12 reports FedAvg/FedProx retraining on the complete edge pool after in-training model selection. The search selected 64 hidden units and 20 rounds for both methods, with an optimal proximal coefficient of $\mu = 0.001$ for FedProx.

FedAvg/FedProx achieved sealed-test F1 of 0.9096/0.9095, AUPRC of 0.9482/0.9482, and FAR

Table 11. Five-seed sensitivity of the complete 14-dimensional method to the reference-data ratio. The feature pool is fixed at 5% of the training set; meta-learning and threshold-tuning pools equally divide the reported combined ratio

Combined ratio of two pools	Meta labels	Threshold labels	Mean F1 [95% CI]	Mean FAR [95% CI]
0.5%	33,881	33,881	0.984180 [0.983858, 0.984526]	0.037107 [0.031197, 0.042642]
1%	67,762	67,762	0.984654 [0.984469, 0.984842]	0.037734 [0.033272, 0.043045]
2%	135,524	135,524	0.984317 [0.983725, 0.985003]	0.038119 [0.035270, 0.040726]
5%	338,810	338,810	0.983955 [0.983688, 0.984227]	0.034584 [0.031645, 0.039898]
10%	677,620	677,620	0.984252 [0.983725, 0.984899]	0.039404 [0.035981, 0.043784]

Table 12. FedAvg/FedProx baselines retrained on the complete edge pool after in-training selection. Test thresholds use the same in-training rule as the primary method

Model	Hidden units	μ	Rounds	F1	AUPRC	FAR	Parameters/size	Training communication (MB)
FedAvg-MLP	64	0	20	0.9096	0.9482	0.2693	769 / 3.004 KB	0.5867
FedProx-MLP	64	0.001	20	0.9095	0.9482	0.2704	769 / 3.004 KB	0.5867

of 0.2693/0.2704. Their training trajectories and test operating points were similar, indicating that this proximal coefficient did not materially alter the decision boundary of the current MLP. Each MLP contains 769 float32 parameters (3.004 KB), substantially smaller than the 819.7 KB tree-model bundle. This low storage cost is accompanied by a pronounced reduction in detection performance, reflecting the capacity and inductive-bias limitations of a single-hidden-layer ReLU network on the present 10-dimensional tabular traffic data. Assuming five clients, 20 rounds, and one parameter upload and download per round, training communication is 0.5867 MB for both FedAvg and FedProx. For RA-LADS, one-time model collection and distribution to five gateways require approximately 0.80 MB and 4.00 MB, respectively. Federated communication accumulates with training rounds, whereas RA-LADS traffic is dominated by replication of the complete tree bundle.

FedAvg/FedProx produced ROC-AUC of 0.92615/0.92613, FAR@TPR95 of 0.24970/0.24972, Brier scores of 0.11044/0.11046, 15-bin ECE of 0.10087/0.10089, and in-training thresholds of 0.234/0.233. The two methods were similar in ranking ability, probability quality, and operating point, and both were substantially below tree-model stacking. Retraining on the complete edge pool required 477.7/437.9 s, while candidate-configuration search required 60.2/177.4 s.

4.8. Local arm64 Inference and Single-Host Process Scaling

After joblib compression, the distribution-representation deployment bundle occupied 284.858 KB and had cold-load p50/p95/p99 latencies of 4.049/5.258/5.834 ms. RSS increased from 170.875 to 176.094 MB during loading, an increment of 5.219 MB; peak process memory including the Python runtime was 178.656 MB. For batch size 1, per-flow p50/p95/p99 latencies were 1.386/1.524/1.702 ms and median throughput was 721.6 flows/s. At batch size 4096, per-flow p50 fell to 0.00305 ms and throughput reached 327,966 flows/s. The 2-dimensional mean variant had single-flow p50/p95 latencies of 1.323/1.450 ms. Key resource metrics are summarized in Table 13.

Latency quantiles and throughput across batch sizes are shown in Fig. 6.

A 32-flow warm-up preceded timing, and batch sizes 1/32/256/4096 were repeated 200/100/60/30 times, respectively. Larger batches amortized fixed Python-call and tree-traversal overhead, but total batch latency increased from 1.386 ms to 12.489 ms. At 4096 flows, the 2-dimensional mean representation achieved 350,763 flows/s, compared with 327,966 flows/s for the 14-dimensional distribution representation, quantifying the additional cost of dispersion and quantile computation.

