RESCENE: SERVER-SIDE REPLAY FOR STRUCTURAL MITIGATION OF CATASTROPHIC FORGETTING IN FEDERATED CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated continual learning must integrate new tasks over time without losing earlier-task knowledge. Most existing methods attach an anti-forgetting mechanism to the client-trained, server-aggregated loop of federated learning, which holds back new learning to preserve earlier knowledge and burdens resource-constrained clients. We propose ReSCENE, which structurally mitigates catastrophic forgetting by having each client upload a small condensed surrogate of its local data while the server keeps the surrogates of past tasks and trains the global model on them together with the current task surrogates. For efficient server memory, we introduce *temporal herding*, which selects the more recent surrogates from the pool accumulated over a task into a compressed buffer. Our study provides a theoretical analysis showing that this buffer can represent the original task data more closely than full accumulation of all surrogates. Across CIFAR-10, CIFAR-100, and TinyImageNet, ReSCENE achieves the strongest accuracy over seven baselines, by up to 31.1 points of average accuracy, while requiring as little as $0.11\times$ of the client computation and up to $179\times$ less upload than the model-update baselines. ReSCENE further demonstrates its effectiveness when scaled to larger client populations and larger models while remaining efficient, which makes it a practical method for federated continual learning.

1 INTRODUCTION

Federated learning (FL) has emerged as a privacy-preserving paradigm that trains a shared model across data owners without centralizing their data (McMahan et al., 2017). Standard FL assumes that each client's local data is static, yet in practice new classes keep arriving and the data distribution shifts over time. Thus, the shared model must keep learning from new data while retaining previously acquired knowledge. This dynamic setting within a federated environment is *federated continual learning* (FCL) (Yoon et al., 2021). Each time the model trains on a new task, earlier knowledge is progressively overwritten as its parameters shift toward the new task, so the model fails on the tasks it learned before. This loss of earlier-task ability is *catastrophic forgetting* (McCloskey & Cohen, 1989; Bakman et al., 2024), and mitigating it is the core problem of FCL.

Many FCL methods counter forgetting by adding a continual-learning mechanism to FL. *Regularization-based* methods restrain updates to the parameters important to earlier tasks (Kirkpatrick et al., 2017; Li et al., 2025), and *distillation-based* methods have the current model imitate a previous-task teacher (Li & Hoiem, 2018; Ma et al., 2022). *Replay-based* methods rehearse past data, by storing exemplars on the clients (Dong et al., 2022; Li et al., 2024) or by synthesizing it with a generator (Zhang et al., 2023; Chen et al., 2024). These mechanisms preserve earlier-task knowledge, but in exchange they either hold back new learning or load resource-constrained clients with extra memory and computation. Moreover, they only slow the erasure, so as the task sequence grows (Li et al., 2024), the capability on earlier tasks keeps degrading and performance drops.

Instead of adding the burden to resource-constrained clients, we shift the anti-forgetting responsibility to the resource-rich server (Kairouz et al., 2021). Each client condenses its local data into a small set of surrogates and uploads it, as in surrogate-aggregation FL (SA-FL) (Xiong et al., 2023; Wang et al., 2024b). The server keeps these earlier-task surrogates at no additional communication

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.

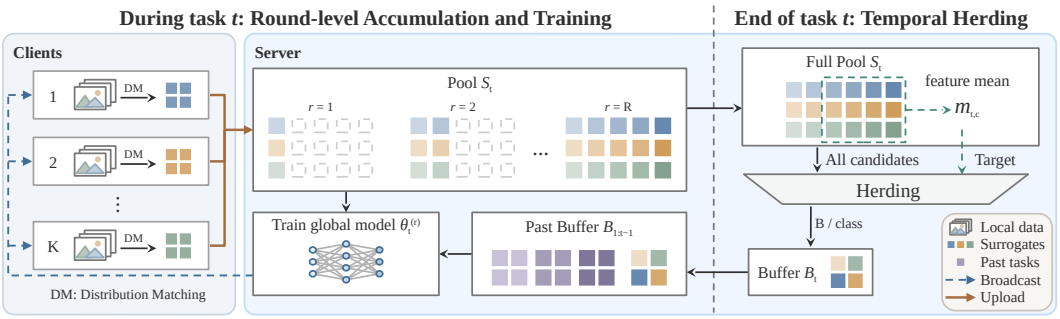


Figure 1: Illustration of ReSCENE. During task t, clients upload condensed surrogates every round, and the server trains the global model on the accumulated pool together with the buffers of earlier tasks. At the end of the task, temporal herding compresses the pool into a fixed-size buffer centered on the recent rounds, which joins the past buffer. Darker tiles denote more recent rounds.

cost, and trains the global model on them together with those of the current task when a new task arrives. Therefore, old and new tasks are optimized at once, which structurally mitigates catastrophic forgetting (Figure 1). Whereas prior SA-FL trains on each surrogate once and discards it, we turn the accumulated ones into a replay source for continual learning. We call this method ReSCENE (*re* + *scene*), since the server re-sees the scene of previous tasks while it learns the current one.

Keeping every surrogate, however, would take up too much server memory, and not all of them are equally worth keeping. Within a task, the global model and the surrogates co-evolve, since each round's surrogates train the model used for condensation in the next round. Surrogates from later rounds thus represent the task more faithfully. We therefore introduce *temporal herding*, which compresses each task into a small buffer centered on the recent rounds, selecting from its whole pool by herding (Welling, 2009). Our theoretical analysis shows that this buffer can represent the original task data more closely than keeping the whole pool, and we empirically confirm this advantage on longer task sequences while using less server memory and computation.

We evaluate ReSCENE on CIFAR-10, CIFAR-100, and TinyImageNet at three heterogeneity levels, following the federated class-incremental learning (FCIL) (Dong et al., 2022) protocol, in which each task introduces new classes and the model is evaluated on all seen classes. ReSCENE demonstrates the strongest performance among seven FCL baselines, reaching 67.96% AA / 73.53% AIA on CIFAR-10 and 49.80% / 57.71% on CIFAR-100, exceeding the strongest baseline by 17.2 and 8.2 AA points. It is also far more efficient on the client side, requiring as little as $0.11\times$ of the client computation of FedAvg and an upload up to $179\times$ smaller than a model update. We further show that ReSCENE remains effective with larger client populations and larger models while keeping this efficiency. Overall, ReSCENE provides a practical and effective solution for FCL, balancing strong performance with significantly lower client-side computation and communication. We summarize our main contributions as follows:

- We propose ReSCENE, which keeps the surrogates of earlier tasks on the server and trains the global model on them together with those of the current task, structurally mitigating forgetting.
- We introduce temporal herding, which compresses the retained surrogates into a bounded buffer centered on the recent generations, and prove that this buffer can represent the original task data more closely than full accumulation.
- We demonstrate the effectiveness and practicality of ReSCENE on three FCIL benchmarks, where it outperforms seven baselines at far lower client cost and keeps its advantage with larger client populations and larger models.

2 Related Work

Anti-forgetting in FCL. FCL methods can be categorized into regularization-, distillation-, and replay-based families (Hamedi et al., 2025). Regularization-based methods restrict changes to the parameters that contributed most to earlier tasks (Kirkpatrick et al., 2017; Li et al., 2025), which leaves little room for new knowledge as tasks continue. Distillation-based methods use the model trained on earlier tasks as a teacher and make the current model match its predictions on old tasks (Li

& Hoiem, 2018; Ma et al., 2022). Since the teacher is stale, the model is pulled back to earlier states, which hinders the continual learning process. Replay-based methods rehearse earlier-task data, by storing exemplars on the clients (Dong et al., 2022; Li et al., 2024) or by synthesizing them with a generator (Zhang et al., 2023; Chen et al., 2024). Exemplar methods retain earlier knowledge well, since the client trains directly on raw data of earlier tasks, but they add computation and consume scarce memory of resource-poor clients. Generative methods store no raw data and can produce as many past-class samples as training needs, but the generated samples often drift from the real distribution in the heterogeneous federated setting and become a poor substitute for the real data. We discuss each family in detail in Appendix B.

Surrogate-aggregation FL. Surrogate-aggregation FL (SA-FL) has been introduced as an alternative to traditional model-transmitting FL. FedDM (Xiong et al., 2023) has each client condense its local data into a small set of surrogates and upload them, and the server trains the global model on the pooled set. FedAF (Wang et al., 2024b) follows the same structure and lets each client synthesize surrogates that reflect the global data distribution, which reduces client drift and tackles data heterogeneity. Both study static training, where the local data for each client is fixed. The server thus trains on each surrogate once and then discards it, so the information that clients have already sent is used only once. ReSCENE instead keeps them at the server without any further communication cost and replays them as later tasks arrive.

3 METHOD

We propose ReSCENE, which replays the surrogates of earlier tasks at the server to structurally mitigate catastrophic forgetting. In every round, the participating clients condense their local data into surrogates with the frozen global model and upload them, and the server trains the global model on the surrogates of the current task together with the buffers of earlier tasks (Sec. 3.2) under a class-balanced objective (Sec. 3.4). At the end of each task, the server compresses the surrogates of the task into a buffer by temporal herding (Sec. 3.3). We provide a theoretical analysis of how closely this buffer represents the task in Sec. 3.5. Figure 1 and Algorithm 1 summarize the full procedure.

3.1 PROBLEM SETUP AND NOTATION

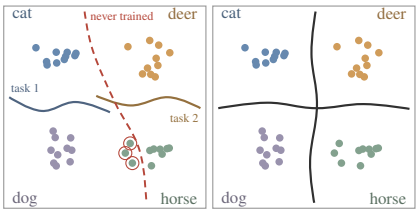
We consider federated class-incremental learning with N clients and T tasks, each of R rounds. Task t introduces a disjoint set of new classes $\mathcal{C}_t$, from which client k draws its local data $\mathcal{D}_t^k$. In each round r of task t, a subset $\mathcal{K}_{t,r}$ of K clients participates. We denote the global model by f_θ and its feature extractor by ϕ_θ, with parameters $\theta_t^{(r)}$ after round r of task t. On the server, S_t is the pool of surrogates collected during task t, $\mathcal{B}_t$ is the replay buffer kept from task t, and $\mathcal{B}_{1:t} = \bigcup_{s \le t} \mathcal{B}_s$ denotes the buffers of tasks 1 through t.

3.2 SURROGATE AGGREGATION FOR FEDERATED CONTINUAL LEARNING

Surrogate aggregation. Unlike conventional model-aggregation FL, clients in ReSCENE communicate condensed surrogates of their local data (Xiong et al., 2023; Wang et al., 2024b). In each round r of task t, client k condenses its local data by distribution matching (Zhao & Bilen, 2023). For each class c, it optimizes a small synthetic set $S_{t,c}^{k,(r)}$ so that its mean embedding and mean logits match those of the client's real data of that class, $\mathcal{D}_{t,c}^k$, using the frozen global model $\theta_t^{(r-1)}$,

$$\min_{S_{t,c}^{k,(r)}} \mathbb{E}_{\theta'} \left[\left\| \overline{\phi_{\theta'}}(S_{t,c}^{k,(r)}) - \overline{\phi_{\theta'}}(\mathcal{D}_{t,c}^k) \right\|^2 + \left\| \overline{f_{\theta'}}(S_{t,c}^{k,(r)}) - \overline{f_{\theta'}}(\mathcal{D}_{t,c}^k) \right\|^2 \right], \tag{1}$$

where $\overline{(\cdot)}$ is the average over the set. This objective trains the synthetic set so that its loss matches the client's local loss. Since matching the two over the whole parameter space is infeasible, we take the expectation over θ', which is $\theta_t^{(r-1)}$ perturbed by isotropic Gaussian noise within a ball of radius ρ (Xiong et al., 2023). The surrogates thus remain a faithful substitute for the real data within the ball. Each surrogate set $S_{t,c}^{k,(r)}$ contains a fixed number of images per class (IPC), regardless of how much local data the class has or how large the model is. The client then uploads the collected surrogates $S_t^{k,(r)} = \bigcup_c S_{t,c}^{k,(r)}$ to the server. In Appendix F, we analyze the differential privacy of these synthetic sets and the accuracy under gradient perturbation.



(a) Task-split optimization (b) ReSCENE

Figure 2: Schematic of the example in Sec. 3.2. Solid curves denote decision boundaries, and the red dashed curve is the cross-task boundary that task-split optimization never trains.

Algorithm 1: ReSCENE: server-side surrogate replay with temporal herding

Input: N clients, K per round, T tasks, R rounds per task, images per class IPC, budget B, temporal fraction λ, sampling exponent α, logit strength τ

initialize $\theta_1^{(0)}$
for $t = 1$ **to** T **do**
  expand the classifier head by $|\mathcal{C}_t|$ outputs; $S_t \leftarrow \emptyset$
  for $r = 1$ **to** R **do**
    sample participants $\mathcal{K}_{t,r}$ and broadcast $\theta_t^{(r-1)}$
    for each *client* $k \in \mathcal{K}_{t,r}$ in parallel **do**
      upload $S_t^{k,(r)} \leftarrow \textsc{Condense}(\mathcal{D}_t^k; \theta_t^{(r-1)})$ ▷ Eq. 1
    end for
    $S_t \leftarrow S_t \cup \bigcup_{k \in \mathcal{K}_{t,r}} S_t^{k,(r)}$
    $\theta_t^{(r)} \leftarrow \textsc{ServerTrain}(\theta_t^{(r-1)}, \mathcal{B}_{1:t-1} \cup S_t; \alpha, \tau)$
  end for
  for each *class* $c \in \mathcal{C}_t$ **do**
    $\mathcal{B}_{t,c} \leftarrow \textsc{TemporalHerding}(S_{t,c}, \lambda, B; \theta_t^{(R)})$ ▷ Eq. 2
  end for
  $\mathcal{B}_t \leftarrow \bigcup_{c \in \mathcal{C}_t} \mathcal{B}_{t,c}$; $\theta_{t+1}^{(0)} \leftarrow \theta_t^{(R)}$
end for

Structural mitigation of forgetting. ReSCENE makes full use of the surrogates that SA-FL trains on once and discards, without any further communication cost. The uploaded surrogates join the pool S_t of the current task, on which the server trains the global model in every round. At the end of task t, the server keeps the task's pool as a compressed buffer $\mathcal{B}_t$, which Sec. 3.3 describes. The server thus retains access to previous tasks and trains the global model on $\mathcal{B}_{1:t-1} \cup S_t$, so the earlier tasks are optimized together with the current task, which structurally mitigates catastrophic forgetting.

