FEDRUN: EFFICIENT CLIENT-SIDE FEDERATED UNLEARNING WITH RÉNYI ENTROPY

Anonymous authors
Paper under double-blind review

ABSTRACT

With the rise of privacy regulations, machine learning models should be able to remove the influence of designated data from the trained model upon request. In Federated Learning (FL), this task becomes more challenging because the training data remain distributed across multiple clients while only model updates are shared with the server. Retraining without target data provides a natural reference for unlearning, but requires substantial distributed computation and communication costs. Existing federated unlearning (FU) methods may help to reduce this cost, but most of them rely on stored training history, previous global models, or recovery procedures involving non-requesting clients, which incurs excessive runtime, massive historical storage, repeated communications, and intensive computation overhead. To address these limitations, we propose **FedRUn**, an efficient client-side FU framework based on Rényi entropy maximization. FedRUn performs unlearning directly from the final trained global model without requiring stored historical model updates or repeated communication and computation costs, reducing the system overhead and making unlearning very efficient. The unlearning process is performed locally at the target client, eliminating the need for the server to store historical client model updates while allowing each client to keep its data locally. Since the target client has direct access to its local data, FedRUn can apply the Rényi-entropy objective directly to either a subset of samples or the entire local dataset. Specifically, FedRUn maximizes predictive Rényi entropy on the forget set to reduce predictive confidence, while retaining and proximal terms preserve non-forgotten samples and limit model changes. Only the resulting client-computed unlearning update is sent to the server to update the global model. This design enables client-level, sample-level, and multi-client unlearning within the same framework while reducing computation, communication, and storage requirements.

1 INTRODUCTION

Federated Learning (FL) has become a practical paradigm for training machine learning models over distributed data McMahan et al. (2017); Kairouz & McMahan (2021). Instead of collecting raw data at a central server, FL allows clients to train locally and share only model updates for aggregation Karimireddy et al. (2020). However, even though raw data is not transmitted to the global server, once the global model is trained, the influence of local data may remain embedded in the final model. Regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) allow users to request deletion of their data Regulation (2016); Bonta (2022). In machine learning, satisfying such a request requires more than deleting the corresponding raw records because information learned from those records may remain encoded in the trained model Wang et al. (2019). Therefore, if a client later requests to forget all or part of its local data, the residual influence must be removed and evaluated. Shokri et al. (2017); Bagdasaryan et al. (2020)

Machine unlearning was proposed to reduce the influence of selected training data from a learned model Cao & Yang (2015); Bourtoule et al. (2021); Chen et al. (2021). Retraining from scratch provides a natural reference for unlearning, but repeating the complete training process can be computationally expensive Ginart et al. (2019); Neel et al. (2021). The problem becomes more challenging in FL, because the server does not store clients' raw data. Therefore, federated unlearning (FU) must

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.

remove a target client's influence while preserving the core FL assumption that raw data remains local Liu et al. (2024); Wu et al. (2022). Existing FU methods provide useful directions, but they still face important limitations. Full retraining without the target client provides a strong reference, but it repeats the entire FL process and incurs high computation and communication costs. To reduce the costs, FedEraser Liu et al. (2021) uses stored historical updates to approximate client-level retraining, but it requires intermediate global models or client updates and additional calibration. Other approaches use gradient-ascent-based local unlearning Halimi et al. (2022), recovery-based correction Zhang et al. (2023), and class-discriminative pruning Wang et al. (2022). Despite this progress, these methods still depend on stored historical states, repeated recovery rounds, or method-specific assumptions. This motivates the design of a lightweight post-training unlearning mechanism that can operate without reconstructing the original federated training process.

Therefore, we propose **FedRUn**, a client-side federated unlearning framework based on Rényi entropy maximization Rényi (1961). The main idea of FedRUn is to reduce the model's confidence on the forget set. A model that strongly remembers a sample often assigns high confidence to one class. FedRUn instead maximizes the Rényi entropy of the prediction distribution on forgotten data, encouraging the model to produce less confident predictions. By increasing prediction entropy on the forget set, FedRUn weakens residual membership evidence measured by confidence-based membership inference evaluation. To prevent uncontrolled model drift, FedRUn includes a proximal regularization term that keeps the local update close to the original global model. FedRUn operates directly on the final trained global model and performs unlearning locally at the target client, avoiding storage of previous global models and client updates to perform unlearning. Moreover, FedRUn can support sample-level, client-level and multi-client unlearning while preserving non-target data. The experimental results show that FedRUn substantially reduces computation and communication cost while effectively reducing target-data influence and preserving the utility. It further supports sample-level and multi-client unlearning while preserving non-target data.