The single-host process experiment fixes 10 node–learner fitting tasks, includes process startup and scheduling in wall-clock time, and excludes pregenerated caches. Increasing the worker count from 1 to 5 reduced runtime from 38.37 ± 7.34 s to 17.86 ± 3.38 s,

Table 13. Key resource metrics for the five-node distribution-representation deployment bundle

Metric	Measured value
Compressed deployment bundle	284.858 KB
Cold-load p95	5.258 ms
RSS loading increment	5.219 MB
Single-flow inference p95	1.524 ms
Batch-4096 throughput	327,966 flows/s

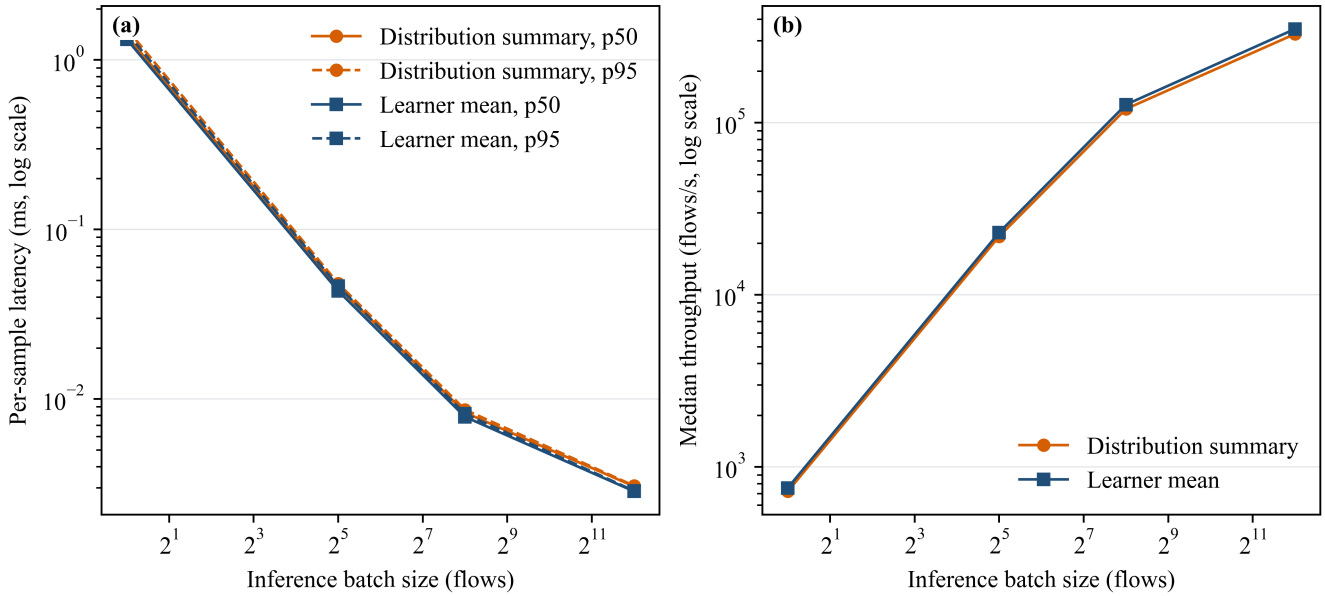

Figure 6. Local inference profile on a single Apple M3/arm64 host. The left panel shows per-flow latency quantiles at different batch sizes; the right panel shows throughput. Measurements include all base-model and meta-model inference, but exclude networking, acquisition, and feature extraction

corresponding to speedup 2.149 and parallel efficiency 0.430. Replicate variability at 2–4 processes indicates that unified-memory bandwidth, internal threading in the tree learners, and process scheduling limited linear scaling (Fig. 7).

4.9. Per-Dataset Retraining Reproduction

Five-seed per-dataset retraining results on the two external NetFlow-v2 datasets are reported in Table 14.

On NF-UNSW-NB15-v2, the three confidence intervals overlapped and the distribution summary showed no advantage. On NF-BoT-IoT-v2, all mean F1 scores exceeded 0.9996, indicating a nearly saturated task, and the FAR intervals again overlapped. These results support reproducibility after independent per-dataset retraining only; they do not establish cross-temporal, cross-site, real-industrial, or zero-shot generalization.