One objective over all seen classes. The goal of FCL is to minimize the risk over all classes of tasks 1 through t. Existing methods split this problem into a sequence of subproblems, each the risk over the classes of one task, and merge the knowledge of the separate subproblems into a single model. The merged solution does not add up to the solution of the original problem, since the classes of different tasks are never optimized against each other. ReSCENE optimizes the original problem itself, since the server's training set covers every seen class in one objective. Consider task 1 with classes $\{\text{cat}, \text{dog}\}$ and task 2 with classes $\{\text{deer}, \text{horse}\}$. Under task-split optimization, prior methods learn to separate cat from dog and deer from horse, yet what is asked after task 2 is the four-way classification, including the boundaries between cat and deer or between dog and horse, which have never been learned (Figure 2a). In ReSCENE, by contrast, the server trains on the pooled surrogates of both tasks, so it learns to tell cat, dog, deer, and horse apart at once and acquires a discrimination ability that spans all seen classes (Figure 2b).

3.3 Temporal Herding

Co-evolution of the model and the surrogates. Within a task, the global model and the surrogates co-evolve. In each round, the clients condense their local data using the model trained on the surrogates of the previous rounds, and the server then trains that model further on the new surrogates. Since the global model improves over the rounds, the clients of later rounds optimize their surrogates under a better model, so the surrogates of later rounds represent the local data more faithfully than those of earlier rounds.

Temporal herding. Keeping all surrogates at the server would let its memory grow with every round and every task. We therefore introduce temporal herding, which compresses the pool of a finished task into a bounded buffer by herding (Welling, 2009). Following the co-evolution observation, we take the recent surrogates as the target and select the candidates from the whole pool.

Let $S_{t,c}$ be the class-c part of the pool S_t, $\lambda \in (0, 1]$ the fraction of the R rounds that forms the recent window, and $W_{t,c} \subseteq S_{t,c}$ the class-c surrogates uploaded in the last $\lceil \lambda R \rceil$ rounds of the task. Under the task-end model $\theta_t^{(R)}$ we normalize each surrogate feature, $\boldsymbol{z}(s) = \phi_{\theta_t^{(R)}}(s) / \|\phi_{\theta_t^{(R)}}(s)\|$, and take the mean over the recent window, $\boldsymbol{m}_{t,c} = |W_{t,c}|^{-1} \sum_{s \in W_{t,c}} \boldsymbol{z}(s)$, as the target. Herding then picks B surrogates $p_1, \ldots, p_B$ from $S_{t,c}$, where p_j is chosen among the remaining candidates

so that the mean of $p_1, \ldots, p_j$ is closest to the target,

$$p_j \in \underset{s \in S_{t,c} \setminus \{p_1, \ldots, p_{j-1}\}}{\arg\min} \left\| \boldsymbol{m}_{t,c} - \frac{1}{j} \left(\boldsymbol{z}(s) + \sum\nolimits_{i<j} \boldsymbol{z}(p_i) \right) \right\|^2, \tag{2}$$

and selected B surrogates form $\mathcal{B}_{t,c}$. Collecting $\mathcal{B}_{t,c}$ over the classes of the task gives the final buffer of the task, $\mathcal{B}_t = \bigcup_{c \in \mathcal{C}_t} \mathcal{B}_{t,c}$. We provide a theoretical analysis of when this buffer represents the task more closely than full accumulation in Sec. 3.5, and ablate λ and B in Appendix D.4.

3.4 Resolving Class Imbalance in Server-Side Training

Each previous class is capped at B herded surrogates, whereas the current task receives fresh surrogates every round, so the server's training pool $\mathcal{B}_{1:t-1} \cup S_t$ becomes skewed toward new classes with every round. To resolve this imbalance, we sample classes with probability $q_c = n_c^\alpha / \sum_{c'} n_{c'}^\alpha$, where n_c is the class count in this pool and $\alpha \in [0, 1]$ spans count-proportional sampling at $\alpha = 1$ and class-balanced sampling at $\alpha = 0$. In addition, we train with logit adjustment (Menon et al., 2021) under the surrogate-pool prior $\pi_c = n_c / \sum_{c'} n_{c'}$, minimizing $\mathcal{L}_{\text{server}} = \text{CE}(f_\theta(x) + \tau \log \boldsymbol{\pi}, y)$ with $c \sim q$. We ablate both components in Appendix D.5.

3.5 Theoretical Analysis of Temporal Herding

Full accumulation, which keeps the whole pool of a task without any selection, is a natural choice for replay, since it discards none of the surrogates. We establish the condition under which the buffer formed by temporal herding represents the class more closely than full accumulation, even though it keeps only B surrogates per class. We provide the full proofs in Appendix C.

Setup. We fix a class and write P for the indices of all its surrogates in the task pool. $P_r \subseteq P$ holds those uploaded in round r and tracks how their error changes over rounds, $W \subseteq P$ holds those from the recent $\lceil \lambda R \rceil$ rounds, and $H_\ell \subseteq P$ holds those selected in the first ℓ steps of Eq. (2), so that H_B indexes the buffer $\mathcal{B}_{t,c}$. Each surrogate is represented by its normalized task-end feature $\boldsymbol{z}_i$, and $\boldsymbol{m}_A$ denotes the mean of $\boldsymbol{z}_i$ over an index set A. Let $\boldsymbol{\mu}(\theta)$ be the real class mean under model θ, and write $\boldsymbol{\mu} = \boldsymbol{\mu}(\theta_t^{(R)})$ for the mean of the real class under the task-end model, the quantity that $\mathcal{B}_{t,c}$ should represent faithfully. Our analysis is based on the following assumptions.

Assumption 1. (Directional coverage) *For each $2 \le \ell \le B$, some $q \in P \setminus H_{\ell-1}$ satisfies*

$$\langle \boldsymbol{m}_W - \boldsymbol{m}_{H_{\ell-1}}, \boldsymbol{z}_q - \boldsymbol{m}_W \rangle \ge 0.$$

Assumption 1 ensures that at every step some candidate not yet selected moves the mean of the selected set toward the target, since it lies at or beyond $\boldsymbol{m}_W$ along the direction from the current mean to the target.

Assumption 2. (Coherent stale bias) *There exist a unit vector $\boldsymbol{u}$, coefficients $a_r \ge 0$ non-increasing in r, and vectors $\boldsymbol{\xi}_r$ with $\|\boldsymbol{\xi}_r\| \le \sigma$ such that $\boldsymbol{m}_{P_r} - \boldsymbol{\mu} = a_r \boldsymbol{u} + \boldsymbol{\xi}_r$ for every round r.*

Assumption 2 decomposes the task-end error of the round-r surrogates into a stale bias along a fixed direction and a bounded deviation. The bias does not grow over rounds, since later surrogates are matched under models closer to the task-end model and are thus less stale (Proposition 1).

Let Δ_W be the largest distance from a candidate to the target, which is at most 2 since the features are unit vectors. Let A_P and A_W be the weighted averages of a_r over the whole pool and over the recent window, each round weighted by its number of surrogates. These quantities represent the stale biases carried by $\boldsymbol{m}_P$ and $\boldsymbol{m}_W$, and $A_W \le A_P$ since W holds the latest surrogates.

Lemma 1. *Under Assumption 1, for $B \le |P|$, the buffer $\mathcal{B}_{t,c}$ satisfies $\|\boldsymbol{m}_{H_B} - \boldsymbol{m}_W\| \le \Delta_W / \sqrt{B}$.*

Each surrogate picked by Eq. (2) keeps the mean of the selected set close to the target, so the finite buffer $\mathcal{B}_{t,c}$ preserves the recent target up to an approximation error of at most $2/\sqrt{B}$.

Dataset	β	Metric	FedAvg	FedEWC	FedLwF	TARGET	GLFC	FedCBDR	Re-Fed	ReSCENE
CIFAR-10	0.1	AA	10.00±0.00	10.00±0.00	10.41±0.61	9.95±0.34	14.91±5.36	25.94±3.63	30.67±3.72	**61.78**±1.14
		AIA	23.34±0.88	23.37±0.92	22.93±0.09	22.80±0.04	24.65±2.41	37.23±3.36	36.74±5.41	**68.35**±0.62
	0.5	AA	15.84±3.04	17.54±0.94	16.94±5.70	21.66±7.12	35.38±1.60	46.78±0.93	50.73±3.02	**67.96**±1.33
		AIA	32.90±7.78	36.81±5.03	33.57±11.77	36.76±13.40	48.90±6.83	57.64±4.59	62.69±6.83	**73.53**±0.73
	1.0	AA	14.70±4.25	15.12±4.53	22.82±7.36	18.86±1.39	43.13±0.96	50.06±0.91	56.57±1.26	**69.64**±0.84
		AIA	30.83±1.89	31.78±4.52	33.51±5.96	32.37±3.58	52.56±5.15	57.07±2.63	66.83±2.33	**74.57**±0.67
CIFAR-100	0.1	AA	5.80±0.91	5.84±0.45	9.01±2.41	8.48±4.04	17.84±0.69	34.21±1.42	29.58±0.85	**44.89**±1.13
		AIA	9.93±0.83	10.05±0.84	11.84±3.56	10.53±2.48	26.18±1.24	37.97±0.75	34.98±0.77	**49.67**±0.69
	0.5	AA	8.14±0.11	8.14±0.05	20.65±0.81	20.18±3.87	20.23±0.15	41.64±1.31	33.92±0.47	**49.80**±0.22
		AIA	18.46±0.22	18.83±0.79	28.62±1.24	30.95±2.11	31.67±0.52	49.83±0.45	49.89±0.10	**57.71**±0.34
	1.0	AA	8.24±0.05	8.27±0.21	23.60±0.45	24.90±1.34	20.86±0.14	41.83±0.37	34.65±0.47	**50.70**±0.17
		AIA	20.44±0.92	20.78±0.81	34.54±2.78	35.13±2.86	32.85±0.29	52.03±1.55	51.94±0.56	**58.85**±0.19
TinyImageNet	0.1	AA	4.81±0.19	4.80±0.18	10.97±1.20	13.64±0.13	8.32±0.70	21.10±1.23	14.98±1.20	**23.84**±0.18
		AIA	11.38±0.15	11.26±0.63	16.66±1.01	18.86±0.87	16.45±0.57	27.27±1.49	26.75±0.90	**31.80**±0.68
	0.5	AA	6.04±0.16	6.08±0.01	19.20±0.25	21.50±0.04	10.64±0.46	25.80±0.82	21.78±0.23	**26.09**±1.01
		AIA	17.21±0.29	17.23±0.28	29.06±1.84	30.78±0.50	21.10±0.29	34.45±1.87	**37.96**±0.64	36.07±0.86
	1.0	AA	6.21±0.02	6.23±0.06	19.23±0.59	20.66±3.22	10.82±0.68	26.10±1.17	22.91±0.40	**28.69**±0.43
		AIA	17.39±0.29	17.41±0.17	29.26±1.15	30.56±4.35	22.22±0.28	35.35±3.59	**39.40**±0.31	37.86±0.13

Table 1: AA and AIA (%) of the baselines and ReSCENE. Small gray values are standard deviations over three seeds.

Theorem 1. *Under Assumptions 1 and 2, the buffer $\mathcal{B}_{t,c}$ is strictly closer to the real class mean than full accumulation under the condition*

$$\underbrace{A_P - A_W}_{\text{removed stale bias}} > \underbrace{2\sigma}_{\text{deviation}} + \underbrace{\Delta_W/\sqrt{B}}_{\text{approximation error}} \implies \|\boldsymbol{m}_{H_B} - \boldsymbol{\mu}\| < \|\boldsymbol{m}_P - \boldsymbol{\mu}\|.$$

Theorem 1 shows that, under its condition, the buffer of temporal herding represents the real class mean more closely than full accumulation with only B surrogates per class. The condition requires the removed stale bias to exceed the deviation and the approximation error, and it is increasingly met over rounds, since the earlier surrogates grow more stale while 2σ and $\Delta_W/\sqrt{B}$ stay fixed.

4 Experiments

4.1 Experimental Setup

Datasets and Protocol. We evaluate on three federated class-incremental benchmarks. CIFAR-10 is divided into T=5 two-class tasks, and CIFAR-100 and TinyImageNet into T=10 tasks of ten and twenty classes, respectively. We use N=20 clients, K=10 participants per round, R=20 communication rounds per task, and task-wise Dirichlet label skew $\beta \in \{0.1, 0.5, 1.0\}$. Every configuration is run with three seeds, and we report the mean and standard deviation.

Baselines and Implementation. All methods use ResNet-18 with a classifier that expands as new tasks arrive. We compare with FedAvg (McMahan et al., 2017), with FedEWC (Kirkpatrick et al., 2017) and FedLwF (Li & Hoiem, 2018) as representatives of the regularization and distillation families, and with TARGET (Zhang et al., 2023), GLFC (Dong et al., 2022), FedCBDR (Qi et al., 2025), and Re-Fed (Li et al., 2024) as replay-based methods of the same family as ReSCENE. We further compare with the SA-FL methods FedDM (Xiong et al., 2023) and FedAF (Wang et al., 2024b) under the same condensation settings as ReSCENE. Every baseline trains its local model for 20 epochs per round. ReSCENE uses IPC of 10, a per-class buffer budget B=1000, a temporal fraction λ=0.75, and E_{srv}=100 server epochs per round. Appendix A describes the datasets, settings, and per-method hyperparameters in full detail.

Metrics. We report Average Accuracy (AA), the accuracy over all seen classes after the final task, and Average Incremental Accuracy (AIA), which measures that same accuracy after every task and

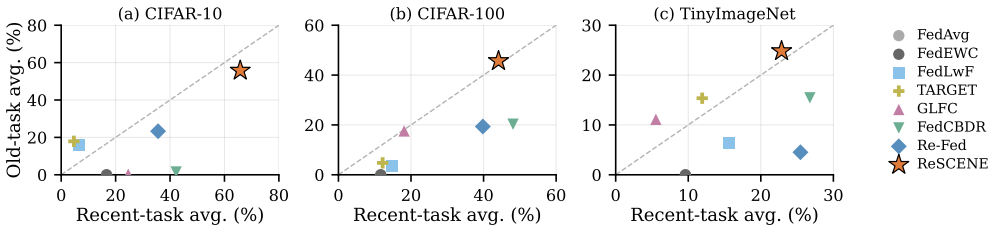


Figure 3: Old-task versus recent-task accuracy at β=0.1. Points high on both axes and near the diagonal forget less, performing well on both old and recent tasks.

averages it over the T tasks. AA captures what the model retains at the end of the sequence, and AIA captures how well it performed throughout it.

