CARAT: CLASS-AWARE ROBUST AGGREGATION FOR MODEL-POISONING-RESILIENT FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning is vulnerable to model-poisoning attacks because the server must aggregate client updates without inspecting client data. Existing robust aggregation rules often rely on update geometry, coordinate-wise robust statistics, or a single trusted reference direction, but these signals can confuse malicious behavior with benign client heterogeneity under non-i.i.d. data. We propose **CARAT**, a class-aware robust aggregation rule that uses labeled server-side examples to sample hidden, class-balanced tasks and score candidate updates by task-level loss reduction. CARAT combines these certificates with robust clipping, common-radius evaluation, coordinate-rank anomaly penalties, and capped-simplex weight optimization. We evaluate CARAT over five seeds on CIFAR-10/VGG19 and CIFAR-100/ResNet18 with 10% malicious clients, three non-i.i.d. regimes, and five attacks. Across the resulting 30 attacked dataset–heterogeneity–attack settings, CARAT achieves the highest defensive accuracy in 12 settings and the second-highest in 17. It is consistently stable on CIFAR-100, whereas CIFAR-10 exposes seed-level collapses under MinSum at $\alpha = 0.5$ and under MinMax and MinSum at $\alpha = 0.1$. A CARAT-only sweep from 10% to 40% malicious clients shows declining attack-averaged accuracy as the fraction rises. The evidence therefore supports conditional competitiveness at the evaluated 10% threat level, not universal robustness.

1 INTRODUCTION

Federated learning (FL) enables many clients to train a shared model without centralizing their private data (McMahan et al., 2017; Bonawitz et al., 2017; Kairouz et al., 2021). This distributed training paradigm is attractive for privacy- and governance-sensitive applications, but it also makes aggregation a security-critical operation. At each round, the server must update the global model from client messages whose underlying data and training process cannot be directly inspected. As a result, the aggregation rule becomes the main line of defense when some clients submit harmful updates.

Model-poisoning attacks exploit this trust bottleneck by replacing benign client updates with crafted messages. These attacks can be targeted, such as backdoor and model-replacement attacks (Bagdasaryan et al., 2020; Xie et al., 2020a; Wang et al., 2020), or untargeted, where the goal is to degrade clean test performance and destabilize collaborative training (Baruch et al., 2019; Fang et al., 2020; Shejwalkar & Houmansadr, 2021). This paper focuses on the untargeted setting. The core challenge is not only that malicious updates can be arbitrary, but also that benign updates are naturally diverse under non-i.i.d. client data. A geometrically unusual update may therefore be either an attack or a legitimate consequence of client heterogeneity (Zhao et al., 2018; Li et al., 2020; Karimireddy et al., 2020; Zhu et al., 2021).

Existing Byzantine-robust defenses address this threat through geometric selection, coordinate-wise robust statistics, or auxiliary validation, but their trust signals can be too coarse for heterogeneous FL. Krum and MultiKrum (Blanchard et al., 2017) select centrally located updates, Bulyan (El Mhamdi et al., 2018) combines selection with a second robust aggregation stage, coordinate-wise median (Yin et al., 2018) suppresses marginal outliers, and RFA (Pillutla et al., 2022) uses a geometric-median

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

objective. These rules can confuse benign dispersion with malicious deviation when client data are non-i.i.d. Auxiliary-information defenses partly address this limitation, but FLTrust (Cao et al., 2021) compresses trusted data into a single reference direction and FLDetector (Zhang et al., 2022) relies on historical behavior prediction. Such signals may miss attacks whose harm is concentrated on a subset of classes or semantic regions. The resulting open problem is how to validate client updates by their task-level usefulness rather than only by their location in update space.

We propose **CARAT**, a class-aware robust aggregation rule that validates updates through hidden, class-balanced tasks sampled from a small labeled server reference set. The key idea is that a benign non-i.i.d. update need not be geometrically central, but it should provide useful evidence on semantic validation tasks that are hidden from clients. CARAT first places candidate updates in a common scoring geometry through robust clipping and common-radius evaluation. It then measures task-level loss reduction to form hidden-task certificates, penalizes coordinate-rank anomalies, and solves a capped-simplex weight optimization problem with temporal regularization. The final update is computed from clipped client updates, so the server can preserve useful benign diversity while limiting updates that are semantically harmful or structurally extreme. This design explicitly targets settings where a modest labeled server reference set is available; it is not intended to be a fully data-free defense.

We evaluate CARAT on CIFAR-10 (Krizhevsky, 2009) with VGG19 (Simonyan & Zisserman, 2015) and CIFAR-100 with ResNet18 (He et al., 2016) under five untargeted model-poisoning attacks, three Dirichlet non-i.i.d. regimes, and a 10% malicious-client ratio. Across the 30 attacked dataset–heterogeneity–attack settings in the completed five-seed comparison, CARAT achieves the highest defensive accuracy in 12 settings and the second-highest in 17. CIFAR-100 remains stable across these settings, but CIFAR-10 reveals chance-level seeds under MinSum at $\alpha = 0.5$ and under MinMax and MinSum at $\alpha = 0.1$. A CARAT-only sweep to 40% malicious clients shows declining attack-averaged accuracy. We therefore make a bounded claim: task-conditioned validation is competitive at the evaluated 10% comparison point, while ratio-matched baseline comparisons under larger attacker populations and stronger adaptive attacks remain open. Our main contributions are as follows.

- We introduce hidden-task certificates for robust FL aggregation, using class-balanced server-side validation tasks to evaluate whether each candidate update improves semantic slices of the learning problem.
- We develop the CARAT aggregation objective, which combines hidden-task utility, coordinate-rank anomaly penalties, capped per-client influence, and temporal regularization into a single robust weighting rule.
- We provide an auditable five-seed evaluation on CIFAR-10/VGG19 and CIFAR-100/ResNet18 against public robust-aggregation baselines under five attacks and three non-i.i.d. partitions, plus a CARAT-only attacker-fraction sweep, with explicit reporting of observed failures.

2 Related Work

Model-poisoning attacks. Model-poisoning attacks directly manipulate client model updates before aggregation; in this paper we focus on untargeted availability attacks that prevent convergence or degrade clean test performance. Early Byzantine and distributed-learning work studied large-noise attacks in Byzantine-tolerant gradient descent (Blanchard et al., 2017), sign-flipping attacks in asynchronous Byzantine SGD (Damaskinos et al., 2018), and local model manipulation in federated learning (Bhagoji et al., 2019), establishing that robust aggregation must tolerate arbitrary client messages rather than only stochastic training noise. Later attacks became substantially more adaptive. ALIE (Baruch et al., 2019) exploits benign gradient variance to remain statistically plausible, IPM (Xie et al., 2020b) reverses progress through inner-product manipulation, FangAttack (Fang et al., 2020) optimizes local model poisoning against specific robust aggregators, and MinMax/MinSum (Shejwalkar & Houmansadr, 2021) search for malicious updates that satisfy distance- or sum-based stealth constraints while damaging the global model. Mimic-style attacks (Karimireddy et al., 2022) further show that heterogeneity itself can be weaponized by making poisoned updates resemble legitimate client drift. Targeted backdoor and model-replacement attacks form an important adjacent threat family, including model replacement (Bagdasaryan et al., 2020), distributed backdoor attacks

(Xie et al., 2020a), and tail-class backdoors (Wang et al., 2020), but CARAT's empirical claim is deliberately limited to untargeted model poisoning.

Defenses against model poisoning. Defenses against model poisoning differ mainly in the trust signal used for aggregation. Classical Byzantine-robust rules rely on geometry or coordinate-wise statistics: Krum and MultiKrum (Blanchard et al., 2017) select centrally located updates, Bulyan (El Mhamdi et al., 2018) combines selection with a second robust aggregation stage, coordinate-wise median (Yin et al., 2018) suppresses marginal outliers, and RFA (Pillutla et al., 2022) uses a geometric-median objective. Subsequent defenses add suspicion scoring through Zeno (Xie et al., 2019), redundant gradients through DRACO (Chen et al., 2018), regularized aggregation through RSA (Li et al., 2019), divide-and-conquer filtering through DnC (Shejwalkar & Houmansadr, 2021), history-aware clipping through CenteredClipping (Karimireddy et al., 2021), sign-pattern filtering through SignGuard (Xu et al., 2022), latent-space screening through FLARE (Wang et al., 2022), and layer-adaptive sparsification through LASA (Xu et al., 2025). Because non-i.i.d. data can make benign updates geometrically dispersed, related work also studies optimization and robustness under heterogeneity through FedProx (Li et al., 2020), SCAFFOLD (Karimireddy et al., 2020), resampling (He et al., 2021), bucketing (Karimireddy et al., 2022), local-SGD analysis (Data & Diggavi, 2021), gradient splitting (Liu et al., 2023), nearest-neighbor mixing (Allouah et al., 2023), label-skew-aware aggregation (Bao et al., 2024), and entropy-guided public-data aggregation (Huang et al., 2024); non-i.i.d. FL surveys provide the broader context (Zhu et al., 2021). Auxiliary-information defenses use trusted data, historical behavior, or diagnostic signals rather than only instantaneous update geometry: AUROR (Shen et al., 2016) screens updates in collaborative learning, FLTrust (Cao et al., 2021) bootstraps trust from a server reference update, FoolsGold (Fung et al., 2020) penalizes repeated sybil-like similarity, and FLDetector (Zhang et al., 2022) predicts suspicious client behavior from history. CARAT follows this auxiliary-information direction but changes the validation principle: instead of trusting one anchor direction, one geometry statistic, or one historical detector, it asks whether each update remains useful across multiple hidden, class-balanced semantic tasks. Our experiments disclose, rather than equate, the server-reference budgets used by CARAT and FLTrust.