4.10. Reference-Pool Perturbation and Model-Availability Stress Tests

Table 15 summarizes conditional sensitivity under the fixed seed-42 base-model responses. Uniform random flipping of meta-learning labels produced mean F1 values from 0.984621 to 0.984498 at 1%–20% noise, with a maximum decrease of 0.000863 relative to the no-noise point estimate. This experiment covers only random symmetric noise under one base-model seed and cannot be extrapolated to targeted reference-pool poisoning. A larger change arose from the threshold-pool attack prior: sampling the threshold pool at a 0.1% attack prior reduced test FAR to 0.000455 but simultaneously reduced recall to 0.630365. Using the original 63.99% attack prior yielded FAR of 0.040680 and recall of 0.993347. Thresholds must be maintained according to site base rates and alert costs rather than copied directly from a high-attack-prevalence benchmark.

Removing one node at a time and refitting the meta-model and threshold produced F1 values of 0.984364–0.985264 and FAR values of 0.031886–0.041180. Removing an entire learner family forced a switch to the 7-dimensional single-family mode; F1 was then 0.983670 for LightGBM and 0.982268 for XGBoost. The system remains usable after static recalibration. The experiment

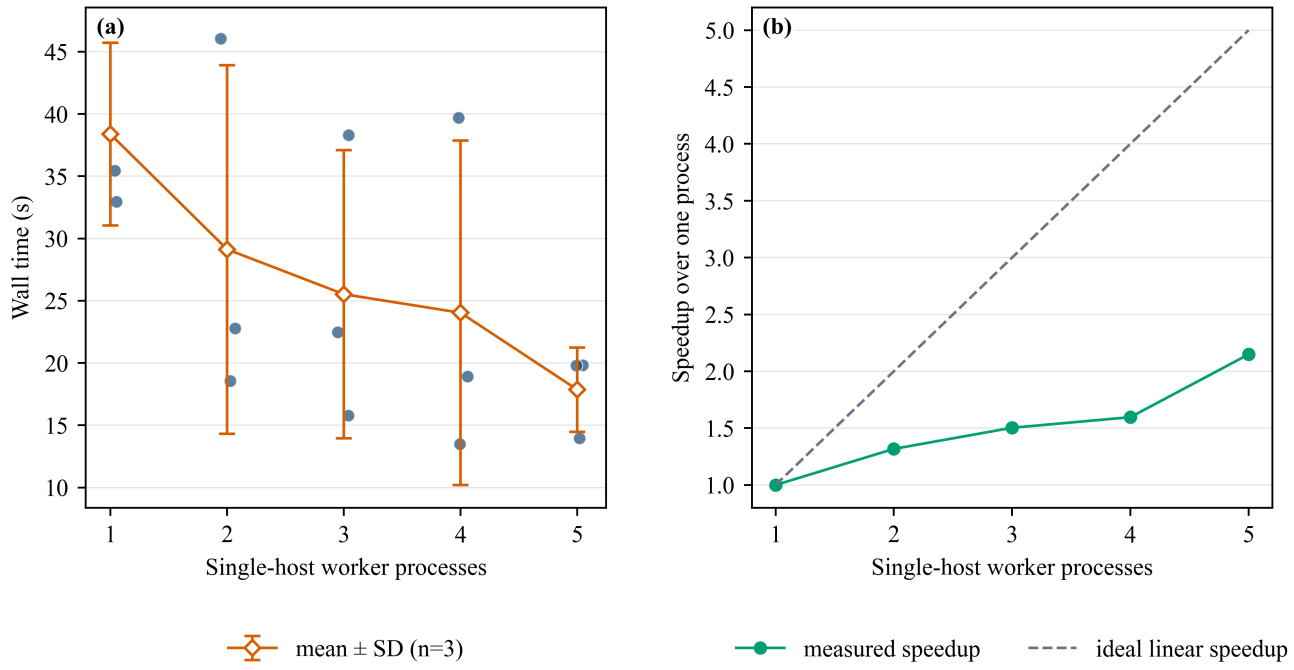

Figure 7. Wall-clock time, speedup, and efficiency for 1–5 worker processes on one host ($n = 3$ per setting). Cache creation is excluded; process startup and scheduling are included