4.2 Main Results

Table 1 reports AA and AIA on the three benchmarks. ReSCENE attains the highest AA in all nine settings and the highest AIA in seven of them, and its margin over the strongest baseline is largest at β=0.1 on every benchmark. On CIFAR-10 at β=0.1 it leads Re-Fed by 31.1 AA points, and on CIFAR-100 it leads FedCBDR by 8 to 11 AA points at every β. On TinyImageNet, where every method reaches lower accuracy, ReSCENE still performs best in all settings except the AIA at β=0.5 and 1.0. We attribute this advantage to the server training on the earlier tasks together with the current one at every round, which lets ReSCENE learn the boundaries between classes of different tasks directly. The advantage over the replay baselines also holds when every replay method receives the same replay budget as ReSCENE, where ReSCENE stays 17.2 to 48.3 AA points ahead (Appendix D.3). We also evaluate ReSCENE in the domain-incremental setting on Office-31, where it best retains the earlier domains and attains the highest AA and AIA (Appendix D.2).

Comparison with SA-FL baselines. We compare ReSCENE with the existing SA-FL methods in the FCL setting. Both methods follow the same condense-and-upload structure as ReSCENE, but the server trains only on the surrogates received in the current round and discards them afterwards. Table 2 shows that both methods fail to retain any earlier task. On all three benchmarks, their final accuracy on every earlier task drops to 0% and only the last task keeps its accuracy, so their AA stays at 16% or below (Appendix D.1). Without earlier-task surrogates at the server, the global model has no data of the earlier classes to train on, and each new task overwrites them entirely. ReSCENE keeps them and trains on them together with the current task, and remains effective on every benchmark, which shows that surrogate aggregation mitigates forgetting only once the surrogates are kept and replayed.

Dataset		FedDM	FedAF	ReSCENE
CIFAR-10	AA	15.69	15.99	**61.78**
	AIA	37.06	37.06	**68.35**
CIFAR-100	AA	6.27	6.37	**44.89**
	AIA	16.31	16.35	**49.67**
TinyImageNet	AA	2.52	2.58	**23.84**
	AIA	10.03	10.37	**31.80**

Table 2: AA and AIA (%) of the SA-FL baselines and ReSCENE at β=0.1.

4.3 Forgetting Analysis

Catastrophic forgetting is a central challenge in FCL, as knowledge acquired on earlier tasks is erased when learning new tasks, so mitigating it requires preserving task-level accuracy throughout the sequence. We split the task sequence into an old and a recent group, measure the accuracy of each group after the entire sequence, and plot them against each other in Figure 3. For CIFAR-10 the old group is T0 and T1 and the recent group T2 to T4, and for the ten-task datasets each group holds five tasks. Appendix D.1 reports the per-task accuracy behind every entry of Table 1 and plots the full task-wise profiles.

Most baselines lie below the diagonal, so their final accuracy relies on the recent tasks while the old tasks are largely forgotten. ReSCENE attains the highest old-task accuracy on every benchmark, and lies close to the diagonal, showing comparable accuracy on old and recent tasks. This balance results from the server keeping the surrogates of every earlier task and training on them together with the current one, which allows ReSCENE to retain earlier-task knowledge throughout the sequence and thus structurally mitigate forgetting.

Method	Client comp. (PFLOP/run)	Uplink (MB/rd)	Accuracy AA	Accuracy AIA
FedAvg	46.33	44.73	14.58 ↓1.26	28.49 ↓4.41
FedEWC	48.65	44.73	15.97 ↓1.57	31.62 ↓5.19
FedLwF	58.69	44.73	14.38 ↓2.56	29.30 ↓4.27
TARGET	132.25	44.73	16.01 ↓5.65	31.29 ↓5.47
GLFC	108.45	44.73	34.52 ↓0.86	49.21 ↑0.31
FedCBDR	326.31	44.73	40.07 ↓6.71	52.12 ↓5.52
Re-Fed	139.87	44.73	47.58 ↓3.15	56.55 ↓6.14
ReSCENE	21.93	0.25	**72.13** ↑4.17	**78.46** ↑4.93

(a) More clients (CIFAR-10, β=0.5)

Backbone	Method	Client comp. (PFLOP/run)	Uplink (MB/rd)	Accuracy AA	Accuracy AIA
ResNet-18	FedCBDR	933.15	44.92	41.64	49.83
	Re-Fed	146.46	44.92	33.92	49.89
	ReSCENE	27.49	1.23	**49.80**	**57.71**
ResNet-50	FedCBDR	2180.66	95.03	36.44	42.05
	Re-Fed	342.34	95.03	29.06	38.54
	ReSCENE	64.26	1.23	**55.25**	**61.70**
ViT-Small/4	FedCBDR	2320.76	85.51	20.25	29.20
	Re-Fed	364.34	85.51	22.66	33.27
	ReSCENE	68.40	1.23	**31.45**	**38.59**

(b) Larger models (CIFAR-100, β=0.5)

Table 3: Accuracy and client cost with (a) more clients and (b) larger models. In (a), the arrows and the numbers beside them give the change from N=20 in Table 1.

5 Ablations and Analyses

5.1 Scalability to Larger Client Populations and Models

Practical federated deployments typically involve many clients and benefit from larger models. We therefore test whether the advantage of ReSCENE holds as the client population and the model grow.

Larger client populations. Cross-device federated learning involves a large population of clients, each holding a small share of the data (Kairouz et al., 2021). To move closer to this regime, we scale the population to N=100 clients with K=70 participants per round on CIFAR-10 at β=0.5. Every baseline loses AA as the population grows, whereas ReSCENE gains 4.17 AA and 4.93 AIA points over the N=20 setting and reaches 72.13 AA (Table 3a). With more clients, each client holds less and more skewed data, so each local update fits a narrower distribution, and averaging many such updates degrades the global model. In ReSCENE, each client condenses its own data, and the server trains on the surrogates of all 70 participants together, so a larger population supplies surrogates from more partitions, while temporal herding keeps the buffer at B=1000 per class. ReSCENE keeps the lowest client cost, 0.47× FedAvg's computation and a 0.25 MB uplink against 44.73 MB.

Larger models. Traditional weight-communicating methods train and upload the full model on every client, so their client cost grows with the model size. ReSCENE, in contrast, trains the model at the server, and its clients upload a fixed number of surrogates per class, so its uplink is independent of the model size. We compare ReSCENE with FedCBDR and Re-Fed, the strongest baselines in Table 1, on ResNet-50 and ViT-Small/4 for CIFAR-100 at β=0.5, with ResNet-18 as the reference.

ReSCENE attains the highest accuracy at every model size, leading the strongest baseline by 8.16, 18.81, and 8.79 AA points (Table 3b). From ResNet-18 to ResNet-50, FedCBDR and Re-Fed lose 5.20 and 4.86 AA points, whereas ReSCENE gains 5.45. The model grows while the local data of each client stays small and skewed, so FedCBDR and Re-Fed must fit more parameters from the same few samples on every client. Meanwhile, ReSCENE trains the global model on the collected surrogate pool, which gathers the surrogates of all clients, so the larger model is trained on more data and benefits from its added capacity. ReSCENE uploads the same 1.23 MB for every backbone, and its client computation stays at about 2.9% of FedCBDR's and 18.8% of Re-Fed's, since its clients only synthesize surrogates through a frozen model. All methods drop on ViT-Small/4 due to its lack of a convolutional inductive bias, yet ReSCENE still leads. Appendix E accounts for the full client and server cost, and Sec. 5.3 shows that ReSCENE stays ahead with far less server training.

5.2 Herding-Buffer Ablation

Following the analysis in Sec. 3.5, we compare temporal herding with other ways of constructing the buffer under the same budget, and with full accumulation, which keeps every surrogate. We also ablate the hyperparameters of buffer construction in Appendix D.4.

Buffer construction under a fixed budget. We compare temporal herding with four other ways of building the buffer under the same budget of B=1000 on CIFAR-10 at β=0.5, which keep the earliest, the latest, or randomly selected surrogates or herd toward the mean of the full pool. In Table 4a, temporal herding attains the highest AA and AIA, which shows that it forms the best

Method	Accuracy AA	AIA
Earliest-1K	64.80	69.82
Latest-1K	66.56	72.67
Random-1K	65.28	70.81
Full-Pool Herd.	66.70	72.86
Temporal Herd.	**67.96**	**73.53**

(a) Buffer construction

Dataset	Method	# Samples	Server compute	Accuracy AA	AIA
CIFAR-10	Full Accml.	16,763 (×1.00)	×1.00	**69.19**	**74.67**
	ReSCENE	10,000 (×0.60)	×0.69	67.96	73.53
CIFAR-100	Full Accml.	123,738 (×1.00)	×1.00	47.54	56.81
	ReSCENE	98,087 (×0.79)	×0.78	**49.80**	**57.71**
TinyImageNet	Full Accml.	250,528 (×1.00)	×1.00	20.02	33.18
	ReSCENE	194,355 (×0.78)	×0.78	**26.09**	**36.07**

(b) Comparison with full accumulation

Table 4: (a) Herding-buffer ablation with a fixed budget of B=1000 per class. (b) Comparison with full accumulation.

buffer within the same budget. Latest-1K also relies on the recent surrogates being less stale, but it keeps only the newest ones and discards the rest. Temporal herding instead uses the recent surrogates to define the target and selects from the whole pool, adding an earlier surrogate whenever it moves the buffer mean toward that target. It thus improves on Latest-1K by 1.40 AA points. Full-pool herding selects from the same pool toward the full-pool mean, and its lower accuracy confirms the benefit of the recent target that Proposition 1 motivates.

Comparison with full accumulation. Full accumulation, which keeps every uploaded surrogate and trains on the largest pool available, is a natural choice when server memory allows. We compare it with ReSCENE at β=0.5 on the three benchmarks. Although full accumulation keeps a small edge of 1.23 AA points on CIFAR-10, ReSCENE is more accurate on CIFAR-100 and TinyImageNet by 2.26 and 6.07 AA points while holding only 0.60 to 0.79× the surrogates and requiring 0.69 to 0.78× the server computation (Table 4b). While full accumulation keeps the stale early surrogates, temporal herding forms a compact buffer centered on the recent target. This is supported by Theorem 1, according to which the lower stale bias can bring this buffer closer to the real class mean than the whole pool. These results therefore show that temporal herding can build a better buffer even with less server memory and computation.

5.3 Server Training Budget Ablation

In this work, we assume a resource-rich server. Since server computation is still not free, we ablate the number of server epochs E_{srv} per round, which is the main factor of the server computation of ReSCENE. We reduce E_{srv} from 100 to 50, 25, and 10 on CIFAR-10 at β=0.1. Cutting E_{srv} to 10 removes 90% of the server computation, whereas AA drops by 8.33 points and AIA by only 3.32 points (Table 5). At E_{srv}=10, ReSCENE still leads Re-Fed, the strongest baseline in this setting, by 22.78 AA points. ReSCENE therefore keeps its lead over every baseline even with far fewer server epochs.

E_{srv}	Server comp.	Accuracy AA	AIA
100	1.00×	61.78	68.35
50	0.50×	58.26	66.91
25	0.25×	53.91	64.81
10	0.10×	53.45	65.03
Re-Fed		30.67	36.74

Table 5: AA and AIA (%) in the server-epoch sweep.

6 Conclusion

In this paper, we present ReSCENE, which shifts forgetting mitigation from the clients to the server in federated continual learning. In ReSCENE, clients upload condensed surrogates of their local data, and the server keeps those of earlier tasks and trains the global model on them together with the current task. We further introduce temporal herding, which compresses the pool of each task into a bounded buffer centered on the recent rounds, and our analysis establishes the condition under which this buffer represents the task more closely than full accumulation. Across three class-incremental benchmarks, ReSCENE attains the strongest accuracy at a fraction of the client computation and uplink of the baselines, and it keeps this advantage with more clients and larger models. We believe that ReSCENE offers a practical and efficient solution for federated continual learning and motivates further research on server-side forgetting mitigation.

Limitations and future work. In ReSCENE, the quality of the surrogates depends on the dataset condensation method. Developing condensation methods that better represent the local data therefore remains an important direction for future work.

AI USE STATEMENT

We used large language models (LLMs) to check grammar and improve the clarity of the text. We also used them to search the related literature more widely, to aid and polish our code. LLMs were not involved in generating scientific contents. The authors reviewed all AI-assisted output and take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

The complete procedure of ReSCENE is given in Algorithm 1, and Appendix A specifies in detail the datasets, task splits, federated protocol, and every hyperparameter of ReSCENE and of the baselines. Complete proofs of the theoretical claims, with all assumptions stated, are provided in Appendix C. Appendix D reports the per-task accuracies underlying Table 1, and Appendix E details how the computation and communication costs are measured. The source code will be released upon publication.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

Yavuz Faruk Bakman, Duygu Nur Yaldiz, Yahya Ezzeldin, and Salman Avestimehr. Federated orthogonal training: Mitigating global catastrophic forgetting in continual federated learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 47351–47375, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/cf53f12ab4ea81d942f16162f7f37e6c-Paper-Conference.pdf.

Yuanlu Chen, Alysa Ziying Tan, Siwei Feng, Han Yu, Tao Deng, Libang Zhao, and Feng Wu. General federated class-incremental learning with lightweight generative replay. *IEEE Internet of Things Journal*, 11(20): 33927–33939, 2024. doi: 10.1109/JIOT.2024.3434600.

Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10164–10173, June 2022.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–487, 08 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

Haiyang Guo, Fei Zhu, Wenzhuo Liu, Xu-Yao Zhang, and Cheng-Lin Liu. Pilora: Prototype guided incremental lora for federated class-incremental learning. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXV*, pp. 141–159, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73649-0. doi: 10.1007/978-3-031-73650-6_9. URL https://doi.org/10.1007/978-3-031-73650-6_9.

Parisa Hamedi, Roozbeh Razavi-Far, and Ehsan Hallaji. Federated continual learning: Concepts, challenges, and solutions. *Neurocomput.*, 651(C), October 2025. ISSN 0925-2312. doi: 10.1016/j.neucom.2025.130844. URL https://doi.org/10.1016/j.neucom.2025.130844.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1–2):1–210, June 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL https://doi.org/10.1561/2200000083.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Yichen Li, Qunwei Li, Haozhao Wang, Ruixuan Li, Wenliang Zhong, and Guannan Zhang. Towards efficient replay in federated incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12820–12829, June 2024.