3 Preliminaries and Problem Statement

3.1 Federated Learning and Threat Model

We study synchronous FL in which the server must update a global model from client messages without inspecting client data. Let $\mathcal{N} = \{1, \dots, N\}$ be the client population. Client i holds local data $\mathcal{D}_i$ and objective $F_i(w)$, giving the standard weighted objective

$$F(w) = \sum_{i=1}^{N} p_i F_i(w), \qquad p_i = \frac{|\mathcal{D}_i|}{\sum_{j=1}^{N} |\mathcal{D}_j|}. \tag{1}$$

At communication round r, the server broadcasts w_r to selected clients $\mathcal{S}_r$. Each selected client initializes $w_{i,r}^0 = w_r$ and performs K local stochastic-gradient steps,

$$w_{i,r}^{k+1} = w_{i,r}^k - \eta_r \nabla F_i(w_{i,r}^k; \xi_{i,r}^k), \qquad k = 0, \dots, K-1. \tag{2}$$

The returned model message is converted to an update vector

$$\Delta_i^r = w_{i,r}^K - w_r, \tag{3}$$

and a server-side aggregation rule maps the received updates to the next global model,

$$w_{r+1} = w_r + \mathcal{A}\big(\{\Delta_i^r : i \in \mathcal{S}_r\}\big). \tag{4}$$

FedAvg is recovered when $\mathcal{A}$ is the sample-size-weighted mean.

The adversary controls a subset $\mathcal{M}_r \subseteq \mathcal{S}_r$ of selected clients and may replace their benign updates with arbitrary crafted messages $\widetilde{\Delta}_i^r$. We focus on untargeted model poisoning: malicious updates aim to degrade clean test accuracy while remaining plausible enough to evade robust aggregation. This setting is difficult under non-i.i.d. data because benign clients can naturally differ in update direction, norm, sparsity, and class emphasis. A robust aggregator must therefore reduce harmful influence without treating every geometrically atypical update as malicious.

The adversary is assumed to know the training protocol, defense family, and public hyperparameters, but not the realized hidden validation tasks sampled by the server in a given round. This defines CARAT's reference-assisted security boundary: the labeled reference set and task sampler are server-side resources, not public client inputs. Attackers that directly observe the reference examples or optimize against the exact hidden tasks are outside the empirical claim set; CARAT still limits accepted influence through clipping, rank penalties, and capped weights.

3.2 Reference-Assisted Robust Aggregation

CARAT assumes that the server maintains a small labeled reference set $\mathcal{D}^{\mathrm{ref}}$ covering the task label space. From this set it constructs a probe pool $\mathcal{P}$ and samples hidden, class-balanced validation tasks during aggregation. This is a stronger setting than fully data-free robust aggregation, and we treat it as part of the threat model. The distinction from single-anchor trusted-data methods is that CARAT does not compress $\mathcal{D}^{\mathrm{ref}}$ into one trusted update direction; instead, it uses the reference data to instantiate multiple semantic checks that are hidden from clients.

Given the round-level update set $\{\Delta_i^r\}_{i\in\mathcal{S}_r}$ and reference data $\mathcal{D}^{\mathrm{ref}}$, the server constructs nonnegative aggregation weights and an aggregate update that preserve useful benign diversity while limiting poisoned updates. This goal imposes four design requirements. First, update scoring should be scale-controlled so that large-norm messages cannot receive high trust solely by magnitude. Second, trust should depend on task-conditioned utility, not only on distance to other clients. Third, the method should still penalize structurally extreme updates, because a harmful update may look useful on part of the validation signal. Finally, the aggregation weights should have bounded per-client influence and temporal stability across rounds. CARAT implements these requirements through common-geometry scoring, hidden-task certificates, coordinate-rank penalties, and capped-simplex optimization.

4 CARAT

4.1 Overview

CARAT turns the requirements above into the three-stage server-side pipeline shown in Figure 1. First, the server constructs hidden class-balanced tasks from a labeled reference set, making the validation queries semantic and unobserved by clients. Second, CARAT evaluates each candidate update on these hidden tasks after placing updates in a common scoring geometry, producing a certificate matrix of task-level loss reductions. Third, CARAT learns bounded aggregation weights by optimizing worst-task utility under rank-anomaly penalties, temporal stability, and per-client influence caps. The final server update is a weighted average of clipped client updates, not of the normalized evaluation copies.

Appendix A.3 summarizes how each component maps to a concrete failure mode in non-i.i.d. robust aggregation.

4.2 Hidden Tasks and Task-Conditioned Validation

Within a round, CARAT first prepares client updates for task-conditioned validation. We suppress the round superscript in this section. Before semantic evaluation, the server clips each update so that a small number of large-magnitude messages cannot dominate the comparison stage. Let $\|\Delta_i\|_2$ denote the norm of client i's update. In our implementation, the clipping threshold τ_r is estimated by the MAD-based rule

$$\tau_r = \mathrm{median}(\{\|\Delta_i\|_2\}) + k \cdot 1.4826 \cdot \mathrm{MAD}(\{\|\Delta_i\|_2\}), \tag{5}$$

and the clipped update is

$$\bar{\Delta}_i = \min\left(1, \frac{\tau_r}{\|\Delta_i\|_2 + \varepsilon}\right) \Delta_i. \tag{6}$$

CARAT then rescales each clipped update to a common evaluation radius q_r,

$$\tilde{\Delta}_i = q_r \frac{\bar{\Delta}_i}{\|\bar{\Delta}_i\|_2 + \varepsilon}, \tag{7}$$

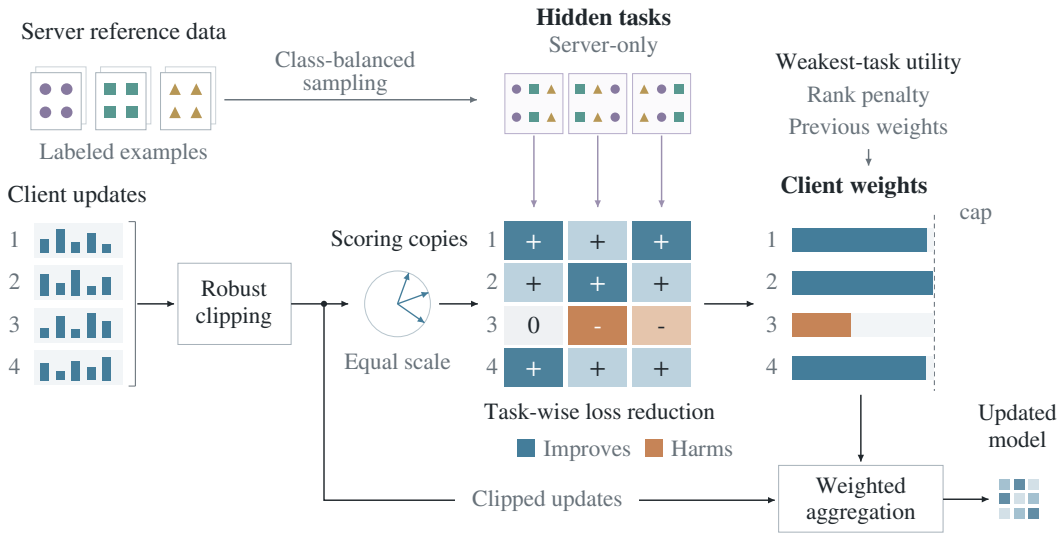


Figure 1: Overview of CARAT. Under non-i.i.d. data, benign diversity can resemble poisoning. CARAT addresses this ambiguity through three server-side stages: (A) hidden class-balanced task construction from a labeled reference set, (B) task-conditioned validation that forms a certificate matrix of loss reductions, and (C) bounded weight learning with rank-anomaly penalties, temporal stability, and per-client caps. The server updates the model using a weighted average of clipped client updates.

where q_r is chosen as a quantile of the positive clipped norms. The normalized vector $\tilde{\Delta}_i$ is used only to evaluate candidate utility; the final aggregation step uses $\bar{\Delta}_i$. This separation makes certificate scoring scale-neutral without changing the conservative clipped update applied to the global model.

CARAT then constructs hidden semantic validation tasks from the server reference data. It samples T hidden tasks from the probe pool $\mathcal{P}$, with each task constructed to be approximately class-balanced. This class-balanced construction is the source of CARAT's class awareness: a malicious update may remain globally plausible while degrading only a subset of classes, whereas repeated balanced tasks make localized failures more visible to the server. For task t, let $\ell_t(w)$ denote the average cross-entropy loss of model w on that task. CARAT defines a *hidden-task certificate*

$$c_{i,t} = \ell_t(w_r) - \ell_t(w_r + \tilde{\Delta}_i). \tag{8}$$

Here, "certificate" denotes an internal task-level utility score, not a formal certified-robustness guarantee. Positive values indicate that client i improves task t relative to the current global model.

Stacking these scores yields a certificate matrix $C \in \mathbb{R}^{n \times T}$. Because tasks may have different intrinsic difficulty and baseline loss scale, CARAT robustly standardizes each column of C before optimization. This preserves within-task ordering while making scores comparable across hidden slices. The matrix gives the server a richer signal than client-to-client similarity alone: an update receives high weight when it remains broadly useful across hidden tasks, not merely because it lies near the geometric center of the client population.