The key contributions of this paper are summarized as follows: (i) **Client-side federated unlearning:** We introduce FedRUn, a post-training federated unlearning framework that performs unlearning locally at the target client and sends only the client-computed update to the server. This design avoids full retraining and storage of previous global models to perform unlearning. (ii) **Rényi-entropy-based forgetting:** We formulate a local unlearning objective that maximizes predictive Rényi entropy on the forget set, while retain and proximal terms preserve remaining data and limit model deviation. (iii) **Unified support for multiple unlearning requests:** FedRUn supports client-level, sample-level, and simultaneous multi-client unlearning within a single framework, without requiring request-specific unlearning methods.

2 Background and Preliminaries

2.1 Federated Learning and Federated Unlearning

Consider a federated learning system with K clients, where client k holds a private local dataset $\mathcal{D}_k$ with $n_k = |\mathcal{D}_k|$ samples. Following the standard FedAvg formulation (McMahan et al., 2017; Kairouz & McMahan, 2021; Romandini et al., 2024), the global objective is

$$\min_w F(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w), \qquad n = \sum_{k=1}^{K} n_k, \tag{1}$$

where

$$F_k(w) = \frac{1}{n_k} \sum_{(x_i, y_i) \in \mathcal{D}_k} \ell(w; x_i, y_i) \tag{2}$$

is the local empirical objective of client k. At communication round t, the server distributes w_t to participating clients, which locally optimize on their private data and return model updates for aggregation. After T rounds, the server obtains the trained global model w_T.

Machine unlearning seeks to reduce the influence of selected training data from a learned model (Cao & Yang, 2015; Bourtoule et al., 2021). Federated unlearning extends this problem to the federated setting Romandini et al. (2024), where the requested data remain distributed across clients (Liu et al., 2024; Wu et al., 2022). Retraining after excluding the requested data provides a natural reference, but

requires repeating federated optimization and communication. Existing FU methods reduce this cost through mechanisms such as historical-update reconstruction (Liu et al., 2021), client-side gradient-based unlearning with recovery (Halimi et al., 2022), gradient-conflict mitigation for improving post-unlearning utility recovery Pan et al. (2025), training-free history-based removal (Huynh et al., 2024), and communication-efficient unlearning procedures (Tao et al., 2024). These methods differ in the historical state, client participation, and post-request computation and communication they require. We formalize the post-training setting considered in this work in Section 3.

2.2 RÉNYI ENTROPY

Rényi entropy is a one-parameter family of information measures indexed by an order α (Rényi, 1961). For a discrete distribution $p = (p_1, \ldots, p_C)$ over C outcomes and $\alpha > 0$, $\alpha \neq 1$, it is defined as

$$H_\alpha(p) = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{C} p_i^\alpha \right). \tag{3}$$

As $\alpha \to 1$, H_α converges to Shannon entropy. For $\alpha > 1$, the power term p_i^α places greater emphasis on high-probability outcomes, making the measure sensitive to concentrated distributions. Rényi entropy has also been used in modern learning objectives for predictive uncertainty, uncertainty decomposition, and entropy-based regularization (Narimatsu et al., 2023; Zheng & Yang, 2024).

For classifier h_w, let $p_w(i \mid x)$ denote the probability assigned to class i and define $p_w(x) = (p_w(1 \mid x), \ldots, p_w(C \mid x))$. We use $H_\alpha(p_w(x))$ to denote the predictive Rényi entropy obtained by applying Eq. 3 to this predictive distribution. Higher entropy corresponds to a less confident predictive distribution, which motivates the Rényi-entropy forget objective introduced in Section 4.

3 FEDERATED UNLEARNING SYSTEM SETUP

We consider post-training federated unlearning after standard FL has completed T communication rounds and produced the trained global model w_T. An unlearning request arrives after training. The server retains w_T but does not access clients raw data, while the client holding the requested data remains available to participate. The goal is to construct an unlearned global model $\widetilde{w}$ that reduces the influence of the requested data while preserving the contributions associated with the data that should remain.