Yichen Li, Yuying Wang, Haozhao Wang, Yining Qi, Tianzhe Xiao, and Ruixuan Li. FedSSI: Rehearsal-free continual federated learning with synergistic synaptic intelligence. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 36151–36169. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/li25cq.html.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40 (12):2935–2947, December 2018. ISSN 0162-8828. doi: 10.1109/TPAMI.2017.2773081. URL https://doi.org/10.1109/TPAMI.2017.2773081.

Chenghao Liu, Xiaoyang Qu, Jianzong Wang, and Jing Xiao. Fedet: A communication-efficient federated class-incremental learning framework based on enhanced transformer. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 3984–3992. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/443. URL https://doi.org/10.24963/ijcai.2023/443. Main Track.

Yaxin Luopan, Rui Han, Qinglong Zhang, Chi Harold Liu, Guoren Wang, and Lydia Y. Chen. Fedknow: Federated continual learning with signature task knowledge integration at edge. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 341–354, 2023. doi: 10.1109/ICDE55515.2023.00033.

Yuhang Ma, Zhongle Xie, Jue Wang, Ke Chen, and Lidan Shou. Continual federated learning based on knowledge distillation. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 2182–2188. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/303. URL https://doi.org/10.24963/ijcai.2022/303. Main Track.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. URL https://api.semanticscholar.org/CorpusID:61019113.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

Frank D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pp. 1930, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585512. doi: 10.1145/1559845.1559850. URL https://doi.org/10.1145/1559845.1559850.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=37nvvqkCo5.

Hongming Piao, Yichen Wu, Dapeng Wu, and Ying Wei. Federated continual learning via prompt-based dual knowledge transfer. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 40725–40739. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/piao24a.html.

Daiqing Qi, Handong Zhao, and Sheng Li. Better generative replay for continual federated learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cRxYWKiTan.

Zhuang Qi, Ying-Peng Tang, Lei Meng, Han Yu, Xiaoxiao Li, and Xiangxu Meng. Class-wise balancing data replay for federated class-incremental learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 145309–145334. Curran Associates, Inc., 2025. doi: 10.52202/085713-4863. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/d611d06e3207330555fbc10810e70163-Paper-Conference.pdf.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, 2010. URL https://api.semanticscholar.org/CorpusID:7534823.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/0efbe98067c6c73dba1250d2beaa81f9-Paper.pdf.

Rui Sun, Yumin Zhang, Varun Ojha, Tejal Shah, Haoran Duan, Bo Wei, and Rajiv Ranjan. Exemplar-condensed federated class-incremental learning, 2025. URL https://arxiv.org/abs/2412.18926.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(8):53625383, August 2024a. ISSN 0162-8828. doi: 10.1109/TPAMI.2024.3367329. URL https://doi.org/10.1109/TPAMI.2024.3367329.

Yuan Wang, Huazhu Fu, Renuga Kanagavelu, Qingsong Wei, Yong Liu, and Rick Siow Mong Goh. An aggregation-free federated learning for tackling data heterogeneity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26233–26242, June 2024b.

Max Welling. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 1121–1128, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553517. URL https://doi.org/10.1145/1553374.1553517.

Yuanhao Xiong, Ruochen Wang, Minhao Cheng, Felix Yu, and Cho-Jui Hsieh. Feddm: Iterative distribution matching for communication-efficient federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16323–16332, June 2023.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12073–12086. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yoon21b.html.

Jie Zhang, Chen Chen, Weiming Zhuang, and Lingjuan Lyu. Target: Federated class-continual learning via exemplar-free distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4782–4793, October 2023.

Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6514–6523, January 2023.

Appendix

A Detailed Experimental Setup and Hyperparameters

We provide the detailed experimental settings, including the datasets and task splits, the federated protocol, and the hyperparameters of ReSCENE and the baselines.

Datasets. CIFAR-10 is split into five two-class tasks, CIFAR-100 into ten ten-class tasks, and TinyImageNet into ten 20-class tasks. For these class-incremental datasets, task-wise label heterogeneity follows a Dirichlet distribution with $\beta \in \{0.1, 0.5, 1.0\}$. Office-31 evaluates domain-incremental learning over the Amazon, DSLR, and Webcam sequence with a shared 31-class label space. As specified in Sec. D.2, it uses N=10, K=4, and β=1.0, following the released code of Re-Fed (Li et al., 2024).

Common protocol. All methods use the federated protocol of Sec. 4.1, with N=20 clients, K=10 participants per round, and R=20 communication rounds per task. FedAvg, FedEWC, FedLwF, TARGET, GLFC, FedCBDR, and Re-Fed use 20 local model-training epochs per participating client and round. ReSCENE instead follows the client-side condensation schedule specified below. The default model is ResNet-18 with an incrementally expanded classifier head. CIFAR experiments use the CIFAR variant, whereas TinyImageNet uses the 64×64 variant with max pooling. Client optimization uses SGD with learning rate 0.01, momentum 0.9, weight decay 5×10^{-4}, and batch size 128. Augmentation consists of random crop and horizontal flip, with padding 4 for CIFAR and 8 for TinyImageNet.

ReSCENE. Distribution matching uses IPC 10, 25 condensation steps, synthetic-image learning rate 1.0, and projected Gaussian parameter perturbation with radius ρ=5. The server trains for 100 epochs per round using SGD with initial learning rate 0.01, cosine annealing, momentum 0.9, and weight decay 5×10^{-4}. The per-class buffer budget is B=1000, the balanced sampling exponent is α=0.5, and the logit-adjustment strength is τ=1.0. Temporal herding uses the temporal fraction λ=0.75.

FedAvg (McMahan et al., 2017). No additional hyperparameters beyond the common protocol.

FedEWC (Kirkpatrick et al., 2017). EWC penalty weight λ_{EWC}=1000. Fisher information is recomputed on each participating client's local data after every round of local training, and Table 11 includes this extra pass.

FedLwF (Li & Hoiem, 2018). Knowledge distillation temperature T_{KD}=2, KD loss weight λ_{KD}=3.

TARGET (Zhang et al., 2023). Server-side synthesis with a DeepInversion generator (latent dimension 256, base width 64, batch size 256) for 100 synthesis rounds, with a 20-round warm-up and 400 student distillation steps per round. Per synthesis round, CIFAR-10/100 use 10 generator steps, generator learning rate 0.002, loss weights $(\lambda_{\mathrm{BN}}, \lambda_{\mathrm{OH}}, \lambda_{\mathrm{adv}})$=$(10, 0.5, 1)$, and student temperature 20, and TinyImageNet uses TARGET's dataset-specific setting of 50 generator steps, learning rate 0.0002, loss weights $(0.1, 0.1, 1)$, and temperature 5. Client-side distillation on the synthesized data uses KD weight 25 and temperature 2.

GLFC (Dong et al., 2022). Gradient compensation weights λ_1=λ_2=0.5. Exemplar memory budget 20 images per class (herding selection). Prototype perturbation takes 50 SGD iterations with one tenth of the local learning rate and weight decay 10^{-5}. Server-side gradient inversion (LBFGS) runs 250 iterations with learning rate 0.1. LeNet-based encode model for prototype gradient communication.

FedCBDR (Qi et al., 2025). Task-aware temperature scaling uses τ_{old}=0.8, τ_{new}=1.2, w_{old}=1.2, and w_{new}=0.8. Exemplar budget 150 images per class. Coreset selection via SVD leverage-score sampling across clients.

Re-Fed (Li et al., 2024). Local storage budget M=1000 for CIFAR-10/100, M=2000 for TinyImageNet, and M=300 for Office-31, covering the current-task data and the cached previous samples together. The personalized informative model is trained for 40 epochs with learning rate 0.01, batch

size 64, and proximal coefficient $q(\lambda) = (1-\lambda)/(2\lambda)$ with λ=0.5. Previous samples are ranked by gradient-norm importance with the early-emphasis $1/p$ weighting.

FedDM (Xiong et al., 2023). Distribution matching with IPC=10, 25 condensation steps, synthetic-image learning rate 1.0, and Gaussian model perturbation with radius ρ=5, matching feature and logit means. The server trains for 100 epochs per round on the surrogates received in that round only, with neither balanced sampling nor logit adjustment.

FedAF (Wang et al., 2024b). Same condensation budget and server schedule as FedDM. Following the public implementation, collaborative data condensation weight λ_{cdc}=10^{-4}, local-global knowledge matching weight λ_{lgkm}=0.01, soft-label temperature 2, and model mixing coefficient 0.9. The perturbation noise is drawn at the scale of each weight tensor, since unit-variance noise drives the ResNet-18 features to overflow.

B Extended Related Work

This appendix expands the discussion of Section 2.

B.1 Anti-Forgetting in Federated Continual Learning

FCL methods are conventionally grouped into regularization-, distillation-, architecture-, and replay-based families (Wang et al., 2024a; Hamedi et al., 2025). We summarize the representative methods of each family and the limitation the family shares.

Regularization-based methods. Regularization-based methods preserve earlier-task knowledge by restraining the change of parameters that contributed most to previous tasks while learning a new task. EWC (Kirkpatrick et al., 2017) penalizes the change of each weight in proportion to its Fisher importance for the earlier tasks. FedSSI (Li et al., 2025) measures that importance on each client along the trajectory of a surrogate model that is pulled toward the global model, so the weights protected on a client are those that matter for the data of all clients. FedKNOW (Luopan et al., 2023) stores the most influential weights of each finished task and projects the update of a new task away from the directions those tasks depend on. The shared limitation is that the constraints accumulate with the task sequence. Since every finished task restricts the weight change individually, the longer the sequence, the less the model is allowed to change. Therefore, new knowledge is hindered more and more toward the end of the sequence.

Distillation-based methods. Distillation-based methods set the earlier-task model as a teacher and force the current model to reproduce its outputs while learning the new task. LwF (Li & Hoiem, 2018) records the outputs of the previous model on the new-task images and penalizes deviation from them during new-task training. CFeD (Ma et al., 2022) applies distillation on both the clients and the server. Each client learns the new task under the previous-task model as its teacher, and the server distills the received local models into the averaged global model. FedET (Liu et al., 2023) learns small adapters on top of a frozen pre-trained transformer, treating the adapter of the old tasks as the teacher for the one trained on the new task. The common drawback of these methods is that the teacher is a stale model trained on the earlier tasks alone, so forcing the current model to reproduce its outputs hinders learning of the new task. Furthermore, in class-incremental learning the class distribution completely changes from one task to the next, so the new task and the teacher model come from entirely different distributions, which makes it much harder to transfer knowledge.

Architecture-based methods. Architecture-based methods assign each task a dedicated subset of parameters or a separate module, so that the knowledge of a task stays untouched when later tasks are learned. FedWeIT (Yoon et al., 2021) splits each client's model into a shared base and sparse task-specific parameters, and exchanges the task-specific parameters among clients. Powder (Piao et al., 2024) attaches a separate set of prompts to a frozen pre-trained backbone for each task, and the server pools these prompts and shares the related ones across clients. PILoRA (Guo et al., 2024) adds, for each task, a separate low-rank adapter to a frozen pre-trained backbone, trained so that it does not overlap with the adapters of earlier tasks. In the federated setting, however, task-specific parameters are hard to coordinate across clients whose task sequences and classes differ. Moreover, as the parameters set aside for tasks accumulate, the capacity left for a new task shrinks, as it does

for regularization-based methods. Another drawback of Powder and PILoRA is that they rest on a pre-trained backbone that every client must hold.

Replay-based methods. Replay-based methods rehearse past data alongside the new data and divide into exemplar-based and generative methods by where that data comes from.

Exemplar-based methods store a subset of raw samples on the clients and mix them into local training. iCaRL (Rebuffi et al., 2017) selects exemplars by herding, and GLFC (Dong et al., 2022) keeps a per-class exemplar memory on each client and corrects the gradients of old classes during local training. ECoral (Sun et al., 2025) condenses the stored exemplars so that each stored sample carries more information. Re-Fed (Li et al., 2024) caches the previous-task samples that a personalized model scores as most informative. FedCBDR (Qi et al., 2025) lets the server choose the exemplars from obfuscated client features so that the replay is class-balanced across clients. What limits these methods is that they burden the resource-poor clients. Raw data of earlier tasks have to stay in the client's scarce, privacy-sensitive storage, and the demand increases with every new task. Furthermore, each client trains on the exemplars alongside the current task and runs the exemplar selection itself, which adds computation to every local update.

Generative methods, inspired by deep generative replay in centralized continual learning (Shin et al., 2017), synthesize past-class data with a generator in place of stored samples. TARGET (Zhang et al., 2023) inverts the previous global model into a generator on the server, sends the generated images to the clients, and has each client train on them alongside its new-task data while distilling the previous global model. FedCIL (Qi et al., 2023) equips every client and the server with a conditional generative adversarial network, so that each client replays its earlier classes from its own generator, and the server consolidates the global model on data drawn from the clients' generators. GenFCIL (Chen et al., 2024) trains a lightweight generator on the server from the class representations that clients upload, and sends generated features of the old classes back to the clients, which train on them together with their new data. The difficulty of these methods is that the synthesized data are a poor substitute for the real data of the earlier classes. A client-side generator is unstable under heterogeneous and small local data, and a server-side generator relies only on indirect signals of real data. In both cases the generated samples drift from the real distribution, which weakens the retention of earlier classes.

C Theoretical Analysis of Temporal Herding

This section provides the full theoretical analysis behind Sec. 3.5. Temporal herding uses the recent surrogates to define a target mean but selects the retained surrogates from the entire task pool. After fixing the notation (Sec. C.1), we first show why the recent surrogates make a suitable target, since they carry the least staleness under the task-end model (Proposition 1, Sec. C.2). We then bound the error of the finite buffer built toward this target (Lemma 1, Sec. C.3). Finally, we establish when this buffer represents the real task-end class mean more closely than two alternatives, full accumulation of the whole pool without any selection and full-pool herding toward the mean of the whole pool with λ=1 (Theorem 1 and Corollary 1, Sec. C.4). Remark 2 explains how replaying such a buffer mitigates forgetting.