Task utility alone is insufficient, because a malicious client may look useful on part of the semantic signal while remaining structurally extreme in parameter space. CARAT therefore augments hidden-task certificates with a coordinate-rank anomaly score. For a sampled coordinate set $S \subseteq \{1, \ldots, d\}$, it ranks clients coordinate-wise and measures the average normalized distance from the center rank:

$$\gamma_i^{(S)} = \frac{1}{|S|} \sum_{j \in S} \left| \frac{\mathrm{rank}_{i,j}}{n-1} - \frac{1}{2} \right|. \tag{9}$$

CARAT repeats this computation over several random coordinate subsets and aggregates the resulting values robustly. After robust standardization, only positive deviations are retained as the rank-penalty

vector $\gamma \in \mathbb{R}^n$. This signal is intentionally asymmetric: suspicious extremeness is penalized, whereas central ranks are not explicitly rewarded. Together, the certificate matrix and rank penalty separate two notions that are often conflated in robust aggregation: whether an update is useful and whether it is structurally atypical.

4.3 Bounded Weight Learning

CARAT maps task-conditioned validation signals to aggregation weights through a bounded optimization problem on the capped simplex

$$\mathcal{W}(\hat{\rho}) = \left\{ \alpha \in \mathbb{R}^n : \alpha_i \geq 0,\ \sum_{i=1}^{n} \alpha_i = 1,\ \alpha_i \leq \frac{1}{(1-\hat{\rho})n} \right\}, \tag{10}$$

where $\hat{\rho}$ is an estimate of the attacker fraction. The cap prevents any single client from dominating the aggregate and encodes a conservative trust budget at the weight level: even when a client appears useful, it cannot absorb arbitrary influence within one round.

Let α^{prev} be the previous round's weights. CARAT uses the quadratic temporal-variation penalty $\|\alpha - \alpha^{\text{prev}}\|_2^2$ and solves

$$\max_{\alpha \in \mathcal{W}(\hat{\rho})} \ \operatorname*{softmin}_{\beta}(C^\top \alpha) - \lambda_r \gamma^\top \alpha - \lambda_p \|\alpha - \alpha^{\text{prev}}\|_2^2, \tag{11}$$

where

$$\operatorname*{softmin}_{\beta}(s_1, \ldots, s_T) = -\frac{1}{\beta} \log \left(\frac{1}{T} \sum_{t=1}^{T} e^{-\beta s_t} \right). \tag{12}$$

The first term is a smooth approximation to worst-task utility: large β places more emphasis on the weakest hidden tasks, while finite β preserves optimization stability. The second term penalizes structurally extreme updates, and the third term regularizes the solution toward the previous round to reduce unnecessary oscillation. Eq. equation 11 implements a simple decision rule: a candidate should retain influence only if it remains useful across hidden tasks and does not appear as a consistent rank-space outlier.

After optimization, CARAT aggregates the clipped updates:

$$\Delta_{\text{agg}} = \sum_{i=1}^{n} \alpha_i \bar{\Delta}_i. \tag{13}$$

This aggregate is the update applied to the global model. It inherits the robustness benefits of clipping while using weights learned from the richer evaluation geometry defined above. In practice, the main additional computation comes from evaluating n candidate updates over T hidden tasks and computing rank penalties on coordinate subsamples. Probe-model forward passes dominate, while the ranking and optimization stages remain tunable through explicit budget parameters.

A compact round-level algorithm summary is provided in Appendix A.2. Structural properties of feasible weighting, clipped aggregation, malicious-weight influence, hidden-task evaluation geometry, objective behavior, and per-round complexity are deferred to Appendix A.4.

5 Experiments

5.1 Experimental Setup

We evaluate CIFAR-10/VGG19 (Krizhevsky, 2009; Simonyan & Zisserman, 2015) and CIFAR-100/ResNet18 (He et al., 2016) with 20 fully participating clients, Dirichlet non-i.i.d. partitions ($\alpha \in \{1.0, 0.5, 0.1\}$), 200 communication rounds, and four local epochs per round. We test ALIE (Baruch et al., 2019), FangAttack (Fang et al., 2020), MinMax, MinSum (Shejwalkar & Houmansadr, 2021), and Mimic (Karimireddy et al., 2022) with 10% malicious clients, and compare CARAT with NormClipping, MultiKrum (Blanchard et al., 2017), FLTrust (Cao et al., 2021), and FLDetector (Zhang et al., 2022). Unattacked Mean is the clean reference. Appendix B gives the optimizer and method details.

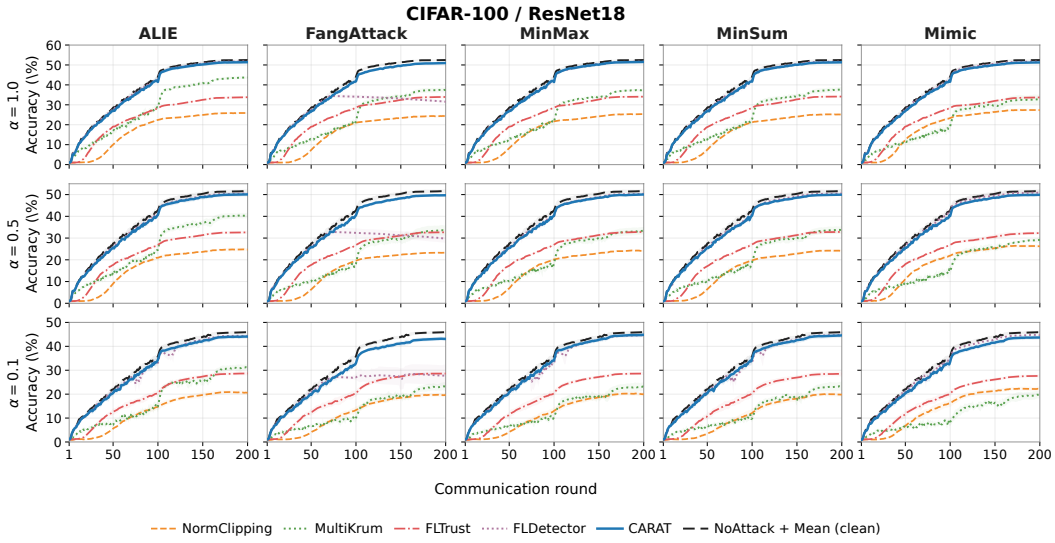


Figure 2: Round-wise test accuracy on CIFAR-100/ResNet18 under five model-poisoning attacks with 10% malicious clients. Rows denote Dirichlet concentrations and columns denote attacks. Curves and bands show the mean and sample standard deviation over seeds 42–46; NoAttack + Mean is the shared clean reference.

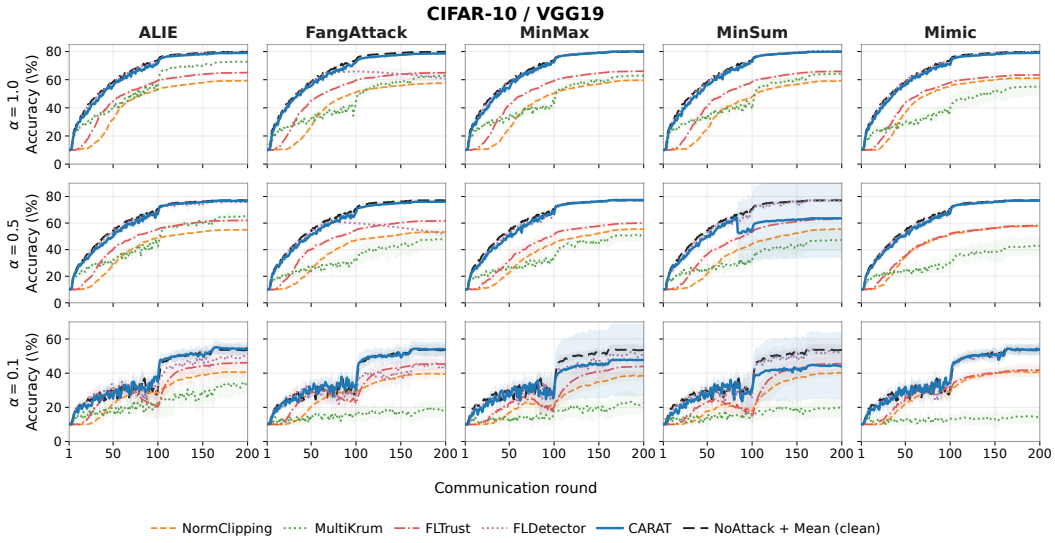


Figure 3: Round-wise test accuracy on CIFAR-10/VGG19 under the same threat model, using the layout and uncertainty convention of Figure 2.

Figures 2 and 3 report means and sample-standard-deviation bands over seeds 42–46; numerical comparisons below use round 200. CARAT uses $T = 8$ hidden tasks and a 512-example server pool, whereas FLTrust uses 100 server examples. Both pools are drawn from the training set without removing those examples from client partitions, so the curves compare the recorded method–reference configurations rather than isolated aggregation rules. The shared NoAttack + Mean curve is excluded from defense rankings.

5.2 Experimental Results

5.2.1 Robustness to Model-Poisoning Attacks

Figure 2 shows that CARAT closely tracks the clean trajectory after the learning-rate transitions, rather than reaching a similar endpoint through a late recovery. On CIFAR-100 with $\alpha \in \{1.0, 0.5\}$, CARAT is the strongest attacked defense under FangAttack and is second under each of the other conventional attacks. In those six second-place cases it trails the strongest defense by only 0.18–0.54 percentage points. It also avoids FLDetector's sharp FangAttack degradation and exceeds MultiKrum and FLTrust throughout both settings. Relative to the NoAttack + Mean references of 52.47% and 51.61%, CARAT's conventional-attack results remain within 0.95–1.91 points.