Table 14. Five-seed per-dataset retraining on external datasets. Feature locking, model training, and threshold optimization are performed independently within each training group; brackets denote bootstrap 95% CIs

Dataset	Model	Mean F1 [95% CI]	Mean recall [95% CI]	Mean FAR [95% CI]
NF-UNSW-NB15-v2	Distribution-summary stacking	0.954376 [0.952742, 0.956124]	0.976168 [0.971648, 0.980699]	0.002868 [0.002626, 0.003201]
	Learner-family mean	0.954461 [0.950688, 0.956947]	0.978412 [0.971880, 0.987462]	0.002966 [0.002480, 0.003699]
	Global probability average	0.955604 [0.954020, 0.957112]	0.974546 [0.971111, 0.978054]	0.002687 [0.002474, 0.002950]
NF-BoT-IoT-v2	Distribution-summary stacking	0.999672 [0.999630, 0.999705]	0.999354 [0.999268, 0.999422]	0.002980 [0.002233, 0.003616]
	Learner-family mean	0.999624 [0.999528, 0.999687]	0.999258 [0.999065, 0.999387]	0.002929 [0.002137, 0.004031]
	Global probability average	0.999669 [0.999640, 0.999698]	0.999353 [0.999300, 0.999408]	0.004193 [0.002655, 0.006456]

does not demonstrate seamless switching during an abrupt runtime disconnection.

5. Discussion

5.1. Representation Mechanism for Heterogeneous-Model Collaboration

The main contribution of RA-LADS is an auditable response representation for collaboration among heterogeneous trees. Reference inputs supply common functional coordinates; learner families define candidate groups; symmetric statistics remove dependence on node identity. Instead of a high-capacity set encoder, the design uses 14 explicit statistics and a linear meta-learner. Removing family semantics lowered F1 consistently across 20 seeds. Random balanced groups, however, were not significantly worse than true family groups after Holm correction. The mechanism supported by the data is thus structured grouping rather than complete pooling, with family labels acting as a useful but non-unique prior. These

Table 15. Sensitivity to reference-pool perturbation and model availability conditional on the seed-42 base models. Label-noise intervals are based on 20 independent flips; all other values are conditional point estimates

Condition	Rep./ config.	F1 [95% CI]	Recall [95% CI]	FAR [95% CI]
No label noise	1	0.985361	0.993347	0.040680
Random meta-label flips, 5%	20	0.984592 [0.984586, 0.984598]	0.989361 [0.989339, 0.989389]	0.036169 [0.036119, 0.036236]
Random meta-label flips, 20%	20	0.984498 [0.984472, 0.984520]	0.989504 [0.989432, 0.989565]	0.036775 [0.036717, 0.036822]
Threshold-pool attack prior, 0.1%	20 samples	0.773159 [0.773159, 0.773159]	0.630365 [0.630365, 0.630365]	0.000455 [0.000455, 0.000455]
Threshold-pool attack prior, 1%	20 samples	0.915795 [0.915715, 0.915871]	0.845464 [0.845328, 0.845594]	0.001674 [0.001673, 0.001675]
Threshold-pool attack prior, 10%	20 samples	0.965152 [0.965087, 0.965208]	0.936368 [0.936238, 0.936479]	0.007092 [0.007077, 0.007106]
Remove one node and refit	5	0.984364–0.985264	0.987243–0.993426	0.031886–0.041180
LightGBM family only (7-dimensional)	1	0.983670	0.987157	0.035468
XGBoost family only (7-dimensional)	1	0.982268	0.987619	0.041419

matched controls separate that effect from dimensionality and group count more directly than the family-mean comparison alone.

In a manufacturing network, this interface is most relevant when production segments or gateway generations may retain different tree learners but can be evaluated on the same governed reference flows. Its practical value is response-level interoperability and a traceable decision interface; the experiments do not show that one grouping will remain optimal after equipment, production recipes, or network policies change.

Raw concatenation and the full summary had almost identical mean F1. In this five-node setting, the 14 symmetric statistics therefore retained nearly all useful response information while removing dependence on node order and K. Tiny Deep Sets, despite its 41 parameters, did not improve the result; these data favor explicit statistics over extra meta-layer capacity. That finding is not a rejection of set networks in general. For $K \neq 5$, five quantiles must interpolate between or compress more responses, so their adequacy still depends on the empirical response distribution.