C.1 Setup

We fix task t and class c. We use R communication rounds per task, $r \in \{1, \dots, R\}$ to index a round within task t, and B for the number of class-c surrogates retained at the task boundary. We define P_r as the index set of class-c surrogates uploaded in round r, with surrogate s_i for each $i \in P_r$, and $\mathcal{I} = \{r : |P_r| > 0\}$ as the set of rounds containing at least one such surrogate. We then define

$$P = \bigcup_{r \in \mathcal{I}} P_r, \qquad W = \bigcup_{r \in \mathcal{I}_W} P_r \subseteq P$$

as the candidate index set and the recent target index set, respectively. The parameter $\lambda \in (0, 1]$ specifies the fraction of recent communication rounds used to form the target, and $w = \lceil \lambda R \rceil$ is the resulting number of rounds. We define $\mathcal{I}_W = \{r \in \mathcal{I} : r \geq R - w + 1\}$. If this set is empty, we use $\mathcal{I}_W = \{\max \mathcal{I}\}$. We denote the global model after the server update in round r by $\theta_t^{(r)}$. Hence $\theta_t^{(R)}$ is the task-end model. We define the task-end feature vector of each candidate and, for any

nonempty index set $A \subseteq P$, its feature mean as

$$\boldsymbol{z}_i = \frac{\phi_{\theta_t^{(R)}}(s_i)}{\|\phi_{\theta_t^{(R)}}(s_i)\|}, \qquad \boldsymbol{m}_A = \frac{1}{|A|}\sum_{i\in A}\boldsymbol{z}_i. \tag{3}$$

Here $\boldsymbol{z}_i$ is the unit-normalized task-end feature vector of surrogate s_i, and i indexes a candidate surrogate. In particular, $\boldsymbol{m}_W$ is the mean of the recent candidates' normalized feature vectors and serves as the herding target. As λ is a fraction of rounds and $|P_r|$ varies across rounds, the sample fraction $|W|/|P|$ differs from λ in general. Temporal herding first uses the recent index set W to define the target mean $\boldsymbol{m}_W$, and then keeps every surrogate indexed by the full task pool P eligible and constructs a buffer by greedy selection toward this target. We use $\ell \in \{1, \dots, B\}$ for this selection step and H_ℓ for the indices selected through step ℓ, with $H_0 = \varnothing$. At step ℓ, $q \in P \setminus H_{\ell-1}$ is the index of a surrogate that has not yet been selected, s_q is the corresponding surrogate, and $\boldsymbol{m}_{H_{\ell-1}\cup\{q\}}$ is the feature mean that would result from adding s_q. The symbol i_ℓ denotes the index of the surrogate actually selected at step ℓ. Temporal herding selects one surrogate without replacement at each step according to

$$i_\ell \in \operatorname*{arg\,min}_{q\in P\setminus H_{\ell-1}} \|\boldsymbol{m}_W - \boldsymbol{m}_{H_{\ell-1}\cup\{q\}}\|^2, \qquad H_\ell = H_{\ell-1}\cup\{i_\ell\}. \tag{4}$$

C.2 Recent Surrogates Reduce Task-End Error

We first show why temporal herding takes the recent surrogates as its target. A surrogate produced in round r is matched to the real class under the model available in that round, but temporal herding evaluates all candidates under the task-end model. We quantify the mismatch introduced by this model change. For a model θ, let

$$\boldsymbol{m}_r(\theta) = \frac{1}{|P_r|}\sum_{i\in P_r}\frac{\phi_\theta(s_i)}{\|\phi_\theta(s_i)\|}, \qquad \boldsymbol{\mu}(\theta) = \mathbb{E}_{x\sim\mathcal{D}_c}\left[\frac{\phi_\theta(x)}{\|\phi_\theta(x)\|}\right]$$

be the mean normalized feature of the round-r surrogates and of the real class-c data distribution $\mathcal{D}_c$, respectively. Under the task-end model, $\boldsymbol{m}_r(\theta_t^{(R)}) = \frac{1}{|P_r|}\sum_{i\in P_r}\boldsymbol{z}_i$, and $\boldsymbol{\mu} = \boldsymbol{\mu}(\theta_t^{(R)})$ is the real class mean that the buffer should represent. The matching error of the round-r surrogates is measured under the global model $\theta_t^{(r-1)}$ used in that round,

$$\delta_r = \|\boldsymbol{m}_r(\theta_t^{(r-1)}) - \boldsymbol{\mu}(\theta_t^{(r-1)})\|.$$

To transfer this matching error to the task-end feature space, we require that normalized feature means change smoothly along the bounded within-task model trajectory. This is a standard local smoothness condition, and it is needed only over the parameters that the model visits within one task.

Proposition 1 (Task-end surrogate error bound)**.** *Suppose that for any two model states $\theta_t^{(a)}$ and $\theta_t^{(b)}$ on the within-task trajectory, where $0 \le a \le b \le R$, and each $\boldsymbol{m} \in \{\boldsymbol{\mu}\} \cup \{\boldsymbol{m}_r : r \in \mathcal{I}\}$,*

$$\|\boldsymbol{m}(\theta_t^{(b)}) - \boldsymbol{m}(\theta_t^{(a)})\| \le L_\phi\|\theta_t^{(b)} - \theta_t^{(a)}\|. \tag{5}$$

Then the task-end error of the round-r surrogates satisfies

$$\|\boldsymbol{m}_r(\theta_t^{(R)}) - \boldsymbol{\mu}\| \le \delta_r + 2L_\phi\|\theta_t^{(R)} - \theta_t^{(r-1)}\|. \tag{6}$$

Proof. Applying the triangle inequality to the left-hand side of Eq. (6) through the model $\theta_t^{(r-1)}$ and then Eq. (5) gives

$$\begin{aligned}
&\|\boldsymbol{m}_r(\theta_t^{(R)}) - \boldsymbol{\mu}(\theta_t^{(R)})\| \\
&\le \|\boldsymbol{m}_r(\theta_t^{(R)}) - \boldsymbol{m}_r(\theta_t^{(r-1)})\| + \|\boldsymbol{m}_r(\theta_t^{(r-1)}) - \boldsymbol{\mu}(\theta_t^{(r-1)})\| \\
&\quad + \|\boldsymbol{\mu}(\theta_t^{(r-1)}) - \boldsymbol{\mu}(\theta_t^{(R)})\| \\
&\le L_\phi\|\theta_t^{(R)} - \theta_t^{(r-1)}\| + \delta_r + L_\phi\|\theta_t^{(R)} - \theta_t^{(r-1)}\| \\
&= \delta_r + 2L_\phi\|\theta_t^{(R)} - \theta_t^{(r-1)}\|.
\end{aligned}$$

Remark 1 (Recent targets reduce the staleness component)**.** Proposition 1 decomposes the task-end error of a round's surrogates into their matching error δ_r and a staleness term determined by how

far the model is subsequently trained. Recent surrogates are associated with model states closer to $\theta_t^{(R)}$ and therefore reduce this staleness component, which motivates defining the herding target with recent rounds.

C.3 Temporal Herding Approximates the Target within a Bounded Error

Proposition 1 motivates the recent target $\boldsymbol{m}_W$, but ReSCENE retains only B surrogates per class. We now show that the implemented greedy selection keeps the finite buffer close to this target. Using the set-mean definition in Eq. (3), $\boldsymbol{m}_P$ and $\boldsymbol{m}_{H_B}$ denote the feature means of the candidate pool and the final herded buffer, respectively. Both are computed using the task-end model $\theta_t^{(R)}$. We also define

$$\Delta_W = \max_{i \in P} \|\boldsymbol{z}_i - \boldsymbol{m}_W\|,$$

which is the maximum feature-space distance from a candidate to the recent target $\boldsymbol{m}_W$. As $\|\boldsymbol{z}_i\| = 1$ and $\boldsymbol{m}_W$ lies in the unit ball, $\Delta_W \leq 2$.

Tracking the greedy selection error. To track how closely the selected candidates match $\boldsymbol{m}_W$, we define the residual before selection step ℓ as

$$\boldsymbol{r}_{\ell-1} = (\ell-1)\boldsymbol{m}_W - \sum_{j \in H_{\ell-1}} \boldsymbol{z}_j, \qquad \boldsymbol{r}_0 = 0.$$

For $\ell > 1$, the set-mean definition in Eq. (3) gives

$$\boldsymbol{r}_{\ell-1} = (\ell-1)(\boldsymbol{m}_W - \boldsymbol{m}_{H_{\ell-1}}),$$

so $\boldsymbol{r}_{\ell-1}$ is the feature-mean error after $\ell-1$ selections, scaled by the number of selected candidates. The base case $\boldsymbol{r}_0 = 0$ follows from $H_0 = \varnothing$ before the first selection.

For an unselected candidate index $q \in P \setminus H_{\ell-1}$, the hypothetical mean error after adding surrogate s_q is

$$\boldsymbol{m}_W - \boldsymbol{m}_{H_{\ell-1}\cup\{q\}} = \frac{1}{\ell}\left[\boldsymbol{r}_{\ell-1} - (\boldsymbol{z}_q - \boldsymbol{m}_W)\right],$$
$$\|\boldsymbol{m}_W - \boldsymbol{m}_{H_{\ell-1}\cup\{q\}}\|^2 = \frac{1}{\ell^2}\|\boldsymbol{r}_{\ell-1} - (\boldsymbol{z}_q - \boldsymbol{m}_W)\|^2.$$

The positive factor $1/\ell^2$ is shared by all unselected candidate indices q, so the same surrogate minimizes both objectives and Eq. (4) is equivalent to

$$i_\ell \in \operatorname*{arg\,min}_{q \in P \setminus H_{\ell-1}} \|\boldsymbol{r}_{\ell-1} - (\boldsymbol{z}_q - \boldsymbol{m}_W)\|^2. \tag{7}$$

After temporal herding selects surrogate s_{i_ℓ}, whose index is i_ℓ, the actual residual is updated as

$$\boldsymbol{r}_\ell = \boldsymbol{r}_{\ell-1} - (\boldsymbol{z}_{i_\ell} - \boldsymbol{m}_W) = \ell(\boldsymbol{m}_W - \boldsymbol{m}_{H_\ell}). \tag{8}$$

The residual thus gives an equivalent form of the greedy mean-matching rule, in which each step picks the candidate that minimizes the next residual norm.

Condition for a finite-buffer guarantee. This equivalence identifies what the greedy rule minimizes. To bound the residual over all B selections, the remaining pool must also retain a candidate that can move the selected mean toward the target.

Assumption 1 (Directional coverage)**.** *For every $\ell = 1, \ldots, B$, the remaining candidate-index set satisfies*

$$\max_{q \in P \setminus H_{\ell-1}} \langle \boldsymbol{r}_{\ell-1}, \boldsymbol{z}_q - \boldsymbol{m}_W \rangle \geq 0. \tag{9}$$

This condition is natural for temporal herding. The target $\boldsymbol{m}_W$ is the mean of the recent index set $W \subseteq P$, and the maximum of a set along any direction is at least its mean, so for every direction some point of W lies at or beyond the target. Such a point, once added, moves the selected mean toward the target, and the condition holds at step ℓ whenever such a point of W along $\boldsymbol{r}_{\ell-1}$ has not yet been selected. For $\ell \geq 2$, the residual $\boldsymbol{r}_{\ell-1}$ is a positive multiple of $\boldsymbol{m}_W - \boldsymbol{m}_{H_{\ell-1}}$, so this is the condition stated in Sec. 3.5, and it holds trivially for $\ell = 1$ since $\boldsymbol{r}_0 = 0$.

Lemma 1 (Finite-budget bound for greedy temporal herding). *For $B \le |P|$, under Assumption 1, the distinct, no-replacement greedy rule in (4) satisfies*

$$\|\boldsymbol{m}_{H_B} - \boldsymbol{m}_W\| \le \frac{\Delta_W}{\sqrt{B}} \le \frac{2}{\sqrt{B}}. \tag{10}$$

Proof. By Eq. (7), the implemented greedy rule selects the surrogate whose index minimizes the new residual norm. By Assumption 1, there is an unselected candidate index $q^\star$ with $\langle \boldsymbol{r}_{\ell-1}, \boldsymbol{z}_{q^\star} - \boldsymbol{m}_W \rangle \ge 0$. Greedy optimality therefore gives

$$\begin{aligned}\|\boldsymbol{r}_\ell\|^2 &\le \|\boldsymbol{r}_{\ell-1} - (\boldsymbol{z}_{q^\star} - \boldsymbol{m}_W)\|^2 \\ &= \|\boldsymbol{r}_{\ell-1}\|^2 - 2\langle \boldsymbol{r}_{\ell-1}, \boldsymbol{z}_{q^\star} - \boldsymbol{m}_W \rangle + \|\boldsymbol{z}_{q^\star} - \boldsymbol{m}_W\|^2 \\ &\le \|\boldsymbol{r}_{\ell-1}\|^2 + \Delta_W^2.\end{aligned}$$

The last step uses the nonnegative inner product from the assumption and $\|\boldsymbol{z}_{q^\star} - \boldsymbol{m}_W\| \le \Delta_W$. Induction from $\boldsymbol{r}_0 = 0$ yields $\|\boldsymbol{r}_B\|^2 \le B\Delta_W^2$. Finally, the residual definition and $\sum_{i \in H_B} \boldsymbol{z}_i = B\boldsymbol{m}_{H_B}$ give

$$\boldsymbol{r}_B = B(\boldsymbol{m}_W - \boldsymbol{m}_{H_B}).$$

Hence $\|\boldsymbol{m}_{H_B} - \boldsymbol{m}_W\| = \|\boldsymbol{r}_B\|/B \le \Delta_W/\sqrt{B}$, proving (10).

The approximation error of Lemma 1 shrinks as $1/\sqrt{B}$ and is at most $2/\sqrt{B}$, so the finite buffer stays close to the recent target that Proposition 1 motivates.

C.4 When the Bounded Buffer Surpasses Full Accumulation

Lemma 1 shows that the buffer stays close to the recent target. To compare it with full accumulation, we next describe how the task-end error of the surrogates changes over rounds. Recall that $\boldsymbol{\mu} = \boldsymbol{\mu}(\theta_t^{(R)})$ is the real task-end class mean and that $\boldsymbol{m}_r(\theta_t^{(R)}) = \frac{1}{n_r}\sum_{i \in P_r} \boldsymbol{z}_i$, with $n_r = |P_r|$, is the task-end mean of the surrogates produced in round r. Proposition 1 bounds the task-end error of each round by a staleness term that shrinks with r, which suggests that the magnitude of the error decreases over rounds. Its direction is shared across rounds as well. Within a task the model moves along a single trajectory from $\theta_t^{(0)}$ to $\theta_t^{(R)}$, and a surrogate matched under an earlier state is displaced, under the task-end model, by the change of the feature map along that trajectory, so the errors of different rounds point in a common direction, with earlier rounds displaced further. Assumption 2 states this structure, with a deviation of bounded norm that covers client sampling and round-to-round variation.