At $\alpha = 0.1$, CARAT is the strongest attacked defense under FangAttack and MinMax, and is second under ALIE and MinSum by 0.29 and 0.16 points, respectively. Its four means remain within 1.14–2.85 points of the 45.94% clean reference, with seed-level standard deviations below one percentage point. The lower attacker fraction therefore removes the chance-level collapses observed in the archived 20% CARAT runs. This is evidence of stability for the evaluated 10% threat level, not evidence that the same behavior persists as the malicious population grows.

On CIFAR-10 at $\alpha = 1.0$, CARAT is the strongest attacked defense under FangAttack and is second under ALIE, MinMax, and MinSum, with gaps of 0.49, 0.44, and 0.53 points. At $\alpha = 0.5$, it leads ALIE, FangAttack, and MinMax, but one MinSum seed finishes at chance-level 10% accuracy; the resulting round-200 accuracy of $63.58 \pm 29.96\%$ is second to FLDetector and lies 13.49 points below the 77.07% clean reference. At $\alpha = 0.1$, CARAT leads ALIE and FangAttack, while one MinMax seed and one MinSum seed again finish at chance level, producing $47.65 \pm 21.10\%$ and $43.94 \pm 19.28\%$. The widening bands in Figure 3 show when this seed sensitivity emerges instead of only exposing it at round 200.

Mimic copies a fixed benign update rather than optimizing an outlier. CARAT ranks second on CIFAR-100 at all three heterogeneity levels, trailing FLDetector by 0.46–1.08 points. On CIFAR-10, it is second at $\alpha = 1.0$ and leads at $\alpha \in \{0.5, 0.1\}$ with 76.96% and 53.96%. It exceeds NormClipping, MultiKrum, and FLTrust in every Mimic setting, indicating that hidden-task validation retains signal against copied benign updates.

5.2.2 Impact of Client Heterogeneity

On CIFAR-100, CARAT's accuracy averaged across all five attacks is 51.30%, 49.94%, and 44.05% for $\alpha = 1.0$, 0.5, and 0.1, respectively, trailing the corresponding clean references by 1.17, 1.67, and 1.89 points. The 1.36-point decline from $\alpha = 1.0$ to 0.5 grows to 5.89 points from 0.5 to 0.1, but remains consistent across attacks rather than being driven by isolated seed collapses. On CIFAR-10, the corresponding attack averages are 79.38%, 74.13%, and 50.81%, compared with clean references of 79.83%, 77.07%, and 53.53%. The larger cross-setting drops of 5.25 and 23.32 points reflect both stronger heterogeneity and the MinSum/MinMax failures. Client heterogeneity therefore remains an important source of degradation even when the malicious-client fraction is 10%, with a substantially sharper effect on CIFAR-10/VGG19.

5.2.3 Impact of Attacker Fraction

We next vary the malicious-client fraction from 10% to 40% (two to eight of the same 20 clients) while retaining the three Dirichlet settings and five attacks. Figure 4 averages the five attack accuracies within each seed, then reports the mean and sample standard deviation across seeds at round 200. On CIFAR-100, the 10%-to-40% increase reduces CARAT's attack-averaged accuracy from 51.30% to 42.17% at $\alpha = 1.0$, from 49.94% to 34.09% at $\alpha = 0.5$, and from 44.05% to 26.42% at $\alpha = 0.1$. CIFAR-10 shows the same endpoint trend, declining from 79.38% to 62.55%, 74.13% to 56.60%, and 50.81% to 28.15%, respectively. The trend is not strictly monotone: the CIFAR-10 $\alpha = 0.5$ mean improves slightly at 20% because a MinSum seed fails at 10%. Higher attacker fractions generally magnify the impact of heterogeneity, but the plot summarizes attack-specific outcomes and does not imply uniform degradation for every attack. CARAT's weight cap uses the specified attacker fraction, so this sweep changes both the malicious population and the supplied prior; it does not test prior misspecification.

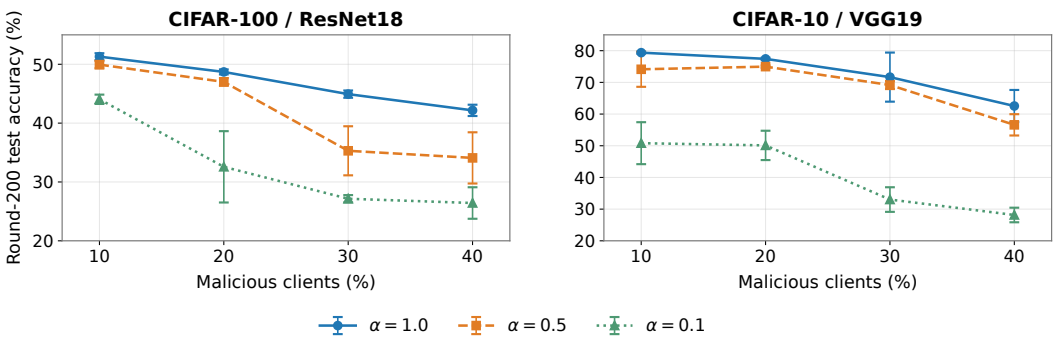


Figure 4: CARAT round-200 accuracy as the malicious-client fraction increases from 10% to 40% with 20 clients. Each point first averages ALIE, FangAttack, MinMax, MinSum, and Mimic within a seed; markers and error bars show the mean and sample standard deviation over seeds 42–46. The attacker-fraction prior is set to the corresponding experimental fraction.

5.2.4 Scope of the Evidence

The main comparison does not cover partial participation, ratio-matched larger-federation baselines, backdoor or data-poisoning attacks, misspecified attacker fractions, or adaptive attackers that infer the hidden tasks. The NoAttack + Mean curve is a clean reference rather than an attacked defense, and the unequal CARAT/FLTrust reference budgets further limit comparison. Figure 4 extends only CARAT to higher attacker fractions, not the baseline comparisons. The separate component study, CARAT-only client-population diagnostic, and earlier three-seed runtime diagnostic are reported in Appendices B.4, B.5, and B.6, respectively. Together with the CIFAR-10 seed-level failures, these boundaries restrict the claim to the evaluated settings.

6 Conclusion

CARAT validates candidate updates through hidden, class-balanced task certificates rather than relying only on geometric centrality or a single trusted reference direction. In the five-seed CIFAR-10/VGG19 and CIFAR-100/ResNet18 comparison with 10% malicious clients, CARAT achieves the highest defensive accuracy in 12 of the 30 attacked settings and the second-highest in 17. CIFAR-100 remains stable across seeds at that ratio, while CIFAR-10 exposes isolated chance-level failures under MinSum and MinMax as heterogeneity increases. A CARAT-only sweep shows decreasing attack-averaged accuracy as the malicious fraction rises to 40%. These observations support conditional competitiveness rather than universal robustness. Disjoint reference data, matched auxiliary-data budgets, ratio-matched baseline comparisons, prior misspecification, and adaptive attacks remain important next steps.

AI use statement

We used OpenAI Codex to assist with discussing and refining experimental protocols and ablations; and polishing portions of the manuscript. The reported measurements were produced by executing the experiments, not generated by the AI tool. Numerical summaries and figures were checked against saved experiment logs where applicable. The authors take responsibility for the final methods, code, analyses, claims, and text.

References

Youssef Allouah, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, and John Stephan. Fixing by mixing: A recipe for optimal byzantine ML under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, pp. 1232–1300, 2023.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948, 2020.

Wenxuan Bao, Jun Wu, and Jingrui He. BOBA: Byzantine-robust federated learning with label skewness. In *International Conference on Artificial Intelligence and Statistics*, pp. 892–900, 2024.

Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, 2019.

Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pp. 634–643, 2019.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 2017.

Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Xiaoyu Cao, Minghong Fang, Jia Liu, Neil Zhenqiang Gong, and Yixin Gong. FLTrust: Byzantine-robust federated learning via trust bootstrapping. In *Network and Distributed System Security Symposium*, 2021.

Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. DRACO: Byzantine-resilient distributed training via redundant gradients. In *International Conference on Machine Learning*, pp. 903–912, 2018.

Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, and Mahsa Taziki. Asynchronous Byzantine machine learning (the case of SGD). In *International Conference on Machine Learning*, pp. 1145–1154, 2018.

Deepesh Data and Suhas Diggavi. Byzantine-resilient high-dimensional SGD with local iterations on heterogeneous data. In *International Conference on Machine Learning*, pp. 2478–2488, 2021.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in Byzantium. In *International Conference on Machine Learning*, 2018.

Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to Byzantine-robust federated learning. In *USENIX Security Symposium*, 2020.

Clement Fung, Chris J. M. Yoon, and Ivan Beschastnikh. The limitations of federated learning in sybil settings. In *International Symposium on Research in Attacks, Intrusions and Defenses*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. Byzantine-robust learning on heterogeneous datasets via resampling. In *International Conference on Learning Representations*, 2021.

Wenke Huang, Zekun Shi, Mang Ye, He Li, and Bo Du. Self-driven entropy aggregation for byzantine-robust heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 20096–20110, 2024.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurelien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. *Advances and Open Problems in Federated Learning*. Now Publishers, 2021.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143, 2020.

Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for Byzantine robust optimization. In *International Conference on Machine Learning*, pp. 5311–5319, 2021.

Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Byzantine-robust learning on heterogeneous datasets via bucketing. In *International Conference on Learning Representations*, 2022.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Liping Li, Wei Xu, Tianyi Chen, Georgios B. Giannakis, and Qing Ling. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *AAAI Conference on Artificial Intelligence*, pp. 1544–1551, 2019.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, 2020.

Yuchen Liu, Chen Chen, Lingjuan Lyu, Fangzhao Wu, Sai Wu, and Gang Chen. Byzantine-robust learning on heterogeneous data via gradient splitting. In *International Conference on Machine Learning*, pp. 21404–21425, 2023.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, 2017.