For a single-client request, let $k^\star$ denote the target client with local dataset $\mathcal{D}_{k^\star}$. We denote the requested samples by the forget set $\mathcal{D}_{\mathrm{fg}}$. For sample-level unlearning, the remaining target-client samples form the retain set $\mathcal{D}_{\mathrm{rt}}$. We consider three request granularities. For *client-level unlearning*, $\mathcal{D}_{\mathrm{fg}} = \mathcal{D}_{k^\star}$ and $\mathcal{D}_{\mathrm{rt}} = \emptyset$. For *sample-level unlearning*, $\mathcal{D}_{\mathrm{fg}} \subset \mathcal{D}_{k^\star}$ and $\mathcal{D}_{\mathrm{rt}} = \mathcal{D}_{k^\star} \setminus \mathcal{D}_{\mathrm{fg}}$ requiring forgetting of the requested subset while preserving the remaining local samples. For *multi-client unlearning*, $\mathcal{F} \subseteq \{1, \ldots, K\}$ denotes the set of target clients. Each $k \in \mathcal{F}$ performs unlearning using its locally available requested data, and the resulting client-computed unlearning updates are combined at the server to construct a single $\widetilde{w}$.

For all unlearning settings, the server starts with the trained global model w_T, while the data requested for removal remain at the corresponding target clients. FedRUn uses these local data to compute an unlearning update at each target client, which is then sent to the server and applied to the global model. Section 4 describes this procedure in detail.

4 FEDRUN: CLIENT-SIDE FEDERATED UNLEARNING

FedRUn operates directly from the final trained global model w_T and performs the data-dependent unlearning optimization at the target client $k^\star$, where the requested data remain locally available. Figure 1 summarizes the complete workflow. Upon receiving an unlearning request, the server sends w_T to the target client, which constructs and locally optimizes the FedRUn objective for S steps. The client then forms the client-computed unlearning update $\Delta_{k^\star} = w_{k^\star}^{(S)} - w_T$ and returns only this update to the server. The server applies the update using the scaling parameter β to construct the unlearned model $\widetilde{w}$, the same procedure extends to multiple target clients by aggregating their independently computed updates.

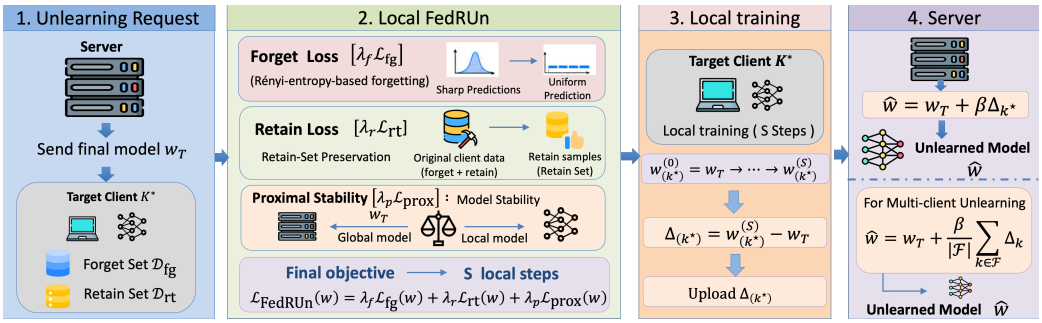

Figure 1: **FedRUn client-side federated unlearning workflow.** The target client performs local unlearning from the trained global model and returns the client-computed update to the server. The server constructs the final unlearned global model, while multiple target-client updates are aggregated for simultaneous unlearning.

4.1 Client-Side Unlearning Objective

The local FedRUn objective is designed to balance three requirements, reducing the model's confidence on the forget set, preserving useful behavior on retained target-client samples when they are available, and limiting unnecessary deviation from the trained global model. We therefore define

$$\mathcal{L}_{\text{FedRUn}}(w) = \lambda_f \mathcal{L}_{\text{fg}}(w) + \lambda_r \mathcal{L}_{\text{rt}}(w) + \lambda_p \mathcal{L}_{\text{prox}}(w), \tag{4}$$

where λ_f, λ_r, and λ_p control the contributions of the forget, retain, and proximal terms, respectively.