Assumption 2 (Coherent stale bias). *There exist a unit vector $\boldsymbol{u}$, nonnegative coefficients a_r that are non-increasing over $r \in \mathcal{I}$, and deviation vectors $\boldsymbol{\xi}_r$ such that*

$$\boldsymbol{m}_r(\theta_t^{(R)}) - \boldsymbol{\mu} = a_r \boldsymbol{u} + \boldsymbol{\xi}_r, \qquad \|\boldsymbol{\xi}_r\| \le \sigma. \tag{11}$$

Assumption 2 places little restriction on the errors themselves, since the deviation $\boldsymbol{\xi}_r$ absorbs whatever part of the error is not aligned with $\boldsymbol{u}$, and any error vectors satisfy it with a large enough σ. What matters is how large the coherent part is relative to this deviation, which is exactly what the condition of Theorem 1 compares.

The sample-weighted systematic biases of the full and recent pools are

$$A_P = \frac{\sum_{r \in \mathcal{I}} n_r a_r}{\sum_{r \in \mathcal{I}} n_r}, \qquad A_W = \frac{\sum_{r \in \mathcal{I}_W} n_r a_r}{\sum_{r \in \mathcal{I}_W} n_r}.$$

As W is a suffix and a_r is non-increasing, every excluded early coefficient is at least every included recent coefficient. Hence $A_W \le A_P$, so the recent target removes a stale bias of magnitude $A_P - A_W$ relative to full accumulation. It remains to determine when this removed bias outweighs the deviation of the surrogates and the approximation error of Lemma 1.

Theorem 1 (Greedy improvement over full accumulation). *Under Assumptions 1 and 2, the buffer selected by the greedy rule in Eq. (4) is strictly closer to $\boldsymbol{\mu}$ than full accumulation whenever*

$$A_P - A_W > 2\sigma + \frac{\Delta_W}{\sqrt{B}}. \tag{12}$$

Specifically,

$$\|\boldsymbol{m}_{H_B} - \boldsymbol{\mu}\| < \|\boldsymbol{m}_P - \boldsymbol{\mu}\|. \tag{13}$$

Proof. Averages of the deviations in (11) have norm at most σ. The reverse triangle inequality therefore gives $\|\boldsymbol{m}_P - \boldsymbol{\mu}\| \geq A_P - \sigma$, while the triangle inequality gives $\|\boldsymbol{m}_W - \boldsymbol{\mu}\| \leq A_W + \sigma$. Lemma 1 and another triangle inequality give

$$\|\boldsymbol{m}_{H_B} - \boldsymbol{\mu}\| \leq A_W + \sigma + \frac{\Delta_W}{\sqrt{B}}.$$

Under Condition (12), these bounds combine as

$$\|\boldsymbol{m}_{H_B} - \boldsymbol{\mu}\| \leq A_W + \sigma + \frac{\Delta_W}{\sqrt{B}} < A_P - \sigma \leq \|\boldsymbol{m}_P - \boldsymbol{\mu}\|,$$

which proves Eq. (13).

Corollary 1 (Improvement over full-pool herding)**.** *Let H_B^P be the buffer selected by the same greedy rule with the full-pool target $\boldsymbol{m}_P$ in place of $\boldsymbol{m}_W$ (that is, λ=1), and let $\Delta_P = \max_{i \in P} \|\boldsymbol{z}_i - \boldsymbol{m}_P\|$. Under Assumptions 1 (for both targets) and 2,*

$$A_P - A_W > 2\sigma + \frac{\Delta_W + \Delta_P}{\sqrt{B}} \quad \Longrightarrow \quad \|\boldsymbol{m}_{H_B} - \boldsymbol{\mu}\| < \|\boldsymbol{m}_{H_B^P} - \boldsymbol{\mu}\|.$$

Proof. Lemma 1 applied to the full-pool target gives $\|\boldsymbol{m}_{H_B^P} - \boldsymbol{m}_P\| \leq \Delta_P/\sqrt{B}$, so by the reverse triangle inequality and $\|\boldsymbol{m}_P - \boldsymbol{\mu}\| \geq A_P - \sigma$, $\|\boldsymbol{m}_{H_B^P} - \boldsymbol{\mu}\| \geq A_P - \sigma - \Delta_P/\sqrt{B}$. The proof of Theorem 1 gives $\|\boldsymbol{m}_{H_B} - \boldsymbol{\mu}\| \leq A_W + \sigma + \Delta_W/\sqrt{B}$, and the stated margin makes the upper bound smaller than the lower bound.

When the condition holds. Condition (12) compares three quantities that are governed by different parts of the setting. The deviation bound σ is set by how far the surrogates of a round deviate from their systematic bias, which comes from client sampling and round-to-round variation, and the approximation error $\Delta_W/\sqrt{B}$ is set by the budget, with $\Delta_W \leq 2$ for unit-normalized features. Neither depends on how many rounds a task runs. The removed bias $A_P - A_W$ does. As the rounds proceed the model keeps learning, the surrogates of the early rounds are matched under models ever farther from $\theta_t^{(R)}$, and the bound of Proposition 1 on their task-end error grows relative to that of the late rounds, so the gap between the average bias of the whole pool and that of the recent window widens. With many rounds per task, as in our protocol with R=20, the removed bias is therefore expected to exceed the two fixed terms. This is the regime in which full accumulation keeps the most surrogates and costs the most memory and server compute, and Theorem 1 states that a bounded buffer aimed at the recent rounds then represents the class more closely than the whole pool, so the saving in memory comes together with a gain in fidelity.

Remark 2 (From reduced stale bias to forgetting mitigation)**.** Equation (13) shows that a bounded buffer aimed at the recent rounds represents a completed task more closely than retaining every surrogate. Full accumulation gives early, stale generations the same influence on its mean as later generations. Temporal herding lets the recent generations define the target while keeping every surrogate eligible for selection, so useful earlier samples still enter the buffer while the recent generations set its representation. The resulting finite buffer provides better-matched old-class training signals during later replay, which supports forgetting mitigation.

D Additional Experiments

We report additional experiments of ReSCENE. We report the per-task accuracies behind Table 1 (Sec. D.1), evaluate ReSCENE in the domain-incremental setting (Sec. D.2), and compare it with the replay baselines under the same replay budget for a fair comparison (Sec. D.3). We also sweep the two hyperparameters of temporal herding (Sec. D.4) and ablate the class-imbalance corrections of the server objective (Sec. D.5).

D.1 Detailed Per-Task Results

AA and AIA in Table 1 average the accuracy over tasks, so two methods with the same average can forget in very different ways. To show how each method retains the earlier tasks, we report the final accuracy of every task after the complete sequence, for all methods, benchmarks, and β values, in Table 6, and plot the same values in Figure 4. T0 is the oldest task, so forgetting appears as low accuracy on the left of each row.

Per-task accuracy. The baselines fall into three groups. FedAvg and FedEWC keep nothing of the earlier tasks, and neither do the SA-FL methods FedDM and FedAF. Their accuracy is at most 0.1% on every task but the last, so their AA reduces to the accuracy of the last task divided by the number of tasks. FedCBDR and Re-Fed retain the earlier tasks in part, but their accuracy rises steeply toward the most recent ones. FedCBDR on CIFAR-100 at β=0.5 reaches 66.9% on T9 against 22.1% on T0, and Re-Fed on TinyImageNet at β=0.5 reaches 45.8% on T8 against 5.6% on T0. GLFC and TARGET skew the other way on CIFAR-100 and TinyImageNet and lose the newest tasks. At β=1.0, TARGET holds T0 at 46.5% on CIFAR-100 and 42.8% on TinyImageNet, while its T9 falls to 21.3% and 6.4%. ReSCENE instead keeps every task within a high band. Its lowest per-task accuracy exceeds the lowest of every baseline in all nine settings, 36.2 to 46.9% against at most 24.4% on CIFAR-10, 32.3 to 44.8% against at most 17.8% on CIFAR-100, and 19.9 to 21.6% against at most 14.8% on TinyImageNet. On CIFAR-10, it also holds the oldest task at 75 to 78%, where no baseline exceeds 39%.

Accuracy across the task sequence. Figure 4 plots the per-task accuracies of Table 6 as curves over the task index, which shows how each method distributes its accuracy between the old and the recent tasks. The curves of FedAvg and FedEWC stay at zero and rise only at the final task, those of FedCBDR and Re-Fed climb toward the most recent tasks, and those of GLFC and TARGET decline toward them on CIFAR-100 and TinyImageNet. The curve of ReSCENE is not flat, since it dips on harder tasks such as T1 on CIFAR-10 and varies by 23 points on TinyImageNet at β=1.0. It nevertheless stays high across the whole sequence, and its lowest point lies above the lowest point of every baseline in every panel, so no task is sacrificed regardless of when it arrived. This follows from the server replaying every completed task in every round.

D.2 Domain-Incremental Learning

Federated incremental learning is defined in prior work by two scenarios, new classes and new domains (Li et al., 2024), and the main text covers the first. We evaluate whether server-side surrogate replay also holds when each task changes the visual domain while the label space stays fixed.

Setting. We learn the Amazon, DSLR, and Webcam domains of Office-31 (Saenko et al., 2010) in sequence under a single shared 31-class classifier. Following Re-Fed (Li et al., 2024) we set N=10 clients, K=4 participants, β=1.0, 20 rounds per domain, and 20 local epochs for the model-training baselines, and everything else follows Table 1.

Method	Domain			Accuracy	
	Amazon	DSLR	Webcam	AA	AIA
FedAvg	18.83	23.23	65.19	35.75	36.21
FedEWC	19.18	21.21	61.39	33.93	35.94
FedLwF	25.75	37.37	56.96	40.03	37.07
TARGET	29.48	20.20	39.87	29.85	34.37
GLFC	36.94	17.17	11.39	21.84	22.81
FedCBDR	55.42	48.48	53.80	52.57	49.33
Re-Fed	48.31	42.42	53.80	48.18	47.77
ReSCENE	**56.48**	**65.66**	**76.58**	**66.24**	**56.10**

Table 7: Domain-incremental results on Office-31 (%). Final accuracy on each domain, AA, and AIA.

Results. ReSCENE attains the highest accuracy on every domain (Table 7), reaching 56.48%, 65.66%, and 76.58% on Amazon, DSLR, and Webcam, while FedCBDR reaches $55.42/48.48/53.80$ and Re-Fed reaches $48.31/42.42/53.80$. This per-domain advantage shows that uploading condensed surrogates and training a single model on the server is more effective than training separate client models and aggregating their weights. Furthermore, ReSCENE reaches 66.24% AA and 56.10% AIA, the best of all methods. As it gathers the surrogates of all domains at the server and trains one model over them, ReSCENE maintains high accuracy on the earlier domains, preserving knowledge across domain transitions and limiting forgetting as the input distribution shifts. Overall, ReSCENE shows its effectiveness not only in class-incremental settings but also under domain-incremental distribution shift.

Method	β=0.1					β=0.5					β=1.0				
	T0	T1	T2	T3	T4	T0	T1	T2	T3	T4	T0	T1	T2	T3	T4
FedAvg	0.0	0.0	0.0	0.0	50.0	0.0	0.0	0.0	0.0	79.2	0.0	0.0	0.0	0.0	73.5
FedEWC	0.0	0.0	0.0	0.0	50.0	0.0	0.0	0.0	0.0	87.6	0.0	0.0	0.0	0.0	75.6
FedLwF	31.7	0.2	0.0	0.4	19.6	25.1	11.3	30.0	2.8	15.5	32.6	18.4	13.3	45.5	4.3
TARGET	35.8	0.0	0.2	0.4	13.4	38.7	6.7	20.9	22.2	19.6	29.0	18.9	6.3	30.6	9.5
GLFC	0.4	0.2	3.3	20.5	50.1	4.9	4.3	16.8	58.2	92.7	13.9	7.0	28.9	74.3	91.6
FedCBDR	1.7	1.3	3.8	59.9	63.0	21.2	11.7	34.6	73.4	93.0	23.9	12.6	43.2	77.4	93.2
Re-Fed	37.5	9.1	31.4	54.1	21.3	23.4	17.9	50.9	80.5	80.9	31.1	24.4	57.8	89.2	80.4
FedDM	0.0	0.0	0.0	0.0	78.5	0.0	0.0	0.0	0.0	88.6	0.0	0.0	0.0	0.0	90.5
FedAF	0.0	0.0	0.0	0.0	80.0	0.0	0.0	0.0	0.0	88.5	0.0	0.0	0.0	0.0	90.3
ReSCENE	75.3	36.2	52.1	70.5	74.8	77.4	43.9	57.8	74.4	86.3	78.1	46.9	61.9	75.4	85.9