The interface complements the collaboration paradigms reviewed in Section 2. Shared subnetworks, prototypes, and distillation assume a parameter backbone, semantic prototypes, or a unified student, whereas RA-LADS retains usable tree models and learns in reference coordinates. Prior IIoT studies show that device semantics and client distributions alter the collaboration problem[32, 33]; neither supplies common parameter coordinates for heterogeneous trees. The engineering case here is the auditable fixed-dimensional interface and one-time deployment topology, not universal superiority over federated alternatives.

Compact LightGBM trained on the reference pool already achieved F1 of 0.983887. The complete method added only 0.000626. Node-model responses should therefore be credited with a small increment beyond a strong reference learner, not with the entire detection capability. A different trade-off appeared in the one-model-per-node setting: F1 was 0.983865 at roughly half the bundle size. Normalizing over the available family members can be a practical lightweight option when each node can support only one tree family.

5.2. Joint Trade-Off Between Detection Performance and System Cost

Maximum F1 and fixed FAR budgets produced different rankings. The full summary recalled 0.658 of attacks under a 0.1% budget. At 0.5%, recall rose to 0.909, yet the operating point still implied about 5,014 false alarms per million benign flows. Differences from a single LightGBM were not significant between 0.1% and 2% FAR; reproducible F1 and recall gains appeared only at 5%. Prior adjustment sharpens the warning: with a 0.1% attack base rate, prior-adjusted PPV was only about 0.40 even when FAR was also near 0.1%. Site prevalence, traffic volume, and alert capacity must define an acceptable operating point before models are compared.

Node-reliability weighting did not provide a benefit commensurate with its complexity. Composite-quality weighting differed from equal weighting in F1 by only 4.16×10^{-5} and was not significant after multiplicity correction; equal weighting was therefore retained as

the more reproducible default. Random reference-label flips also produced small changes, but this result shows only that the linear meta-layer is insensitive to uniform symmetric noise. It does not replace adversarial evaluation of trusted-root concepts or robust aggregation mechanisms reviewed in Section 2.

System measurements quantified growth beyond the fixed-dimensional interface. In the five-seed node-count scan, F1 intervals overlapped from $K = 2$ to $K = 20$, while mean uncompressed bundle size per gateway increased from 319.8 KB to 3,045.1 KB and complete-test inference time from 3.905 s to 36.964 s. For the five-node compressed bundle, single-flow p95 latency was 1.524 ms on the Apple M3/arm64 Python software stack. The measurement excludes acquisition, feature extraction, energy use, network updates, and sustained load. It is therefore a desktop-class arm64 software profile rather than a benchmark of an operational industrial gateway.

For an industrial rollout, the excluded stages are precisely where production constraints must be tested: traffic acquisition, feature computation under concurrent gateway load, maintenance-window updates, fail-safe rollback, and energy or thermal limits. The reported latency and bundle sizes can therefore screen candidate hardware, but they cannot serve as production-line acceptance evidence.

5.3. Scope of Applicability and Study Boundaries

Representative labeled reference data are a requirement, not a minor convenience. Even the smallest ratio leaves 33,881 labeled flows in each of the meta-learning and threshold-tuning pools, so the 0.5% result is not evidence of few-shot learning. The attack-prior experiments also move both recall and FAR when the base rate changes. In operation, monitoring the reference distribution, updating thresholds, and checking alert capacity belong to the same maintenance process.

When no curated labeled archive exists, commissioning cost includes analyst review, coverage of normal production phases and rare attacks, reconciliation of rule definitions, and repeated review after drift. A staged process could prioritize uncertain or operationally critical flows for active-learning review, use rules or signatures only as auditable provisional pseudo-labels, and apply semi-supervised initialization before replacing provisional labels with reviewed records. These measures may reduce initial review volume but can propagate signature gaps or confirmation bias. They are therefore candidate commissioning strategies, not validated substitutes for the mutually exclusive labeled pools evaluated here.