Rényi-entropy forget loss. The forget term is constructed to increase predictive uncertainty on $\mathcal{D}_{\text{fg}}$. For an input x, Section 2.2 defines the predictive Rényi entropy $H_\alpha(p_w(x))$. FedRUn uses its negative average over the forget set:

$$\mathcal{L}_{\text{fg}}(w) = -\frac{1}{|\mathcal{D}_{\text{fg}}|} \sum_{x \in \mathcal{D}_{\text{fg}}} H_\alpha(p_w(x)). \tag{5}$$

Because the overall objective in Eq. 4 is minimized, minimizing $\mathcal{L}_{\text{fg}}$ is equivalent to maximizing predictive Rényi entropy on the forget set. This discourages sharply peaked predictive distributions on the requested samples and reduces dominance by high-probability classes. Importantly, this entropy increase is used as the optimization mechanism for FedRUn by itself. It is not treated as evidence that unlearning has been achieved.

Retain loss. For sample-level unlearning, only a subset of $\mathcal{D}_{k^\star}$ belongs to the forget set, while the remaining samples form $\mathcal{D}_{\text{rt}}$. To preserve predictive behavior on these samples, FedRUn minimizes the standard cross-entropy loss

$$\mathcal{L}_{\text{rt}}(w) = -\frac{1}{|\mathcal{D}_{\text{rt}}|} \sum_{(x,y) \in \mathcal{D}_{\text{rt}}} \log p_w(y \mid x). \tag{6}$$

For client-level unlearning, $\mathcal{D}_{\text{rt}} = \emptyset$, and we set $\lambda_r = 0$. Thus, the retain term is activated only when samples belonging to the target client must remain represented after unlearning.

Proximal loss. Optimizing only the forget objective can move the local model farther from the trained solution than necessary. FedRUn therefore anchors the local optimization to the trained global model w_T using the mean squared parameter deviation

$$\mathcal{L}_{\text{prox}}(w) = \frac{1}{d} \|w - w_T\|_2^2, \tag{7}$$

where d is the number of trainable model parameters. This term penalizes excessive local parameter drift and encourages the unlearning procedure to remain close to the trained global model when $\lambda_p > 0$.

4.2 Local Unlearning Optimization and Client Update

After receiving the trained global model w_T, the target client $k^\star$ initializes its local model as

$$w_{k^\star}^{(0)} = w_T. \tag{8}$$

The target client then minimizes the FedRUn objective in Eq. 4 for S local unlearning steps using AdamW with learning rate η. The optimization is performed entirely at the target client using the locally available forget set $\mathcal{D}_{\mathrm{fg}}$ and, when applicable, the retain set $\mathcal{D}_{\mathrm{rt}}$.

After S local unlearning steps, the target client computes the difference between its locally unlearned model and the trained global model:

$$\Delta_{k^\star} = w_{k^\star}^{(S)} - w_T. \tag{9}$$

We refer to $\Delta_{k^\star}$ as the *client-computed unlearning update*. Only $\Delta_{k^\star}$ is returned to the server, while $\mathcal{D}_{\mathrm{fg}}$ and $\mathcal{D}_{\mathrm{rt}}$ remain local to the target client.

4.3 Server-Side Model Construction

After receiving the client-computed unlearning update $\Delta_{k^\star}$, the server applies it to the trained global model w_T to construct the final unlearned global model:

$$\widetilde{w} = w_T + \beta \Delta_{k^\star}, \tag{10}$$

where β is a server-side scaling parameter that controls the magnitude of the applied unlearning update.

The server does not perform the local unlearning optimization or compute $\Delta_{k^\star}$. These operations are carried out by the target client. Instead, the server is responsible only for applying the received update to the trained global model. Substituting Eq. 9 into Eq. 10 gives

$$\widetilde{w} = w_T + \beta \left(w_{k^\star}^{(S)} - w_T \right), \tag{11}$$

which makes explicit that the final model is obtained by moving the trained global model in the direction produced by the target client's local unlearning procedure.

This separation preserves the client–server roles of the federated setting, the data-dependent unlearning optimization remains at the target client, while the server constructs the final unlearned global model from the returned update.