Krishna Pillutla, Sham M. Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

Virat Shejwalkar and Amir Houmansadr. Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Network and Distributed System Security Symposium*, 2021.

Shiqi Shen, Shruti Tople, and Prateek Saxena. AUROR: Defending against poisoning attacks in collaborative deep learning systems. In *Annual Computer Security Applications Conference*, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In *Advances in Neural Information Processing Systems*, 2020.

Ning Wang, Yang Xiao, Yimin Chen, Yang Hu, Wenjing Lou, and Y. Thomas Hou. FLARE: Defending federated learning against model poisoning attacks via latent space representations. In *ACM Asia Conference on Computer and Communications Security*, pp. 946–958, 2022.

Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. DBA: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020a.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, pp. 6893–6901, 2019.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking Byzantine-tolerant SGD by inner product manipulation. In *Conference on Uncertainty in Artificial Intelligence*, pp. 261–270, 2020b.

Jiahao Xu, Zikai Zhang, and Rui Hu. Achieving Byzantine-resilient federated learning via layer-adaptive sparsified model aggregation. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1508–1517, 2025.

Jian Xu, Shao-Lun Huang, Linqi Song, and Tian Lan. SignGuard: Byzantine-robust federated learning through collaborative malicious gradient filtering. In *IEEE International Conference on Distributed Computing Systems*, pp. 1026–1036, 2022.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, 2018.

Zaixi Zhang, Xiaoyu Xu, Jinyuan Jia, Matthew Jagielski, Gauri Joshi, Mingyi Hong, and Carl A. Gunter. FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, et al. Federated learning with non-IID data. *arXiv preprint arXiv:1806.00582*, 2018.

Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-IID data: A survey. *Neurocomputing*, 465:371–390, 2021.

Appendix organization. The appendix is organized to make each claim auditable. Appendix A gives the round-level algorithm, implementation invariants, and proofs of the structural properties used in the main text. Appendix B gives protocol details, CARAT hyperparameters, the component ablation, client-population scalability results, and runtime diagnostics.

A Algorithmic and Theoretical Details

This appendix provides the technical material needed to reproduce and interpret CARAT. It first fixes the round-level notation and server inputs, then gives executable pseudocode, implementation invariants, and structural propositions for the scoring and weighting rule. These results are deliberately stated as invariants and bounds for the proposed aggregation procedure, not as a complete Byzantine-robustness guarantee against arbitrary adaptive attackers.

A.1 Notation and Round Inputs

All quantities in this appendix refer to one communication round r unless stated otherwise. The server receives $n = |\mathcal{S}_r|$ client messages and converts them into vector updates Δ_i^r. CARAT keeps three versions of each update:

$$\Delta_i^r \quad \text{(raw client update)}, \qquad \bar{\Delta}_i^r \quad \text{(clipped update)}, \qquad \tilde{\Delta}_i^r \quad \text{(common-radius scoring copy)}.$$

Only $\bar{\Delta}_i^r$ is used in the final server update. The normalized copy $\tilde{\Delta}_i^r$ is used only to evaluate hidden-task utility, so the validation score is less sensitive to residual norm differences after clipping.

The server maintains a labeled reference set $\mathcal{D}^{\mathrm{ref}}$ and derives from it a working probe pool $\mathcal{P}$. Each hidden task is a small approximately class-balanced subset sampled from $\mathcal{P}$. The raw certificate matrix is $C^{\mathrm{raw}} \in \mathbb{R}^{n \times T}$, where T is the number of hidden tasks and $C_{i,t}^{\mathrm{raw}} = c_{i,t}$ is the loss reduction obtained by candidate update i on hidden task t. After robust column-wise standardization, the optimization uses $C \in \mathbb{R}^{n \times T}$. The vector $\gamma \in \mathbb{R}_{\geq 0}^{n}$ denotes coordinate-rank anomaly penalties. The vector α^{prev} is the previous round's aggregation weight vector and acts as a temporal prior.

A.2 CARAT Round Algorithm

Algorithm 1 summarizes one CARAT server aggregation round. The algorithm is server-side: clients do not observe the realized hidden tasks sampled from $\mathcal{P}$.

A.3 Implementation Invariants

This subsection records implementation choices that affect reproducibility but are too low-level for the main method section. The purpose is to make the server-side scoring rule auditable, not to introduce additional assumptions beyond those stated in the threat model.

Component rationale. Table 1 summarizes why each CARAT component is included. The design is not intended as an arbitrary stack of heuristics: each term targets a distinct failure mode that appears when heterogeneous benign drift and malicious manipulation coexist.

Persistent state and initialization. CARAT maintains two state variables across rounds: the probe pool $\mathcal{P}$ derived from $\mathcal{D}^{\mathrm{ref}}$, and the previous-round weight vector α^{prev}. At the first round, α^{prev} is initialized to the uniform vector. This makes the first aggregation unbiased across clients while allowing later rounds to use temporal consistency once weight trajectories have been observed.

Reference-pool construction and hidden-task sampling. Let $\mathcal{P}_c \subseteq \mathcal{P}$ denote the subset of probe examples with class label c. A hidden task is constructed by sampling a small number of examples from each available class pool and concatenating them into one balanced micro-task. If the desired number of samples per class is not available without replacement, CARAT falls back to replacement and, if needed, to a random subset of the probe pool rather than aborting the round. This fail-soft design preserves the intended semantic diversity of the task sampler while ensuring that the server always obtains a well-defined certificate matrix.

Algorithm 1 CARAT server aggregation at round r

Require: Current global model w_r, client messages $\{u_i^r\}_{i=1}^n$, previous-round weights α^{prev}, server reference set $\mathcal{D}^{\mathrm{ref}}$, probe pool $\mathcal{P}$, CARAT hyperparameters
Ensure: Aggregated update Δ_{agg}, updated temporal prior α^{prev}
if $\mathcal{P}$ is uninitialized **then**
  Construct a class-balanced probe pool $\mathcal{P}$ from $\mathcal{D}^{\mathrm{ref}}$
end if
if $r = 1$ **then**
  $\alpha^{\mathrm{prev}} \leftarrow \frac{1}{n}\mathbf{1}$
end if
for $i = 1$ to n **do**
  Convert u_i^r into a vector-form delta Δ_i^r
end for
Compute the clipping threshold τ_r from $\{\|\Delta_i^r\|_2\}_{i=1}^n$
for $i = 1$ to n **do**
  $\bar{\Delta}_i^r \leftarrow \min\Big(1, \frac{\tau_r}{\|\Delta_i^r\|_2+\varepsilon}\Big)\,\Delta_i^r$
end for
Choose the evaluation radius q_r from a quantile of positive clipped norms
for $i = 1$ to n **do**
  $\tilde{\Delta}_i^r \leftarrow q_r \bar{\Delta}_i^r / (\|\bar{\Delta}_i^r\|_2 + \varepsilon)$
end for
Sample T hidden tasks from $\mathcal{P}$ using class-balanced sampling
for $t = 1$ to T **do**
  Evaluate the baseline loss $\ell_t(w_r)$
end for
for $i = 1$ to n **do**
  for $t = 1$ to T **do**
    $c_{i,t} \leftarrow \ell_t(w_r) - \ell_t(w_r + \tilde{\Delta}_i^r)$
  end for
end for
Robustly standardize certificate columns to obtain C
Compute rank penalties γ from repeated coordinate subsamples of $\{\bar{\Delta}_i^r\}_{i=1}^n$
$\alpha^{(0)} \leftarrow \alpha^{\mathrm{prev}}$
for $k = 0$ to $K - 1$ **do**
  $z \leftarrow C^\top \alpha^{(k)}$
  $\pi \leftarrow$ softmin task weights induced by z and β
  $g \leftarrow C\pi - \lambda_r \gamma - 2\lambda_p(\alpha^{(k)} - \alpha^{\mathrm{prev}})$
  $\alpha^{(k+1)} \leftarrow \Pi_{\mathcal{W}(\hat{\rho})}\big(\alpha^{(k)} + \eta g\big)$
end for
$\alpha^\star \leftarrow \alpha^{(K)}$ and $\alpha^{\mathrm{prev}} \leftarrow \alpha^\star$
$\Delta_{\mathrm{agg}} \leftarrow \sum_{i=1}^n \alpha_i^\star \bar{\Delta}_i^r$
return $\Delta_{\mathrm{agg}}, \alpha^{\mathrm{prev}}$

Robust score calibration. The raw certificate matrix is transformed column-wise before optimization:

$$C_{i,t} = \frac{C_{i,t}^{\mathrm{raw}} - \mathrm{median}_j(C_{j,t}^{\mathrm{raw}})}{1.4826 \cdot \mathrm{MAD}_j(C_{j,t}^{\mathrm{raw}}) + \varepsilon}.$$

This removes task-specific scale effects while preserving within-task ordering by Proposition A.5. Rank penalties are calibrated similarly. For each coordinate subsample S_s, let

$$\rho_i^{(s)} = \frac{1}{|S_s|} \sum_{j \in S_s} \left| \frac{\mathrm{rank}_{i,j}}{n-1} - \frac{1}{2} \right|.$$

CARAT then computes a standardized rank score

$$z_i^{(s)} = \frac{\rho_i^{(s)} - \mathrm{median}_\ell(\rho_\ell^{(s)})}{1.4826 \cdot \mathrm{MAD}_\ell(\rho_\ell^{(s)}) + \varepsilon},$$

Table 1: Design rationale for the CARAT aggregation pipeline. Each component addresses a different ambiguity in non-i.i.d. robust aggregation.