(a) CIFAR-10

Method	β	CIFAR-100										TinyImageNet									
		T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
FedAvg	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	58.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	48.1
FedEWC	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	58.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	48.0
FedLwF	0.1	3.1	0.2	0.8	5.8	7.0	9.0	9.5	24.3	8.3	22.1	2.3	6.6	4.5	8.2	10.1	6.2	11.3	15.1	21.9	23.8
TARGET	0.1	2.0	1.2	1.9	7.8	11.0	15.4	9.7	17.3	7.5	11.0	17.2	13.2	15.0	16.2	15.2	13.2	13.0	13.2	10.4	9.8
GLFC	0.1	11.7	10.5	21.3	17.0	27.4	28.7	27.3	22.9	10.2	1.4	7.7	7.9	12.5	12.0	15.5	13.9	6.4	4.8	0.8	1.8
FedCBDR	0.1	17.4	12.1	27.8	16.7	27.5	39.8	42.4	43.5	58.7	56.2	16.4	10.6	18.5	17.5	14.2	19.2	23.2	34.2	45.3	11.9
Re-Fed	0.1	12.1	13.1	24.6	20.9	26.1	39.2	44.5	43.9	47.8	23.6	1.1	1.7	3.1	7.3	9.4	14.8	22.8	33.0	36.9	19.8
FedDM	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	62.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	25.2
FedAF	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	63.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	25.8
ReSCENE	0.1	48.6	38.5	51.7	42.6	46.9	51.4	52.1	42.5	42.3	32.3	27.6	24.3	26.2	26.1	19.9	23.1	21.4	25.1	23.8	20.9
FedAvg	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	81.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	60.3
FedEWC	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	81.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	60.8
FedLwF	0.5	3.8	1.6	15.9	16.9	21.4	27.1	29.3	27.3	28.1	35.1	14.4	11.9	9.8	18.9	19.0	22.5	19.7	25.7	27.2	22.8
TARGET	0.5	29.9	6.8	16.5	19.0	20.8	27.1	23.9	22.9	19.6	15.3	41.3	16.2	23.0	29.4	19.8	26.6	18.4	19.3	12.6	8.4
GLFC	0.5	18.1	16.9	28.3	20.8	26.6	27.8	27.5	18.3	15.4	2.5	14.8	12.6	15.1	13.8	16.6	14.0	7.8	8.3	1.4	2.0
FedCBDR	0.5	22.1	17.1	35.4	25.2	34.8	44.3	49.1	54.3	67.2	66.9	19.8	14.8	21.1	22.3	18.7	24.0	25.7	35.7	46.4	29.6
Re-Fed	0.5	10.8	7.7	19.5	16.6	27.9	37.4	47.2	56.8	65.0	50.3	5.6	2.8	8.6	11.7	15.8	22.9	28.2	40.0	45.8	36.5
ReSCENE	0.5	50.8	44.1	56.4	45.6	51.4	51.6	52.5	46.2	51.1	48.3	25.0	23.5	26.1	33.5	39.8	26.6	20.6	23.3	22.4	20.1
FedAvg	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	82.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	62.1
FedEWC	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	82.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	62.3
FedLwF	1.0	5.7	10.2	18.0	19.6	19.4	33.3	33.1	28.8	34.6	33.3	9.1	6.6	9.4	18.0	25.6	24.7	23.6	22.4	28.1	24.8
TARGET	1.0	46.5	10.5	20.6	25.6	31.8	28.4	24.3	23.4	16.6	21.3	42.8	18.0	20.1	28.9	19.4	25.0	16.6	17.6	11.8	6.4
GLFC	1.0	18.9	15.8	28.2	21.2	27.5	28.7	28.6	24.2	13.5	2.0	14.1	13.5	14.5	13.8	15.9	16.8	5.9	7.7	2.5	3.4
FedCBDR	1.0	22.6	17.8	35.7	24.1	33.5	42.0	50.0	53.5	70.3	68.5	19.9	14.4	20.1	21.2	18.1	23.8	24.8	35.2	49.2	34.1
Re-Fed	1.0	8.9	8.8	20.4	16.6	31.7	37.3	49.0	55.8	64.4	53.5	5.1	4.5	7.2	12.4	17.1	26.5	29.1	40.5	48.9	37.8
ReSCENE	1.0	50.8	44.8	56.2	46.6	51.2	53.5	54.1	47.9	52.9	49.0	26.5	24.4	26.0	26.5	21.6	26.0	22.5	36.1	44.8	32.5

(b) CIFAR-100 and TinyImageNet

Table 6: Final accuracy (%) on each task after the complete sequence. T0 is the oldest task, and each row averages to the AA of the run.

D.3 Comparison under a Matched Replay Budget

As in ReSCENE, the replay baselines revisit data of earlier tasks during training, so the amount of replay data could explain the gap between them. At the default budgets of Table 1, ReSCENE already holds fewer replay images than FedCBDR and TARGET and still outperforms them. For a fair comparison, we rerun every replay baseline with the same total number of replay images as ReSCENE.

Setting. We use CIFAR-10 at β=0.5 with N=20 clients and count the replay images held by the server and all clients at the end of the sequence. Each method has one memory parameter, which Table 8 lists as Budget. It is the number of exemplars per class on every client for GLFC and FedCBDR, the number of synthetic images per task for TARGET, the total storage of a client for Re-Fed, and the per-class server buffer B for ReSCENE. We set each budget so that the total becomes ReSCENE's 10,000 images. TARGET keeps its synthesis schedule while each client receives a 500-image subset, and Re-Fed already holds 10,253 images at its default and is left unchanged.

Results. With the same 10,000 replay images, ReSCENE outperforms every replay baseline by 17.2 to 48.3 AA points and by 10.8 to 35.9 AIA points (Table 8), so matching the budget does not close the gap. GLFC gains 5.29 AA points when its memory grows to 50 exemplars per class, and FedCBDR loses 13.93 points when its memory shrinks to the same size, which shows how strongly

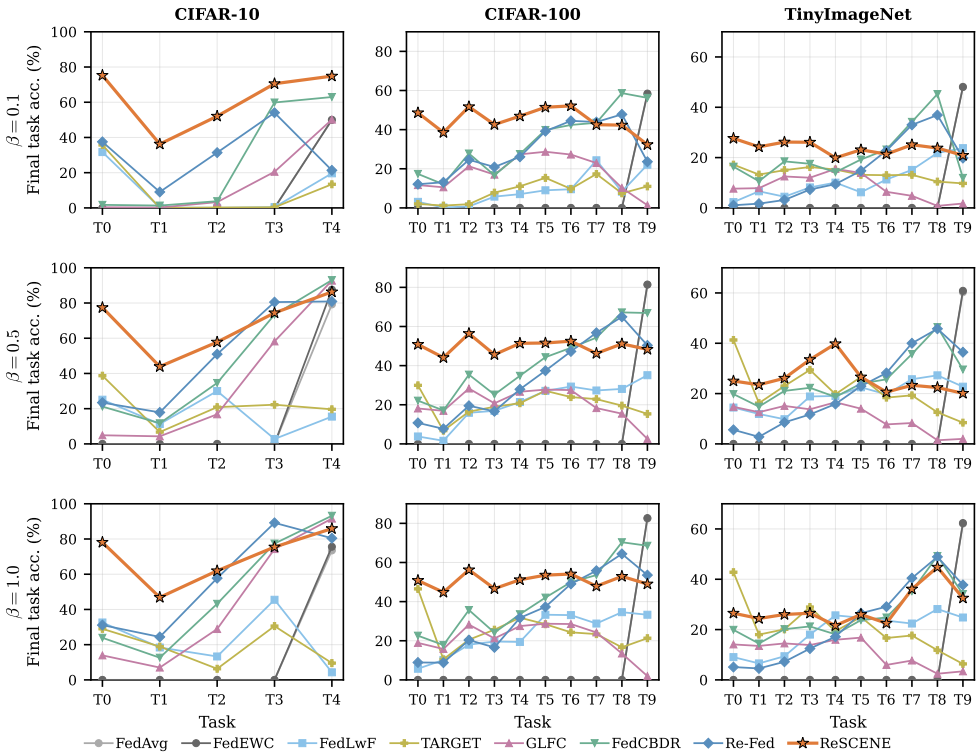


Figure 4: Final accuracy of each task after the complete sequence for the methods of Table 1, one column per benchmark and one row per β (values in Table 6).

Method	Setting of Table 1					Matched setting (10,000 in total)			
	Budget	Per client	Server + clients	AA	AIA	Budget	Server + clients	AA	AIA
TARGET	25,600	25,600	512,000	21.66	36.76	500	10,000	19.62	37.62
GLFC	20	200	4,000	35.38	48.90	50	10,000	40.67	51.69
FedCBDR	150	1,500	30,000	46.78	57.64	50	10,000	32.85	49.86
Re-Fed	1,000	≈513	10,253	50.73	62.69	1,000	10,253	50.73	62.69
ReSCENE	1,000	0	10,000	**67.96**	**73.53**	1,000	10,000	**67.96**	**73.53**

Table 8: Replay images and accuracy on CIFAR-10 (β=0.5) at the budgets of Table 1 and after matching every baseline to ReSCENE's 10,000 images. Budget denotes the memory parameter of each method, defined in the text.

client-side replay depends on the amount of stored data. In the exemplar-based methods, the budget is split across 20 clients, so each client replays only a small, skewed cache of its own data, and the knowledge of these caches is merged only through weight averaging. ReSCENE instead keeps a single copy of the replay data at the server and trains on it together with the current task in one objective at every round. Its advantage therefore comes from where and how the replay data is used, which relies on a resource-rich server that stores the buffer and trains the model.

D.4 Additional Herding-Buffer Sweeps

Temporal herding has two hyperparameters, the fraction λ of the recent rounds that defines the target and the per-class budget B. The main text fixes them at λ=0.75 and B=1000, and we examine how each of them affects the accuracy.

Setting. On CIFAR-10 at β=0.5, we vary $\lambda \in \{0.25, 0.5, 0.75, 1\}$ at B=1000 and $B \in \{500, 1000, 1500, 2000\}$ at λ=0.75, keeping the rest of the protocol. The whole task pool stays eligible for selection in every run, and only the target or the budget changes, so λ=1 becomes full-pool herding.

Temporal target fraction. λ=0.75 attains the highest AA and AIA among the four values (Table 9a). A small λ defines the target from only the last few rounds, so the target rests on few surrogates and can be biased toward the participants of those rounds, which lowers AA to 66.92 at λ=0.25. At the other end, λ=1 averages over every round, including the stale early surrogates, and reduces to full-pool herding with 66.70 AA. The intermediate window removes the earliest rounds from the target while still averaging over enough surrogates. This matches the analysis of Sec. 3.5, where moving the target to the recent window lowers the stale bias from A_P to A_W (Corollary 1), while a window that is too narrow leaves the target to the scatter of a few rounds.

Per-class budget. Raising B from 500 to 1000 improves AA by 6.85 points (Table 9b), since Lemma 1 bounds the error of summarizing the target by $\Delta_W/\sqrt{B}$, which is 41% larger at B=500. On CIFAR-10, a class collects at most 2,000 surrogates in a task and about 1,700 on average, so B=2000 keeps the whole pool and coincides with full accumulation. The bounded buffer at B=1000 stays within 1.5 AA points of the larger budgets while holding 40% fewer surrogates than the whole pool, so temporal herding summarizes each task with far less server memory and replay computation. This advantage grows with the length of the class sequence, since on CIFAR-100 and TinyImageNet the bounded buffer is also more accurate than full accumulation (Sec. 5.2), in line with Theorem 1. We therefore adopt B=1000 as the default, which balances accuracy against the recurring server cost.

(a) Target fraction			(b) Budget		
λ	AA	AIA	B	AA	AIA
0.25	66.92	73.15	500	61.11	69.12
0.50	67.16	73.35	**1000**	67.96	73.53
0.75	**67.96**	**73.53**	1500	69.43	74.07
1.00	66.70	72.86	2000	68.36	73.86

Table 9: Temporal herding sweeps on CIFAR-10 (β=0.5). (a) varies λ at B=1000, and (b) varies B at λ=0.75. Bold marks the default.

D.5 Ablation of Imbalance Resolution

The server's training pool is skewed toward the current task, since every earlier class is capped at B surrogates while the current task receives new surrogates in every round (Sec. 3.4). ReSCENE counters this skew with balanced sampling and logit adjustment, and we measure how much each of them contributes.

Setting. On CIFAR-10 at β=0.5, we switch each correction on and off through its coefficient. The sampling exponent α sets how often each class enters a batch, from count-proportional sampling at α=1, which follows the skewed pool, to square-root sampling at α=0.5. The logit-adjustment strength τ sets how strongly the classifier is corrected by the pool prior, from no correction at τ=0 to the standard adjustment at τ=1. Natural training (α=1, τ=0) uses neither correction, and ReSCENE (α=0.5, τ=1) uses both.

Variant	α	τ	Accuracy AA	Accuracy AIA
Natural	1.0	0.0	67.26	72.55
Sampling only	0.5	0.0	67.64	72.65
Logit only	1.0	1.0	67.70	73.26
ReSCENE	0.5	1.0	**67.96**	**73.53**

Table 10: Ablation of class-imbalance resolution on CIFAR-10 (β=0.5).

Results. Natural training already reaches 67.26 AA and 72.55 AIA (Table 10), so most of ReSCENE's accuracy comes from the server-side replay itself. Each correction alone improves on it, balanced sampling by 0.38 AA and 0.10 AIA points and logit adjustment by 0.44 AA and 0.71 AIA points. The two act on different parts of training, sampling on the data that enters each batch and logit adjustment on the bias of the classifier, and applying both gives the highest AA and AIA, 0.70 and 0.98 points above natural training. Combining the two corrections therefore resolves the class imbalance of the server pool better than either of them alone.

E System Cost Analysis

In weight-communicating methods, each client trains the model on its local data and uploads it, and the server only aggregates the updates. In ReSCENE, the clients only condense surrogates, and most of the computation takes place at the server. We therefore analyze its computation and communication cost in more depth. Sec. E.1 analyzes the cost on the client side, and Sec. E.2 analyzes the total cost of the whole federated system, including the server.

E.1 Client-Side Resource Accounting

ReSCENE optimizes and communicates surrogates in place of training and communicating the model, so its clients should spend less computation and less bandwidth than under any weight-communicating method. We measure this directly, accounting for every client-side resource, namely computation, persistent state, uplink, and downlink, for all eight methods on the three benchmarks.

Dataset	Method	Client comp. (PFLOP/run)	State mem. (MB)	Uplink (MB/rd)	Downlink (MB/rd)
CIFAR-10	FedAvg	32.68 (×1.0)	44.73	44.73	44.73
	FedEWC	34.31 (×1.0)	134.13	44.73	44.73
	FedLwF	41.32 (×1.3)	89.47	44.73	44.73
	TARGET	72.20 (×2.2)	359.31	44.73	359.31
	GLFC	48.43 (×1.5)	91.93	44.73	44.73
	FedCBDR	72.71 (×2.2)	63.17	44.73	44.73
	Re-Fed	71.70 (×2.2)	57.02	44.73	44.73
	ReSCENE	3.59 (×0.1)	44.73	0.25	44.73
CIFAR-100	FedAvg	33.24 (×1.0)	44.92	44.92	44.92
	FedEWC	34.90 (×1.0)	134.68	44.92	44.92
	FedLwF	43.20 (×1.3)	89.84	44.92	44.92
	TARGET	84.30 (×2.5)	359.49	44.92	359.49
	GLFC	203.18 (×6.1)	114.41	44.92	44.92
	FedCBDR	933.15 (×28.1)	229.24	44.92	44.92
	Re-Fed	146.46 (×4.4)	57.21	44.92	44.92
	ReSCENE	27.49 (×0.8)	44.92	1.23	44.92
TinyImageNet	FedAvg	67.21 (×1.0)	45.12	45.12	45.12
	FedEWC	70.58 (×1.1)	135.30	45.12	45.12
	FedLwF	87.38 (×1.3)	90.25	45.12	45.12
	TARGET	158.60 (×2.4)	1303.42	45.12	1303.42
	GLFC	410.42 (×6.1)	286.86	45.12	45.12
	FedCBDR	1884.40 (×28.0)	1519.68	45.12	45.12
	Re-Fed	295.96 (×4.4)	143.43	45.12	45.12
	ReSCENE	56.20 (×0.8)	45.12	9.83	45.12

Table 11: Client-side resources with ResNet-18. Multipliers are relative to FedAvg on the same dataset.