4.4 Multi-Client Unlearning

FedRUn extends naturally to simultaneous unlearning requests involving multiple target clients. Let $\mathcal{F} \subseteq \{1, \ldots, K\}$ denote the set of target clients. Starting from the same trained global model w_T, each $k \in \mathcal{F}$ independently performs the client-side unlearning procedure using its locally available forget set and, when applicable, its retain set. After S local unlearning steps, client k computes its own client-computed unlearning update Δ_k and sends only this update to the server.

The server combines the independently computed target-client updates by averaging them and applies the resulting aggregate to w_T:

$$\widetilde{w} = w_T + \beta \frac{1}{|\mathcal{F}|} \sum_{k \in \mathcal{F}} \Delta_k. \tag{12}$$

Here, β retains the same role as in the single-client setting, controlling the magnitude of the aggregate unlearning update applied by the server.

This formulation preserves the client-side nature of FedRUn: each target client performs its own data-dependent unlearning optimization locally, while the server only aggregates the returned updates and constructs the final unlearned global model $\widetilde{w}$. Consequently, simultaneous requests from multiple clients can be handled without transferring their forget or retain data to the server.

5 ANALYSIS

We analyze the predictive behavior induced by the Rényi-entropy forget objective. The result provides a mechanism-level justification for reducing predictive confidence on the forget set. It does not constitute an exact-unlearning guarantee. The complete proof is provided in Appendix A.

5.1 RÉNYI-ENTROPY FORGETTING BEHAVIOR

The forget loss in Eq. 5 maximizes predictive Rényi entropy on $\mathcal{D}_{\mathrm{fg}}$. The following result relates this objective to the dominant predictive probability.

Lemma 1. *For a categorical distribution $p = (p_1, \ldots, p_C)$ and $\alpha > 1$,*

$$0 \le H_\alpha(p) \le \log C, \qquad p_{\max} \le \exp\left(-\frac{\alpha-1}{\alpha} H_\alpha(p)\right), \tag{13}$$

where $p_{\max} = \max_i p_i$. The upper entropy bound $\log C$ is attained by the uniform distribution.

Lemma 1 shows that increasing predictive Rényi entropy decreases an upper bound on the dominant class probability. Thus, minimizing the negative-entropy forget loss encourages predictions on $\mathcal{D}_{\mathrm{fg}}$ to become less confident. This result justifies the direction of the FedRUn forget objective. Empirical FU effectiveness is evaluated separately through model utility and membership inference in Sec. 6.

6 EXPERIMENTAL SETUP AND ANALYSIS

We evaluate FedRUn on MNIST, CIFAR-10 and Purchase-10 under FL settings with both IID and non-IID client partitions. All methods start from the same trained global model for each dataset and seed. We evaluate unlearning effectiveness using test accuracy, forget-set accuracy, retain-set accuracy when applicable, and membership-inference attack (MIA) metrics, while system efficiency is measured using runtime, client/server computation, persistent storage, and post-request communication. Complete dataset, model, training, and implementation details are provided in Appendix B.

6.1 CLIENT-LEVEL UNLEARNING EFFECTIVENESS AND EFFICIENCY

We first evaluate whether FedRUn can perform client-level unlearning while preserving model utility and reducing membership-inference exposure. Table 1 compares FedRUn with Retrain, FedEraser, Fast-FedUL, and PGA on MNIST, CIFAR-10, and Purchase-10. For each dataset, the complete local dataset of the requesting client $k^\star$ is treated as $\mathcal{D}_{\mathrm{fg}}$.

Table 1 compares client-level unlearning effectiveness across MNIST, CIFAR-10, and Purchase-10 in terms of test accuracy, forget-set accuracy, and MIA F1. On MNIST, FedRUn achieves 0.9901 test accuracy, closely matching retraining (0.9906), with an MIA F1 of 0.5929. On CIFAR-10, FedRUn achieves 0.674 test accuracy, close to retraining (0.680) and substantially higher than FedEraser (0.376) and Fast-FedUL (0.430), while obtaining an MIA F1 of 0.579. PGA achieves 0.6917 test accuracy and an MIA F1 of 0.5798, but requires additional post-request participation from non-requesting clients. FedEraser similarly requires participation from remaining clients for calibration and relies on stored training history. In contrast, FedRUn performs the data-dependent unlearning optimization only at the requesting client. On Purchase-10, FedRUn obtains an MIA F1 of 0.6775, while its test accuracy decreases to 0.9089 compared with 0.9603 for retraining, indicating a larger utility gap on this dataset. Additional experiments on CIFAR-100 show similar behavior in a more fine-grained classification setting, with FedRUn achieving 0.500 ± 0.002 test accuracy and 0.600 ± 0.004 MIA F1 while requiring only 1.41 ± 0.76 s and 86 MB of post-request communication. Complete results are reported in Appendix B.7. Overall, FedRUn preserves utility close to retraining on MNIST and CIFAR-10 while avoiding additional non-requesting-client participation during unlearning.