Component	Failure mode addressed	Role in CARAT
MAD clipping and common-radius scoring	Large-norm or scale-manipulated updates can dominate validation scores, while benign non-i.i.d. clients may naturally have different update norms.	Separates the scoring copy from the final clipped update, making hidden-task evaluation less sensitive to update magnitude.
Hidden class-balanced tasks	A single global score or trusted direction can miss class-local degradation and can confuse semantic usefulness with geometric centrality.	Scores each candidate by loss reduction on multiple hidden semantic slices, producing a task-level certificate matrix.
Robust task standardization	Hidden tasks can have different baseline loss scales and difficulty levels.	Makes certificate columns comparable while preserving within-task ordering.
Coordinate-rank penalty	A client may look useful on part of the reference signal while remaining structurally extreme in update space.	Penalizes consistently outlying coordinate ranks without rewarding unusual centrality.
Capped-simplex weights	Even a high-scoring client should not receive unbounded influence in a Byzantine setting.	Limits per-client weight according to an attacker-fraction prior and bounds direct malicious mass.
Temporal regularization	Round-to-round weight oscillations can overreact to noisy probe evaluations.	Encourages stable trust allocation unless hidden-task evidence supports a change.

and aggregates subsamples through

$$\gamma_i = \max\left(0, \operatorname{median}_s z_i^{(s)}\right).$$

Thus a client is penalized only when it is consistently extreme across repeated coordinate views, not because of a single noisy subsample.

Projected optimization. CARAT initializes the projected optimizer with $\alpha^{(0)} = \alpha^{\mathrm{prev}}$. If

$$z(\alpha) = C^\top \alpha, \qquad \pi_t(\alpha) = \frac{\exp(-\beta z_t(\alpha))}{\sum_{\ell=1}^{T} \exp(-\beta z_\ell(\alpha))},$$

then the ascent direction used in projected updates is

$$g(\alpha) = C\pi(\alpha) - \lambda_r \gamma - 2\lambda_p(\alpha - \alpha^{\mathrm{prev}}).$$

With optimizer step size η, one iteration takes the form

$$\alpha^{(k+1)} = \Pi_{\mathcal{W}(\hat{\rho})}\left(\alpha^{(k)} + \eta g(\alpha^{(k)})\right).$$

The projection maintains feasibility throughout optimization, so every intermediate iterate satisfies nonnegativity, unit-sum, and per-client cap constraints.

Diagnostics. The implementation records learned weights, task scores, softmin task weights, rank penalties, clipping thresholds, evaluation radii, and hidden-task identifiers. These diagnostics distinguish three qualitatively different behaviors: benign but diverse clients that retain moderate weight, harmful clients that lose certificate support across hidden tasks, and structurally extreme clients whose influence is reduced mainly through the rank penalty.

A.4 Proofs and Structural Properties

The following propositions are grouped by the role they play in CARAT. Propositions A.1–A.3 characterize feasible weighted aggregation and bounded influence after clipping. Propositions A.4–A.5 characterize the hidden-task evaluation geometry. Propositions A.6–A.8 characterize the optimization objective, and Proposition A.9 records the per-round cost.

Proposition A.1 (Feasibility and support of the capped simplex)**.** *Let*

$$\mathcal{W}(\hat{\rho}) = \left\{ \alpha \in \mathbb{R}^n : \alpha_i \geq 0,\ \sum_{i=1}^{n} \alpha_i = 1,\ \alpha_i \leq \frac{1}{(1-\hat{\rho})n} \right\},$$

with $\hat{\rho} < 1$. Then $\mathcal{W}(\hat{\rho})$ is non-empty, and every $\alpha \in \mathcal{W}(\hat{\rho})$ satisfies

$$|\operatorname{supp}(\alpha)| \geq (1-\hat{\rho})n.$$

Proof. Let $u = 1/((1-\hat{\rho})n)$. Since $\hat{\rho} < 1$, we have $u \geq 1/n$, so the uniform weight vector is feasible and the set is non-empty. For the support bound, if only k coordinates of α are positive, then

$$1 = \sum_{i=1}^{n} \alpha_i \leq ku = \frac{k}{(1-\hat{\rho})n},$$

which implies $k \geq (1-\hat{\rho})n$. Therefore CARAT can suppress suspicious clients, but cannot collapse the aggregate onto an arbitrarily small subset of participants. □

Proposition A.2 (Norm bound for the final aggregate)**.** *Suppose each clipped update satisfies $\|\bar{\Delta}_i\|_2 \leq \tau_r$ and $\alpha \in \mathcal{W}(\hat{\rho})$. Then the final aggregate*

$$\Delta_{\text{agg}} = \sum_{i=1}^{n} \alpha_i \bar{\Delta}_i$$

satisfies

$$\|\Delta_{\text{agg}}\|_2 \leq \tau_r.$$

Proof. Because $\alpha_i \geq 0$ and $\sum_i \alpha_i = 1$, the aggregate is a convex combination of clipped updates. By the triangle inequality,

$$\|\Delta_{\text{agg}}\|_2 \leq \sum_{i=1}^{n} \alpha_i \|\bar{\Delta}_i\|_2 \leq \sum_{i=1}^{n} \alpha_i \tau_r = \tau_r.$$

Thus the final server step inherits the clipping radius as a hard norm upper bound. □

Proposition A.3 (Influence bound from malicious weight mass)**.** *Let $\mathcal{M}$ be the malicious clients selected in a round and let*

$$q_{\mathcal{M}} = \sum_{i \in \mathcal{M}} \alpha_i$$

be their total CARAT weight. Suppose every clipped update satisfies $\|\bar{\Delta}_i\|_2 \leq \tau_r$. Then the direct malicious contribution obeys

$$\left\| \sum_{i \in \mathcal{M}} \alpha_i \bar{\Delta}_i \right\|_2 \leq q_{\mathcal{M}} \tau_r.$$

If $q_{\mathcal{M}} < 1$ and

$$\Delta_{\mathcal{B}} = \sum_{i \notin \mathcal{M}} \frac{\alpha_i}{1 - q_{\mathcal{M}}} \bar{\Delta}_i$$

denotes the benign-only aggregate obtained by renormalizing the benign weights, then

$$\|\Delta_{\text{agg}} - \Delta_{\mathcal{B}}\|_2 \leq 2 q_{\mathcal{M}} \tau_r.$$

Proof. The first inequality follows from the triangle inequality:

$$\left\| \sum_{i \in \mathcal{M}} \alpha_i \bar{\Delta}_i \right\|_2 \leq \sum_{i \in \mathcal{M}} \alpha_i \|\bar{\Delta}_i\|_2 \leq q_{\mathcal{M}} \tau_r.$$

For the second inequality, write

$$\Delta_{\text{agg}} = (1 - q_{\mathcal{M}}) \Delta_{\mathcal{B}} + \sum_{i \in \mathcal{M}} \alpha_i \bar{\Delta}_i.$$

Therefore

$$\|\Delta_{\text{agg}} - \Delta_{\mathcal{B}}\|_2 \le q_{\mathcal{M}} \|\Delta_{\mathcal{B}}\|_2 + \left\| \sum_{i \in \mathcal{M}} \alpha_i \bar{\Delta}_i \right\|_2 .$$

Since $\Delta_{\mathcal{B}}$ is a convex combination of clipped benign updates, $\|\Delta_{\mathcal{B}}\|_2 \le \tau_r$ by Proposition A.2. Combining this with the first inequality gives the stated bound. □

Proposition A.4 (Directional invariance of certificate scoring)**.** *Suppose two nonzero clipped updates satisfy $\bar{\Delta}_i = a\bar{\Delta}_j$ for some scalar $a > 0$. Then their normalized versions in Eq. equation 7 coincide, i.e.,*

$$\tilde{\Delta}_i = \tilde{\Delta}_j .$$

Consequently, their hidden-task certificates are identical:

$$c_{i,t} = c_{j,t} \qquad \text{for all } t \in \{1, \dots, T\}.$$

Proof. Eq. equation 7 rescales every nonzero clipped update to the common radius q_r. Positive collinearity is therefore removed by normalization:

$$q_r \frac{\bar{\Delta}_i}{\|\bar{\Delta}_i\|_2} = q_r \frac{a\bar{\Delta}_j}{\|a\bar{\Delta}_j\|_2} = q_r \frac{\bar{\Delta}_j}{\|\bar{\Delta}_j\|_2}.$$

Substituting the resulting equality into Eq. equation 8 gives identical task scores. Thus, after clipping, the certificate stage depends on direction rather than residual norm magnitude. □

Proposition A.5 (Task-wise standardization preserves within-task ordering)**.** *Fix a task t and define a standardized certificate score*

$$\hat{c}_{i,t} = \frac{c_{i,t} - m_t}{s_t + \varepsilon},$$

where $m_t \in \mathbb{R}$ and $s_t > 0$ depend only on task t and not on client i. Then for any clients i, j,

$$c_{i,t} \ge c_{j,t} \quad \Longleftrightarrow \quad \hat{c}_{i,t} \ge \hat{c}_{j,t}.$$

Moreover,

$$\hat{c}_{i,t} - \hat{c}_{j,t} = \frac{c_{i,t} - c_{j,t}}{s_t + \varepsilon}.$$

Proof. For a fixed task t, the map $x \mapsto (x - m_t)/(s_t + \varepsilon)$ is affine with strictly positive slope because $s_t + \varepsilon > 0$. Hence it preserves ordering. The difference identity follows by direct subtraction. Therefore column-wise robust standardization calibrates cross-task scale without changing the within-task ranking induced by the raw certificate scores. □