Setting. Table 11 covers one complete run of the Table 1 protocol, 100 rounds on CIFAR-10 and 200 on CIFAR-100 and TinyImageNet with 10 participants per round, and counts compute as floating-point operations with backward passes at twice the cost of forward passes. Client compute is the cost of all client-side work over the run, summed over the participating clients. State memory is the persistent client state between rounds, the model together with any auxiliary model or replay data, and excludes transient autograd and optimizer workspace. Uplink and downlink measure one message per round in each direction.

Analysis. ReSCENE uses the least client computation and uplink on every benchmark and keeps the smallest client state, tied with FedAvg (Table 11). Its client computation is $0.1\times$ that of FedAvg on CIFAR-10 and $0.8\times$ on CIFAR-100 and TinyImageNet, whereas every replay baseline exceeds FedAvg, since its local training set grows with the replay data of every seen class. FedCBDR, which keeps 150 samples per class, reaches $28.0\times$ the client cost of FedAvg on TinyImageNet, and GLFC and Re-Fed reach $6.1\times$ and $4.4\times$. The clients of ReSCENE store only the model, as in FedAvg, whereas every continual-learning baseline also stores an auxiliary model or replay data, up to 1,519.68 MB for FedCBDR on TinyImageNet. Its uplink is 0.25 MB on CIFAR-10 and 9.83 MB on TinyImageNet against the 44.73 to 45.12 MB model update of every other method, while its downlink equals that of FedAvg, since the server broadcasts the global model. These savings follow from the design of ReSCENE, whose clients only condense a fixed number of surrogates per class through the frozen global model, keep no replay data, and upload the surrogates in place of the model.

E.2 Total Computation across the Server and Clients

Cross-device federated learning pairs many resource-constrained clients with a resource-rich server (Kairouz et al., 2021). ReSCENE moves the training from these clients to the server and thus shifts the computational burden from the clients to the server. We therefore compare the total computation of the whole system, summing the server and all clients.

Method	Asymptotic compute (per task)		CIFAR-10 (per task)				
	All clients	Server	All clients (PFLOP)	Server (PFLOP)	Total (PFLOP)	Clients share	Server share
FedAvg	$O(RKED_{\text{loc}}F)$	$O(RKP)$	6.54	≈0	6.54	100%	0%
FedEWC	$O(RKED_{\text{loc}}F)$	$O(RKP)$	6.86	≈0	6.86	100%	0%
FedLwF	$O(RKED_{\text{loc}}F)$	$O(RKP)$	8.26	≈0	8.26	100%	0%
TARGET	$O(RKED_{\text{loc}}F)$	$O(R_{\text{syn}}S_{\text{kd}}B_{\text{syn}}F)$	14.44	38.2	52.64	27.4%	72.6%
GLFC	$O(RKE(D_{\text{loc}}+D_{\text{mem}})F)$	$O(RKP + IK\lvert\mathcal{C}_{t-1}\rvert F_{\text{enc}})$	9.69	<0.01	9.69	>99%	<1%
FedCBDR	$O(RKE(D_{\text{loc}}+D_{\text{mem}})F + ND_{\text{loc}}F)$	$O(RKP + ND_{\text{loc}}d^2)$	14.54	≈0	14.54	100%	0%
Re-Fed	$O(RKE(D_{\text{loc}}+D_{\text{mem}})F + NE_{\text{PIM}}D_{\text{pool}}F)$	$O(RKP)$	14.34	≈0	14.34	100%	0%
ReSCENE	$O(RKS\,\text{IPC}\,\lvert\mathcal{C}_t\rvert F)$	$O(RE_{\text{srv}}\lvert\mathcal{B}\cup S\rvert F)$	0.72	26.6	27.32	2.6%	97.4%

Table 12: Total computation across the server and all clients, asymptotically per task and on CIFAR-10 at task 2.

Setting. Table 12 sums computation over the server and all K participating clients, asymptotically and on CIFAR-10 at task 2 with one completed task's buffer, using E_{srv}=100 server epochs per round, R_{syn}=100 synthesis rounds, and the client figures of Table 11 divided over the five tasks. In the asymptotic columns, R is the rounds per task, N the total clients, E and E_{srv} the local and server epochs, E_{PIM} Re-Fed's personalized-informative-model epochs, D_{loc} the local images of a client, D_{mem} the replay images it stores (the exemplars of GLFC and FedCBDR and the cache of Re-Fed), D_{pool} the previous-sample pool Re-Fed scores at a task boundary, F the per-image forward-and-backward cost, F_{enc} the same for GLFC's LeNet-scale encode model, I its gradient-inversion steps per prototype, $|\mathcal{C}_{t-1}|$ the classes of the previous task, d the feature dimension of the leverage scores FedCBDR computes by SVD on the server, S_{kd} TARGET's distillation steps per synthesis round, B_{syn} its synthesis batch, P the model parameters, S the condensation steps, and $|\mathcal{B}\cup S|$ the buffer-and-pool size.

Analysis. The client computation of the weight-communicating methods grows with the local data D_{loc} of every participant, and with the replay data D_{mem} for GLFC, FedCBDR, and Re-Fed, whereas that of ReSCENE depends only on the condensation budget $\text{IPC}\times|\mathcal{C}_t|$. ReSCENE therefore gains the most when clients hold much local data or when a replay baseline stores a large memory. Its server term grows with E_{srv} and the size $|\mathcal{B}\cup S|$ of the buffer and pool, which grows with the participants only through the IPC surrogates per class that each of them uploads. Adding local data thus changes neither term of ReSCENE, and its server cost can be reduced through E_{srv}.

On CIFAR-10, the clients of ReSCENE spend 0.72 PFLOP per task, 11% of FedAvg's 6.54 PFLOP and 5% of Re-Fed's 14.34 PFLOP, and they carry only 2.6% of the total computation of ReSCENE. The remaining 97.4% runs on the server, which puts the resource-rich side of the system to use. The total of 27.32 PFLOP is about 4× that of FedAvg, but Sec. 5.3 shows that ReSCENE still leads every baseline when the server epochs are cut to a tenth, so this server share can be reduced substantially. TARGET also mitigates forgetting at the server, synthesizing data of earlier tasks by model inversion, but this costs it 38.2 PFLOP on the server, and its clients still spend 14.44 PFLOP. ReSCENE spends less on both sides, 26.6 against 38.2 PFLOP on the server and 0.72 against 14.44 PFLOP on the clients, and 27.32 against 52.64 PFLOP in total.

Round latency. In this work, we assume synchronous federated learning, in which the clients wait for the server to broadcast the next model. In weight-communicating methods, the server only averages the updates, so a round lasts as long as the slowest client takes to train locally and upload its model. In ReSCENE, the server trains the global model for E_{srv} epochs before the next broadcast, which adds a server phase to every round, whereas each client only condenses its surrogates and uploads up to 179× less data. ReSCENE thus moves the time of a round from the slowest client to the resource-rich server, and Sec. 5.3 shows that this server phase can be shortened tenfold while ReSCENE still leads every baseline.

F Privacy Analysis for ReSCENE

In ReSCENE, a client synthesizes its raw local data into condensed surrogates and transmits them to the server. We therefore analyze the privacy of the synthetic surrogates that ReSCENE communicates. We show in Sec. F.1 that the surrogate upload can be made differentially private by the Gaussian mechanism of FedDM (Xiong et al., 2023), and we measure in Sec. F.2 how much accuracy each method keeps under the same client-side gradient perturbation.

F.1 Differential Privacy of the Surrogate Upload

Differential privacy. A randomized mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private (Dwork & Roth, 2014) if, for any two datasets D, D' that differ in a single record and any set of outcomes O,

$$\Pr[\mathcal{M}(D) \in O] \;\leq\; e^{\epsilon}\, \Pr[\mathcal{M}(D') \in O] + \delta. \tag{14}$$

We consider instance-level privacy, where D and D' are the local data of one client that differ in a single image. One image can then change the distribution of what the client uploads by at most a factor e^{ϵ}, up to the slack δ, so an observer of the upload cannot tell whether that image was present.

DP-SGD. DP-SGD makes gradient descent differentially private by clipping the gradient of each example to norm C and adding Gaussian noise $\mathcal{N}(0, \sigma^2 C^2 \mathbf{I})$ to their sum at every step. Its privacy over many steps is guaranteed as follows.

Theorem 2 (Privacy of DP-SGD (Abadi et al., 2016))**.** *There exist constants c_1 and c_2 such that, given the sampling probability q and the number of steps T, for any $\epsilon < c_1 q^2 T$, DP-SGD is (ϵ, δ)-differentially private for any $\delta > 0$ if $\sigma \geq c_2\, q \sqrt{T \log(1/\delta)}/\epsilon$.*

Condensation as DP-SGD. Write the feature-matching term of the client loss for class c as $\mathcal{L}_c = \|\mu_c^{\mathrm{r}} - \mu_c^{\mathrm{s}}\|^2$, where $\mu_c^{\mathrm{r}} = \frac{1}{|\mathcal{R}|}\sum_{i\in\mathcal{R}} \phi(x_i)$ is the mean feature of a real minibatch $\mathcal{R}$ under the embedding ϕ of the frozen model and μ_c^{s} is the mean feature of the surrogate set $\mathcal{S}$. As FedDM shows, the gradient with respect to the surrogate pixels decomposes into one term per real image,

$$\nabla_{\mathcal{S}} \mathcal{L}_c = \frac{1}{|\mathcal{R}|} \sum_{i\in\mathcal{R}} \tilde{g}(x_i), \qquad \tilde{g}(x_i) = 2\left(\partial_{\mathcal{S}} \mu_c^{\mathrm{s}}\right)^{\top} \left(\mu_c^{\mathrm{s}} - \phi(x_i)\right), \tag{15}$$

and the logit-matching term decomposes in the same way. Each $\tilde{g}(x_i)$ depends on a single real image, so the optimization of the surrogates is an instance of DP-SGD, with the surrogates in place of the model parameters, and Theorem 2 applies to it.

Proposition 2 (Differential privacy of ReSCENE)**.** *Suppose that every client initializes its surrogates from random noise and optimizes them by DP-SGD, clipping each per-example term $\tilde{g}(x_i)$ of Eq. (15) to norm C and adding $\mathcal{N}(0, \sigma^2 C^2 \mathbf{I})$ to their sum. If σ satisfies the condition of Theorem 2, then, in each communication round, the uploads of all clients and the global model and buffers that the server derives from them are (ϵ, δ)-differentially private with respect to the local data of every client.*

Proof. By Theorem 2, the surrogates of each client are (ϵ, δ)-differentially private with respect to its local data, since the initialization from random noise does not depend on these data. The clients hold disjoint data, so by parallel composition (McSherry, 2009), the uploads of a round are (ϵ, δ)-differentially private for every client, as in FedDM. The server accumulates these uploads, compresses each task into a buffer by temporal herding, and trains the global model on the buffer and the pool without accessing the data of a client again. By post-processing (Dwork & Roth, 2014, Prop. 2.1), the buffers and the global model therefore satisfy the same (ϵ, δ).

Server-side reuse at no privacy cost. The forgetting mitigation of ReSCENE takes place entirely at the server after communication. The server stores the uploaded surrogates, compresses them into buffers, and replays these buffers in every later round, and none of these steps accesses the data of a client again. The global model that the server broadcasts is also derived from the surrogates alone, so by Proposition 2 the storage and replay raise no further privacy concern, and the privacy budget is spent once, at upload time.

F.2 Empirical Robustness to Gradient Perturbation

We measure how much accuracy each method keeps when its client-side computation is perturbed by the same Gaussian noise, following the evaluation of FedDM (Xiong et al., 2023).

Setting. On CIFAR-10 with β=0.5, we sweep the noise multiplier $\sigma \in \{0, 1, 3, 5\}$ with clip norm C=5. As in the experiments of FedDM, ReSCENE initializes the surrogates from sampled real images, clips the pixel gradient of each class to norm C, and adds Gaussian noise of scale σC divided by the size of the real minibatch, using class-wise real minibatches of up to 256 images. This protocol compares the accuracy of each method under the same perturbation rather than certifying a common (ϵ, δ). For the model-update baselines, we perturb local training in the same way, and for FedCBDR we additionally protect the pseudo-feature representations. Per-example gradients are undefined under BatchNorm, whose batch statistics couple the examples of a minibatch, so the model-update baselines use GroupNorm in place of BatchNorm at every σ, including σ=0. Their σ=0 entries therefore differ from Table 1, which uses BatchNorm. ReSCENE keeps the model of Table 1, since its perturbation acts on the surrogate pixels while the model stays frozen. Its σ=0 entry differs slightly from Table 1, since the privacy runs use class-wise real minibatches of up to 256 images.

	σ=0		σ=1		σ=3		σ=5	
Method	AA	AIA	AA	AIA	AA	AIA	AA	AIA
FedAvg	16.10	36.77	10.00	29.70	10.00	23.91	10.00	23.70
FedCBDR	34.12	56.03	30.45	46.85	10.90	31.70	9.83	27.42
Re-Fed	37.81	46.00	23.05	31.50	10.57	26.36	10.89	25.22
ReSCENE	**67.86**	**72.42**	**64.64**	**71.59**	**59.73**	**67.91**	**55.13**	**65.97**

Method	T0	T1	T2	T3	T4	AA
FedAvg	0.00	0.00	0.00	0.00	50.00	10.00
FedCBDR	0.00	0.00	0.00	0.00	49.15	9.83
Re-Fed	8.10	0.00	0.00	0.00	46.35	10.89
ReSCENE	**64.15**	**36.45**	**43.75**	**62.15**	**69.15**	**55.13**

Table 13: Privacy robustness on CIFAR-10 (β=0.5). Top, AA and AIA (%) at each noise multiplier σ. Bottom, final per-task accuracy (%) at σ=5.

Results. Accuracy decreases for every method as σ grows (Table 13), and the model-update baselines lose AA and AIA sharply. FedAvg falls to 10% AA from σ=1 onward, and FedCBDR and Re-Fed reach a similar value from σ=3. At σ=5, FedAvg and FedCBDR drop to 0% on the earlier tasks T0 to T3, Re-Fed keeps only 8.10% on T0, and all three keep about 50% only on the last task, so their AA of about 10% comes almost entirely from the last task. The perturbation thus removes the forgetting mitigation that these baselines are designed to provide.

ReSCENE keeps 55.13% AA at σ=5, 81.2% of its 67.86% at σ=0, and its accuracy stays spread over the earlier tasks. We attribute this to where the noise enters. In ReSCENE, the noise perturbs only the surrogates that a client uploads, and the server trains on them without adding further noise (Proposition 2). In the model-update baselines, the noise perturbs every local update that forms the global model, so it enters the model in every round.