System cost. Table 2 compares the post-request system overhead of each method in terms of runtime, computation, storage, and communication costs. *Time* reports the measured wall-clock time required to complete the unlearning.

Table 1: **Client-level unlearning effectiveness.** Forget Acc. denotes the classification accuracy of the resulting global model on the requested forget set. **Bold** indicates the best non-retraining result: highest Test Acc. and the value closest to Retrain for Forget Acc. and MIA F1.

Dataset	Method	Test Acc.	Forget Acc.	MIA F1
MNIST	Retrain	0.9906 ± 0.0010	0.9902 ± 0.0016	0.5828 ± 0.0030
	FedEraser	0.9890 ± 0.0003	$\mathbf{0.9878 \pm 0.0016}$	$\mathbf{0.5927 \pm 0.0031}$
	Fast-FedUL	0.9872 ± 0.0014	0.9809 ± 0.0018	0.5996 ± 0.0041
	PGA	0.9870 ± 0.0040	0.9861 ± 0.0052	0.5088 ± 0.1572
	FedRUn	$\mathbf{0.9901 \pm 0.0018}$	0.9965 ± 0.0015	0.5929 ± 0.0080
CIFAR-10	Retrain	0.680 ± 0.012	0.686 ± 0.012	0.515 ± 0.095
	FedEraser	0.376 ± 0.026	0.374 ± 0.022	$\mathbf{0.502 \pm 0.131}$
	Fast-FedUL	0.430 ± 0.027	0.416 ± 0.037	0.563 ± 0.003
	PGA	$\mathbf{0.6917 \pm 0.0216}$	0.7042 ± 0.0267	0.5798 ± 0.0040
	FedRUn	0.674 ± 0.032	$\mathbf{0.698 \pm 0.035}$	0.579 ± 0.006
Purchase-10	Retrain	0.9603 ± 0.0003	0.9608 ± 0.0013	0.6654 ± 0.0018
	FedEraser	0.9530 ± 0.0038	0.9807 ± 0.0024	$\mathbf{0.6731 \pm 0.0031}$
	Fast-FedUL	0.9531 ± 0.0002	0.9873 ± 0.0046	0.6807 ± 0.0067
	PGA	$\mathbf{0.9630 \pm 0.0005}$	0.9996 ± 0.0003	0.7208 ± 0.0035
	FedRUn	0.9089 ± 0.0310	$\mathbf{0.9543 \pm 0.0351}$	0.6775 ± 0.0244

Table 2: **Client-level unlearning system cost.** Post-request runtime, computation, client-data access, server storage, and communication costs. Time reports mean runtime over three seeds. **Bold** indicates the lowest reported cost in each column.

Dataset	Method	Time (s)	Unlearning Computation		Storage Cost		Comm. (MB)
			Client	Server	Client (MB)	Server	
MNIST	Retrain	127.99	3.99 TFLOPs	17.2 MFLOPs	75.46	–	110.35
	FedEraser	28.30	997 GFLOPs	20.7 MFLOPs	75.46	72.55 MB	55.18
	Fast-FedUL	**0.04**	–	11.2 MFLOPs	**0**	70.69 MB	**0**
	PGA	40.9	1.70 TFLOPs	21.6 MFLOPs	94.32	**1.72** MB	141.39
	FedRUn	0.26	**6.65** GFLOPs	**0.862** MFLOPs	18.86	**1.72** MB	3.45
CIFAR-10	Retrain	874.6	1337 TFLOPs	1.12 GFLOPs	245.92	–	7157.47
	FedEraser	64.6	100 TFLOPs	536 MFLOPs	245.92	4.56 GB	1431.52
	Fast-FedUL	**0.30**	–	693 MFLOPs	**0**	4.51 GB	**0**
	PGA	81.9	342.68 TFLOPs	558.7 MFLOPs	307.40	**44.70** MB	3668.22
	FedRUn	1.58	**856** GFLOPs	**22.3** MFLOPs	61.48	**44.70** MB	89.44
Purchase-10	Retrain	248.1	32.6 TFLOPs	191 MFLOPs	341.60	–	1222.64
	FedEraser	19.5	2.72 TFLOPs	91.7 MFLOPs	341.60	774.75 MB	244.32
	Fast-FedUL	**0.026**	–	117 MFLOPs	**0**	463.32 MB	**0**
	PGA	23.8	5.57 TFLOPs	63.7 MFLOPs	427.00	**5.10** MB	417.91
	FedRUn	0.113	**5.88** GFLOPs	**2.55** MFLOPs	85.40	**5.10** MB	10.17