In a manufacturing deployment, maintenance should therefore be scoped to a specific site and operating period rather than to an aggregate benchmark split. Product changeovers, controller firmware updates, and network resegmentation may alter benign traffic, while the threat environment may change the effective attack base rate; either shift would trigger reference-set review, threshold recalibration, and a staged model rollout.

An operational update policy should combine three triggers: scheduled review at site-defined maintenance intervals; distribution-triggered review when governed reference or benign-response statistics cross a prevalidated drift limit; and alarm-triggered review when labeled audit samples show sustained FAR or PPV deviation, or when alert volume exceeds the site budget. A trigger should initiate reference-set review, refitting of the meta-model and thresholds, acceptance testing on a later time block, signed staged rollout, and rollback if alert or resource limits fail. Base models, rather than only the threshold, require retraining when feature availability or node-response distributions change. This procedure is deployment guidance; online drift handling was not evaluated in the present experiments.

After refitting the meta-model and threshold, node-dropout tests retained similar F1. Losing a whole learner family instead required the 7-dimensional single-family mode. Both are static maintenance procedures, not evidence of online disconnection tolerance. The public data offer no reliable device, site, or temporal identity; Dirichlet label allocation, quantity skew, and `L7_PROTO`-sorted partitions isolate statistical factors only. Each external dataset relocks features and thresholds inside its own training group. The result is per-dataset reproduction, not zero-shot transfer.

A definitive external-validity study therefore requires frozen, time-blocked evaluation across multiple identified sites or factories, including end-to-end acquisition and operator workflow. Until such data are available, the current evidence demonstrates per-dataset retraining under public benchmarks only.

Taken together, RA-LADS should be screened for use only when (i) node-response distributions contain stable, distinguishable structure beyond what a strong reference-pool learner captures; (ii) sufficiently large, representative, mutually exclusive labeled reference pools can be governed; (iii) the site-selected operating point produces a material benefit over a lightweight model; (iv) each gateway can store and execute the complete bundle and support recalibration after change; and (v) model distribution and participation satisfy the stated security assumptions. In this study, corrected gains over single LightGBM appeared at 5% FAR but not at 0.1%–2%; 5% is an empirical boundary of these data, not a universal factory threshold.

Model exchange provides no formal privacy or security guarantee. The honest-participant assumption does not cover membership inference, model replacement, targeted reference-pool poisoning, or coordinator compromise. An adversary may infer training membership even with access only to model outputs[34], and a malicious participant may implant a backdoor in the joint model

through model replacement[35]. Time-blocked traffic from operational sites, low-power gateways, energy consumption, sustained load, model signing, and update recovery all require independent validation. In addition, following recommendations for comparisons over multiple datasets, we use independent seeds as paired units and apply Holm correction to multiple comparisons[36]. This practice improves the auditability of conclusions about randomness but cannot substitute for external validity supported by cross-site and cross-time samples.

6. Conclusion

For networked manufacturing, RA-LADS addresses collaboration among production-cell gateways that may differ by equipment generation or local model choice but can be evaluated on a governed common reference pool. RA-LADS aligns heterogeneous tree models through common reference inputs, learner grouping, and symmetric order statistics. Across 20 seeds, structured grouping performed better than pooling every model into one set. The 14-dimensional summary matched node-wise raw concatenation while remaining invariant to node order and fixed in schema. A change in node count still calls for meta-model retraining and threshold recalibration. Comparisons with a strong reference-pool learner, 41-parameter Tiny Deep Sets, and one model per node narrow the attributable gain: node responses add a small amount beyond the reference learner, rather than replacing the contribution of reference labels or relying on second-stage capacity.

The remaining experiments define where that gain is useful. At stringent FAR budgets of 0.1%–2%, a single LightGBM was equally competitive; multiplicity-adjusted F1 and recall gains for the full summary appeared only at 5% FAR. External datasets showed no universal advantage, and storage and inference costs grew approximately linearly with node count despite the fixed input schema. RA-LADS is therefore an auditable interface for heterogeneous tree collaboration, not a universal low-false-alarm or zero-shot-transfer solution.