Proposition A.6 (Softmin bounds and gradient form)**.** *Let $s \in \mathbb{R}^T$ and define*

$$\operatorname*{softmin}_{\beta}(s) = -\frac{1}{\beta} \log \left(\frac{1}{T} \sum_{t=1}^{T} e^{-\beta s_t} \right).$$

Then

$$\min_t s_t \le \operatorname*{softmin}_{\beta}(s) \le \min_t s_t + \frac{\log T}{\beta}.$$

Moreover,

$$\lim_{\beta \to \infty} \operatorname*{softmin}_{\beta}(s) = \min_t s_t, \qquad \lim_{\beta \to 0} \operatorname*{softmin}_{\beta}(s) = \frac{1}{T} \sum_{t=1}^{T} s_t.$$

If $z(\alpha) = C^\top \alpha$ and

$$\pi_t(\alpha) = \frac{\exp(-\beta z_t(\alpha))}{\sum_{\ell=1}^{T} \exp(-\beta z_\ell(\alpha))},$$

then

$$\nabla_\alpha \operatorname*{softmin}_{\beta}(C^\top \alpha) = C\pi(\alpha).$$

Proof. Let $m = \min_t s_t$. Since

$$e^{-\beta m}/T \le \frac{1}{T}\sum_{t=1}^{T} e^{-\beta s_t} \le e^{-\beta m},$$

applying $-(1/\beta)\log(\cdot)$ yields

$$m \le \operatorname*{softmin}_{\beta}(s) \le m + \frac{\log T}{\beta}.$$

The limit $\beta \to \infty$ follows from the vanishing gap bound. For $\beta \to 0$, a first-order expansion of the exponential average gives

$$\frac{1}{T}\sum_{t=1}^{T} e^{-\beta s_t} = 1 - \beta\frac{1}{T}\sum_{t=1}^{T} s_t + o(\beta),$$

so the softmin converges to the arithmetic mean. Differentiating the softmin expression with respect to z_t yields the Gibbs weights $\pi_t(\alpha)$, and the chain rule gives $C\pi(\alpha)$. Hence the ascent direction is a convex combination of certificate columns, with more weight placed on tasks whose current aggregate utility is smaller. □

Proposition A.7 (Temporal prior and projected-iterate invariants)**.** *Define*

$$f(\alpha) = \operatorname*{softmin}_{\beta}(C^\top\alpha) - \lambda_r\gamma^\top\alpha - \lambda_p\|\alpha - \alpha^{\mathrm{prev}}\|_2^2.$$

If $\alpha, \alpha' \in \mathcal{W}(\hat{\rho})$ satisfy $C^\top\alpha = C^\top\alpha'$ and $\gamma^\top\alpha = \gamma^\top\alpha'$, then

$$f(\alpha) - f(\alpha') = -\lambda_p\left(\|\alpha - \alpha^{\mathrm{prev}}\|_2^2 - \|\alpha' - \alpha^{\mathrm{prev}}\|_2^2\right).$$

In addition, under projected gradient ascent

$$\alpha^{(k+1)} = \Pi_{\mathcal{W}(\hat{\rho})}\left(\alpha^{(k)} + \eta\nabla f(\alpha^{(k)})\right),$$

every iterate remains in $\mathcal{W}(\hat{\rho})$.

Proof. The first statement follows by direct substitution: if the certificate and rank terms match, the only remaining difference in objective value is the quadratic prior term. CARAT therefore prefers, among utility-equivalent feasible weights, the solution closer to α^{prev}. The second statement is immediate from the projection operator. Hence the optimizer preserves three round invariants throughout: $\alpha_i \ge 0$, $\sum_i \alpha_i = 1$, and $\alpha_i \le 1/((1-\hat{\rho})n)$. □

Proposition A.8 (Pairwise weight-transfer condition)**.** *Let*

$$f(\alpha) = \operatorname*{softmin}_{\beta}(C^\top\alpha) - \lambda_r\gamma^\top\alpha - \lambda_p\|\alpha - \alpha^{\mathrm{prev}}\|_2^2$$

be the CARAT objective, and let e_a, e_b be coordinate vectors for two clients a and b. Suppose $\alpha + \epsilon(e_b - e_a)$ remains feasible for sufficiently small $\epsilon > 0$. Define

$$\pi_t(\alpha) = \frac{\exp(-\beta(C^\top\alpha)_t)}{\sum_{\ell=1}^{T}\exp(-\beta(C^\top\alpha)_\ell)}.$$

Then moving infinitesimal weight from client a to client b increases the objective to first order whenever

$$(C_b - C_a)^\top\pi(\alpha) - \lambda_r(\gamma_b - \gamma_a) - 2\lambda_p[(\alpha_b - \alpha_b^{\mathrm{prev}}) - (\alpha_a - \alpha_a^{\mathrm{prev}})] > 0,$$

where C_i denotes row i of the standardized certificate matrix.

Proof. Let $d = e_b - e_a$. A first-order expansion gives

$$f(\alpha + \epsilon d) - f(\alpha) = \epsilon d^\top\nabla f(\alpha) + o(\epsilon).$$

Using Proposition A.6, the gradient is

$$\nabla f(\alpha) = C\pi(\alpha) - \lambda_r\gamma - 2\lambda_p(\alpha - \alpha^{\mathrm{prev}}).$$

Substituting $d = e_b - e_a$ yields exactly the displayed expression. Hence a client gains weight when its hidden-task utility advantage, measured through the current softmin focus $\pi(\alpha)$, outweighs its rank penalty and temporal-prior cost. □

Proposition A.9 (Per-round complexity)**.** *Let n be the number of clients, T the number of hidden tasks, m the number of sampled coordinates for rank scoring, S the number of rank subsamples, and K the number of projected-gradient steps. Ignoring constant factors from batch size and model width, the extra defense-side work of CARAT scales as*

$$\textit{probe-model forward passes} + O(Snm + KnT).$$

Proof. CARAT performs one baseline probe evaluation plus n candidate probe evaluations over the T hidden tasks, then computes S rank summaries over m coordinates, and finally executes K projected optimization steps over n weights and T task scores. In the prototype implementation, the probe-model forward passes dominate wall-clock time, while the rank and optimization stages are lower-order terms controlled by m, S, and K. □

Remark on the reference assumption. The object $\mathcal{D}^{\mathrm{ref}}$ denotes the labeled server reference set, while $\mathcal{P}$ is a subsampled working pool derived from it for hidden-task construction. This distinction is part of CARAT's threat model and materially strengthens the setting relative to fully data-free robust aggregation.

A.5 Reproducibility, Ethics, and Availability

Reproducibility. All main reported results use the public CIFAR-10 and CIFAR-100 benchmarks and the fixed training protocol described in Section 5. The aggregation method is specified in Algorithm 1, with implementation defaults listed in Appendix B. The main CARAT settings and component variants each use five seeds, and the result archives retain metrics, completion markers, job metadata, and configuration files so that every reported result can be traced to seed-level runs.

Compute resources. The main CARAT sweep and reported public-baseline logs were generated through the repository launch scripts on a Compute Canada SLURM cluster; the component study was run with the same experiment interface on a laboratory GPU server. Each run writes a metadata file recording its configuration, command line, seed, Python environment, and CUDA requirement; the main sweep also records the git commit.

Ethics. This work studies defenses against model-poisoning attacks in federated learning. The attack implementations are used to evaluate robustness and do not require access to private user data. The main practical risk is dual use: stronger attack simulations may help adversaries tune poisoning strategies. We mitigate this by framing attacks as benchmark stressors and by releasing defensive evaluation code together with the corresponding protocols.

Data availability. CIFAR-10 and CIFAR-100 are public datasets. In the reported implementation, CARAT samples a 512-example server pool from the corresponding public training set used to construct client partitions; those examples are not removed from client local data. This overlap and the unequal CARAT/FLTrust reference budgets are limitations of the current evaluation. A stricter reproduction should additionally test deterministic, disjoint server splits while reporting the resulting reduction in client training data.

B Experimental Protocols and Additional Results

This appendix records the exact scope, CARAT configuration, component ablation, client-population scalability study, and runtime diagnostic underlying Section 5.

B.1 Completed Empirical Scope

The completed comparison covers CIFAR-10/VGG19 and CIFAR-100/ResNet18 with 20 fully participating clients, Dirichlet non-i.i.d. partitions, 10% malicious clients, and five attacks: ALIE, FangAttack, MinMax, MinSum, and Mimic. The paper reports an unattacked Mean reference and attacked results for NormClipping, MultiKrum, FLTrust, FLDetector, and CARAT. The 10% comparison selects 150 CARAT runs across the two datasets, three heterogeneity levels, five attacks,

and five seeds; the baseline defenses and clean Mean references also use five seeds. Figure 4 additionally uses the 10%, 20%, 30%, and 40% CARAT runs for the same three heterogeneity levels and five attacks, totaling 600 completed runs. Baseline defenses are not compared across these higher fractions. Appendix B.5 gives a CARAT-only comparison with 20, 40, and 60 clients using the common completed seeds.

B.2 Reported CIFAR Protocol

Table 2: Detailed protocol for the reported CIFAR-10/VGG19 and CIFAR-100/ResNet18 result sets.