Computation measures the numerical work performed during unlearning and is reported in Floating-point operations (FLOPs), where MFLOPs, GFLOPs, and TFLOPs denote 10^6, 10^9, and 10^{12} floating-point operations, respectively. We separately report client- and server-side computation. Client computation includes local forward/backward passes and parameter updates, while server computation includes reconstruction, aggregation, and calibration.

Server and client storage Server storage measures the model information that must be retained before an unlearning request. Depending on the method, this may include previously saved global models, intermediate model checkpoints, historical training updates, and the final trained global model w_T. Retrain requires no previously stored model state, PGA and FedRUn require only w_T, while FedEraser and Fast-FedUL require additional historical information from the original FL process.

Client storage reports the amount of locally stored training data that must be available and accessed during unlearning. Retraining and FedEraser require the remaining client's data. PGA requires both the target client's data for the local unlearning step and the remaining clients' data for the subsequent recovery stage, which results in a larger client-data footprint than Retrain. Fast-FedUL requires no post-request client-data access, whereas FedRUn requires only the target client's requested data.

Communication measures the total amount of data transmitted between the server and clients after the unlearning request. This includes global-model downloads from the server and client-model uploads to the server, summed over all post-request communication rounds. Thus, methods that require repeated client participation, retraining rounds, or calibration involve communication whenever model parameters are sent to the clients and updated parameters are returned to the server. In contrast, FedRUn requires only sending w_T to the requesting client and returning the client-computed unlearning update to the server.

FedRUn performs a short optimization at the requesting client and requires only a small server-side operation to apply the returned update. On CIFAR-10, FedRUn requires 856 GFLOPs of client computation and 22.3 MFLOPs of server computation, compared with 1337 TFLOPs and 1.12 GFLOPs for Retrain, and 342.68 TFLOPs and 558.7 MFLOPs for PGA. FedRUn also retains only the final global model w_T (44.70 MB) and requires 89.44 MB of post-request communication, whereas FedEraser and Fast-FedUL require several GB of stored historical states, and Retrain and PGA require substantially more communication. Similar reductions are observed on MNIST and Purchase-10.

6.2 Effect of Rényi-Entropy-Based Forgetting

We next examine how the Rényi-entropy forget objective changes the predictive distribution on $\mathcal{D}_{\mathrm{fg}}$. We use normalized predictive Rényi entropy $\widehat{H}_\alpha$ and maximum predictive confidence as *mechanism diagnostics*. These quantities characterize the distributional changes induced by FedRUn.

Dataset	$\widehat{H}_\alpha$ **Before**	$\widehat{H}_\alpha$ **After**	$\Delta\widehat{H}_\alpha$	$\Delta\mathrm{Conf}$
MNIST	2.3×10^{-3}	3.7×10^{-3}	$+1.4 \times 10^{-3}$	$+1.8 \times 10^{-3}$
CIFAR-10	2.825×10^{-1}	3.429×10^{-1}	$\mathbf{+6.03 \times 10^{-2}}$	$\mathbf{+5.31 \times 10^{-2}}$
Purchase-10	6.0×10^{-4}	1.29×10^{-2}	$+1.23 \times 10^{-2}$	$+1.42 \times 10^{-2}$

Table 3: **Effect of the Rényi-entropy forgetting objective.** FedRUn increases normalized predictive Rényi entropy and reduces maximum confidence on $\mathcal{D}_{\mathrm{fg}}$.