A manufacturing deployment should choose between the distribution summary and a lightweight single model from its own attack base rate, alert-handling capacity, and gateway resources, not by copying an operating point from a high-prevalence public benchmark. In networked manufacturing, that decision should also be tied to the target production period, maintenance windows, and the operational consequences of interrupting or investigating a production asset. A cautious plant rollout would first confirm that labeled reference traffic represents the target production period, derive the FAR budget from local alert capacity, profile the complete bundle on the intended gateway, and retain a lightweight single-model option when the full summary provides no operating-point benefit.

Future work should first test frozen cross-temporal and cross-site protocols on time-identified factory traffic, then measure end-to-end latency, sustained load, energy consumption, and recovery on low-power industrial gateways. A second direction is to evaluate label-efficient reference-pool commissioning, drift-triggered meta-model and threshold updates, signed staged distribution, and targeted contamination of the reference pool.

Acknowledgments. This work was supported by the University Scientific Research Project of the Anhui Provincial Department of Education (Natural Science; Grant No. 2023AH052246), the Anhui Provincial University Natural Science Research Project (Innovation Team Program No. 2025AHGXZK10029; "AI + Multi-Scenario Intelligent Decision-Making and Optimization Innovation Team"), and the University-Level Key Discipline of Electronic Information at Suzhou University (2024XK04).

Ethics and Competing Interests

Only publicly released network-traffic benchmarks were used, and source and destination IPv4 addresses were removed before modeling. The work involved no human participants, personal interventions, or animal experiments, so ethics approval and informed consent were not applicable. The authors declare no competing interests related to this study.

References

[1] Sarhan M, Layeghy S, Moustafa N, Portmann M. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In: Big Data Technologies and Applications. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering; 2021. p. 117-35.

[2] Sarhan M, Layeghy S, Moustafa N, Gallagher M, Portmann M. Feature Extraction for Machine Learning-Based Intrusion Detection in IoT Networks. Digital Communications and Networks. 2024;10(1):205-16.

[3] Mirsky Y, Doitshman T, Elovici Y, Shabtai A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In: Network and Distributed System Security Symposium; 2018. .

[4] Barradas D, Santos N, Rodrigues L, Signorello S, Ramos FMV, Madeira A. FlowLens: Enabling Efficient Flow Classification for ML-Based Network Security Applications. In: Network and Distributed System Security Symposium; 2021. .

[5] Xu H, Wu D, Lu Y, Lu J, Zeng H. Models on the Move: Towards Feasible Embedded AI for Intrusion Detection on Vehicular CAN Bus. In: 2024 USENIX Annual Technical Conference; 2024. p. 1049-63.

[6] McMahan HB, Moore E, Ramage D, Hampson S, Agüera y Arcas B. Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. vol. 54 of Proceedings of Machine Learning Research; 2017. p. 1273-82.

[7] Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V. Federated Optimization in Heterogeneous Networks. In: Proceedings of Machine Learning and Systems. vol. 2; 2020. p. 429-50.

[8] Karimireddy SP, Kale S, Mohri M, Reddi S, Stich S, Suresh AT. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In: Proceedings of the 37th International Conference