Component	**Setting**	**Role or qualification**
Dataset and model	CIFAR-10 / VGG19; CIFAR-100 / ResNet18	Complementary ten-class and class-rich benchmarks.
Client population	20 clients, full participation	Fixed participation in every round.
Training schedule	SGD; 200 rounds; 4 local epochs; learning rate 0.05; momentum 0.9; weight decay 5×10^{-4}; batch size 64	Common optimization schedule for attacked runs.
Data heterogeneity	Dirichlet non-i.i.d. $\alpha \in \{1, 0.5, 0.1\}$	Rows displayed in the main figures.
Malicious clients	10% main comparison; 10%–40% CARAT-only sweep	Two to eight of the 20 clients; each fraction is also supplied to CARAT as $\hat{\rho}$.
Attack suite	ALIE, FangAttack, MinMax, MinSum, Mimic	Four conventional poisoning attacks and a benign-update copying attack.
Compared methods	Clean reference: No Attack + Mean; attacked defenses: Norm-Clipping, MultiKrum, FLTrust, FLDetector, CARAT	Curves in Figures 2 and 3.
Seeds and source policy	Five seeds for every reported result	All attacked defenses and clean references use seeds 42–46.
Server examples	CARAT pool: 512; FLTrust: 100	Budgets are unequal; examples are sampled from the training set and not excluded from client partitions.
Reported statistic	Round-wise accuracy, mean $\pm$ sample standard deviation	The discussion compares round-200 means; rankings exclude the clean reference.

B.3 CARAT Hyperparameters

Table 3 lists the CARAT defaults used in the reported result set. The attacker prior is set to the actual malicious-client fraction; robustness to a misspecified fraction is not evaluated here.

Table 3: CARAT hyperparameters used in the reported CIFAR-10/VGG19 and CIFAR-100/ResNet18 runs.

Component	Default value	Purpose
Reference pool	512 training-set examples	Pool from which hidden tasks are sampled; it is not disjoint from client partitions.
Hidden-task evaluation	8 tasks, 4 examples per class, batch size 128	Builds repeated class-balanced certificates; each task contains 40 examples on CIFAR-10 and 400 on CIFAR-100.
Normalization and clipping	Probe-radius quantile 0.5; MAD clipping factor 2.5	Places candidates on a common evaluation radius and limits residual norm influence.
Feasible weight set	Attacker prior $\hat{\rho} = 0.1$ in the main comparison; 0.1–0.4 in the CARAT-only sweep	Caps per-client weight using the specified experimental attacker fraction.
Task and anomaly weighting	Temperature $\beta = 12.0$; rank weight 0.20; prior weight 0.05	Balances weakest-task utility, rank penalty, and temporal smoothness.
Rank diagnostics	4096 sampled coordinates, 5 rank subsamples	Estimates persistent coordinate-wise extremeness.
Projected optimization	60 projected steps, step size 0.5	Solves the capped-simplex weighting problem.

B.4 Ablation Study of CARAT Components

We conduct a one-component-at-a-time ablation on CIFAR-100/ResNet18 with 20 fully participating clients, 20% malicious clients, non-i.i.d. $\alpha = 0.5$, and 200 communication rounds. This is a separate, higher-adversary experiment rather than a ratio-matched ablation of the main 10% comparison. We report ALIE and FangAttack, each with the same five seeds 42–46 for every variant. Full CARAT uses $T = 8$ hidden tasks, MAD clipping, rank-penalty weight 0.20, and temporal-prior weight 0.05. Each reduced variant changes only one of these settings: one task sets $T = 1$, no clipping disables MAD clipping, no rank penalty sets its weight to zero, and no temporal prior sets its weight to zero. The other recorded settings are held fixed within this study.

Table 4: CARAT component ablation on CIFAR-100/ResNet18 with 20% malicious clients and non-i.i.d. $\alpha = 0.5$ under ALIE and FangAttack. Cells show round-200 test accuracy (%) as mean±sample standard deviation over seeds 42–46; higher is better.

Variant	ALIE	FangAttack
Full CARAT	**46.81±1.06**	**47.59±0.61**
One task ($T = 1$)	46.68±0.93	47.12±0.76
No clipping	46.26±0.62	46.79±0.85
No rank penalty	46.36±1.11	46.96±0.59
No temporal prior	46.33±0.80	47.42±1.18

Full CARAT has the highest mean under both reported attacks: 46.81% for ALIE and 47.59% for FangAttack. Relative to the reduced variants, its gains range from 0.13 to 0.55 percentage points under ALIE and from 0.17 to 0.80 points under FangAttack. The differences are modest compared with the 0.59–1.18-point seed-level standard deviations; in particular, one task nearly matches eight tasks under ALIE. These results support a limited benefit for the complete configuration in this higher-adversary setting, not a claim that every component is independently necessary or that the same ranking holds at 10% malicious clients or on other datasets.

B.5 Client-Population Scalability

We evaluate CARAT with 20, 40, and 60 fully participating clients while fixing the malicious-client fraction at 10%, the training schedule at 200 rounds, and all CARAT hyperparameters. Because the available 60-client archive is incomplete for CIFAR-100 at $\alpha = 1.0$ beyond the first three seeds,

Table 5 uses the common completed seeds 42–44 for every client count and setting. The selected slice contains 324 completed 200-round runs. For each seed, the attacked summary first averages ALIE, FangAttack, MinMax, MinSum, and Mimic; the table then reports the mean and sample standard deviation of these seed-level averages. The no-attack column runs CARAT without malicious updates and is distinct from the NoAttack + Mean reference in the main comparison.

Table 5: CARAT round-200 test accuracy (%) as the fully participating client population increases. All cells use 10% malicious clients when attacked and report mean±sample standard deviation over the common seeds 42–44. "Five-attack mean" averages the five attack accuracies within each seed before aggregation across seeds.

Dataset / Model	α	20 clients		40 clients		60 clients	
		No attack	Five-attack mean	No attack	Five-attack mean	No attack	Five-attack mean
CIFAR-100 / ResNet18	1.0	52.08 ± 0.12	51.07 ± 0.61	55.92 ± 0.35	54.77 ± 0.26	56.89 ± 0.44	44.87 ± 0.35
	0.5	50.95 ± 1.22	49.83 ± 0.79	54.83 ± 0.62	53.80 ± 0.68	55.21 ± 0.27	43.86 ± 0.28
	0.1	44.86 ± 1.45	43.80 ± 0.63	48.47 ± 0.48	35.40 ± 5.31	48.77 ± 0.46	32.39 ± 6.04
CIFAR-10 / VGG19	1.0	79.74 ± 0.22	79.51 ± 0.35	81.93 ± 0.29	81.68 ± 0.17	81.97 ± 0.17	82.23 ± 0.13
	0.5	77.57 ± 0.94	72.35 ± 7.02	79.09 ± 0.91	78.81 ± 0.92	80.15 ± 0.41	80.31 ± 0.56
	0.1	55.85 ± 5.59	48.61 ± 8.33	60.03 ± 3.30	57.77 ± 3.45	62.16 ± 1.97	55.29 ± 8.16

The no-attack results improve or remain stable as the client population grows, so increasing the number of clients does not by itself prevent optimization. Under attack, CIFAR-10 also improves from 20 to 40 clients in all three heterogeneity regimes and remains strong at 60 clients for $\alpha \in \{1.0, 0.5\}$. The $\alpha = 0.1$ five-attack mean decreases from 57.77% at 40 clients to 55.29% at 60, with high variability caused by two chance-level MinSum seeds.

CIFAR-100 exposes a different scaling failure. FangAttack finishes at chance-level 1% accuracy for all three common seeds at 60 clients and every reported α, which lowers the five-attack averages despite improving no-attack accuracy. At $\alpha = 0.1$, MinSum additionally collapses for one of three seeds at 40 clients and two of three seeds at 60 clients. Thus, these results show that CARAT's benign training behavior scales to 60 clients, but its robustness does not scale uniformly across attacks. They are a CARAT-only diagnostic based on three common seeds, not a larger-federation baseline comparison or a complete five-seed result.

B.6 Runtime Diagnostic

Figure 5 reports a separate, earlier CIFAR-100/i.i.d. ALIE verification run with three seeds. CARAT records an average per-round time of 9.94 seconds, compared with 2.80 seconds for Mean under attack, or approximately 3.6× higher wall-clock cost. This premium is consistent with evaluating every candidate update on repeated hidden tasks. The diagnostic predates the five-seed sweep and is not hardware-normalized for the main comparison.

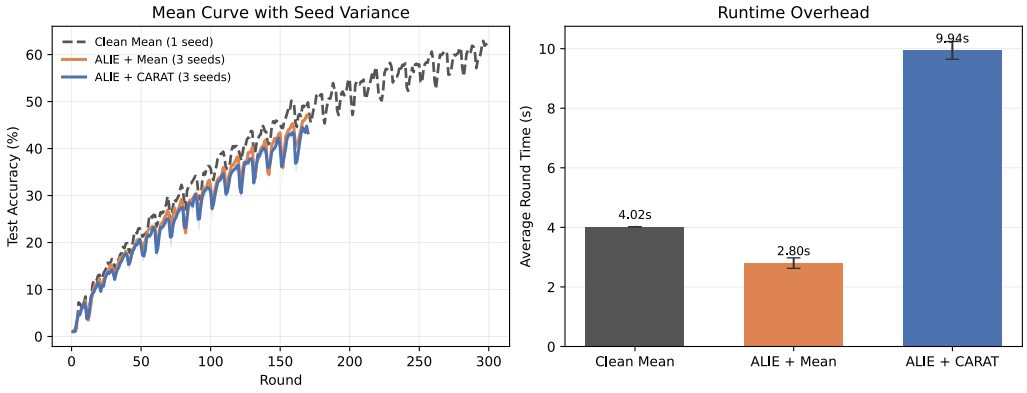


Figure 5: Verification-scale CIFAR-100/i.i.d. results under ALIE. Left: round-wise test accuracy over three seeds; the dashed clean Mean curve is a single-seed no-attack reference. Right: average per-round wall-clock time with seed-level error bars. This earlier run is an overhead diagnostic and is not part of the main five-seed comparison.