on Machine Learning. vol. 119 of Proceedings of Machine Learning Research; 2020. p. 5132-43.
[9] Wang J, Liu Q, Liang H, Joshi G, Poor HV. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In: Advances in Neural Information Processing Systems 33; 2020. p. 7611-23.
[10] Kairouz P, McMahan HB, Avent B, et al. Advances and Open Problems in Federated Learning. Foundations and Trends in Machine Learning. 2021;14(1–2):1-210.
[11] Khan LU, Saad W, Han Z, Hossain E, Hong CS. Federated Learning for Internet of Things: Recent Advances, Taxonomy, and Open Challenges. IEEE Communications Surveys & Tutorials. 2021;23(3):1759-99.
[12] Lai F, Dai Y, Singapuram S, Liu J, Zhu X, Madhyastha H, et al. FedScale: Benchmarking Model and System Performance of Federated Learning at Scale. In: Proceedings of the 39th International Conference on Machine Learning. vol. 162 of Proceedings of Machine Learning Research; 2022. p. 11814-27.
[13] Wolpert DH. Stacked Generalization. Neural Networks. 1992;5(2):241-59.
[14] Breiman L. Random Forests. Machine Learning. 2001;45(1):5-32.
[15] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016. p. 785-94.
[16] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: Advances in Neural Information Processing Systems 30; 2017. p. 3146-54.
[17] Zaheer M, Kottur S, Ravanbakhsh S, Poczos B, Salakhutdinov RR, Smola A. Deep Sets. In: Advances in Neural Information Processing Systems 30. vol. 30; 2017. .
[18] Lee J, Lee Y, Kim J, Kosiorek A, Choi S, Teh YW. Set Transformer: A Framework for Attention-Based Permutation-Invariant Neural Networks. In: Proceedings of the 36th International Conference on Machine Learning. vol. 97 of Proceedings of Machine Learning Research; 2019. p. 3744-53.
[19] Guo C, Pleiss G, Sun Y, Weinberger KQ. On Calibration of Modern Neural Networks. In: Proceedings of the 34th International Conference on Machine Learning. vol. 70 of Proceedings of Machine Learning Research; 2017. p. 1321-30.
[20] Li Q, He B, Song D. Model-Contrastive Federated Learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 10713-22.
[21] Li T, Hu S, Beirami A, Smith V. Ditto: Fair and Robust Federated Learning Through Personalization. In: Proceedings of the 38th International Conference on Machine Learning. vol. 139 of Proceedings of Machine Learning Research; 2021. p. 6357-68.
[22] Li X, Jiang M, Zhang X, Kamp M, Dou Q. FedBN: Federated Learning on Non-IID Features via Local Batch Normalization. In: International Conference on Learning Representations; 2021. .
[23] Zhang J, Li Z, Li B, Xu J, Wu S, Ding S, et al. Federated Learning with Label Distribution Skew via Logits Calibration. In: Proceedings of the 39th International Conference on Machine Learning. vol. 162 of Proceedings of Machine Learning Research; 2022. p. 26311-29.
[24] Diao E, Ding J, Tarokh V. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. In: International Conference on Learning Representations; 2021. .
[25] Tan Y, Long G, Liu L, Zhou T, Lu Q, Jiang J, et al. FedProto: Federated Prototype Learning across Heterogeneous Clients. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36; 2022. p. 8432-40.
[26] Cao X, Fang M, Liu J, Gong NZ. FLTrust: Byzantine-Robust Federated Learning via Trust Bootstrapping. In: Network and Distributed System Security Symposium; 2021. .
[27] Lin T, Kong L, Stich SU, Jaggi M. Ensemble Distillation for Robust Model Fusion in Federated Learning. In: Advances in Neural Information Processing Systems 33. vol. 33; 2020. p. 2351-63.
[28] Zhu Z, Hong J, Zhou J. Data-Free Knowledge Distillation for Heterogeneous Federated Learning. In: Proceedings of the 38th International Conference on Machine Learning. vol. 139 of Proceedings of Machine Learning Research; 2021. p. 12878-89.
[29] Shen J, Yang W, Chu Z, Fan J, Niyato D, Lam KY. Effective Intrusion Detection in Heterogeneous Internet-of-Things Networks via Ensemble Knowledge Distillation-Based Federated Learning. In: 2024 IEEE International Conference on Communications; 2024. p. 2034-9.
[30] Li B, Wu Y, Song J, Lu R, Li T, Zhao L. DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber–Physical Systems. IEEE Transactions on Industrial Informatics. 2021;17(8):5615-24.
[31] Liu Y, Kumar N, Xiong Z, Lim WYB, Kang J, Niyato D. Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things. In: 2020 IEEE Global Communications Conference; 2020. p. 1-6.
[32] Nguyen TD, Marchal S, Miettinen M, Fereidooni H, Asokan N, Sadeghi AR. DÏoT: A Federated Self-Learning Anomaly Detection System for IoT. In: 2019 IEEE 39th International Conference on Distributed Computing Systems; 2019. p. 756-67.
[33] Liu T, Dib O. Advanced Intrusion Detection for IoT Devices Using Federated Deep Learning. International Journal of Information Security. 2026;25(1):19.
[34] Shokri R, Stronati M, Song C, Shmatikov V. Membership Inference Attacks Against Machine Learning Models. In: 2017 IEEE Symposium on Security and Privacy; 2017. p. 3-18.
[35] Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How To Backdoor Federated Learning. In: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. vol. 108 of Proceedings of Machine Learning Research; 2020. p. 2938-48.
[36] Demšar J. Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research. 2006;7:1-30.