Table 3 shows the same directional behavior across all three datasets, normalized Rényi entropy increases while predictive confidence decreases after FedRUn. The effect is strongest on CIFAR-10, where entropy increases from 2.825×10^{-1} to 3.429×10^{-1}, while confidence decreases by 5.31×10^{-2}. Purchase-10 also shows an increase in entropy of 1.23×10^{-2} and a confidence reduction of 1.42×10^{-2}, despite initially having a high-confidence predictive distribution.

These diagnostics summarize predictions over the entire forget set. Moreover, $\widehat{H}_\alpha = H_\alpha / \log C$ places Rényi entropy on a normalized $[0, 1]$ scale. The consistent increase in entropy together with the reduction in confidence indicates that FedRUn makes predictions on the requested data less confident. Unlearning effectiveness itself is evaluated separately through the utility and MIA results in Sec. 6.1. Additional sensitivity analyses for the Rényi order α, forget-loss weight λ_f, server scaling parameter β, and local unlearning steps S are reported in Appendix B.9. We also report the effect of the retain-loss weight λ_r for sample-level unlearning.

6.3 Sample-Level and Multi-Client Unlearning

Sample-level unlearning. We evaluate the sample-level unlearning on CIFAR-10 dataset under three forget-set sizes, corresponding to 30%, 50%, and 70% of the target client's local data. The remaining samples form retain-set $\mathcal{D}_{\mathrm{rt}}$. FedRUn uses the retain-aware objective with $\lambda_r = 1$ and is compared against Retrain and PGA baselines. FedEraser and Fast-FedUL are omitted because they are designed for client-level unlearning.

As shown in Fig. 2, FedRUn maintains test accuracy of 0.571–0.572 and retain accuracy of 0.721–0.735 across all forget ratios, while reducing MIA F1 from approximately 0.84 for the original model to 0.64–0.65. Retraining achieves lower MIA F1, whereas FedRUn preserves slightly higher global and retain accuracy, illustrating the forgetting–utility trade-off. Additional results and retain-weight sensitivity are reported in Appendix B.4.

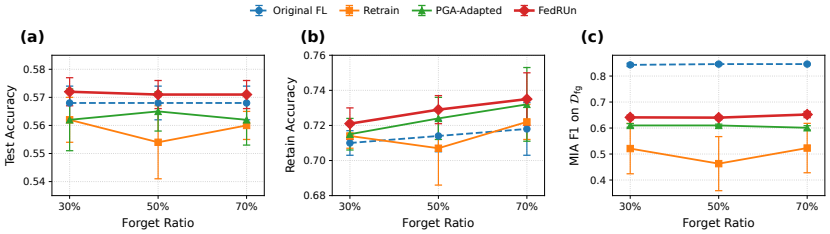

Figure 2: **Sample-level unlearning on CIFAR-10.** (a) Test accuracy, (b) retain accuracy, and (c) MIA F1 for 30%, 50%, and 70% forget ratios. FedRUn uses $\lambda_r = 1$

Multi-client unlearning. We next evaluate multi-client unlearning on CIFAR-10 dataset by varying the number of target clients as $|\mathcal{F}| \in \{1, 2, 3, 4\}$. Each target client independently computes its FU update from the same final global model w_T, and the server aggregates updates using Eq. 12.

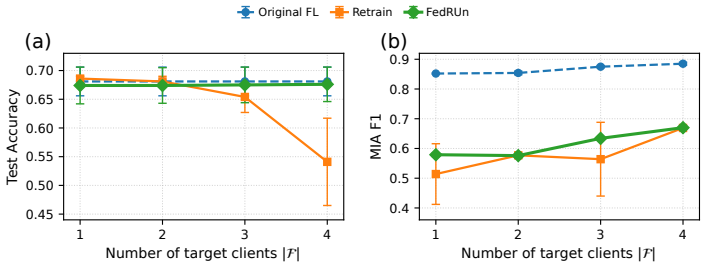

Figure 3: **Multi-client unlearning on CIFAR-10.** (a) Test accuracy and (b) MIA F1 as multiple clients unlearning $|\mathcal{F}|$ increases from 1 to 4.

FedRUn maintains test accuracy between 0.674 and 0.676 as $|\mathcal{F}|$ increases, while retraining decreases from 0.686 to 0.541. Its MIA F1 remains below the corresponding Original FL values, with only 0.31–1.13 s runtime and 85–213 MB post-request communications. Complete multi-client results are reported in Appendix B.5. We further evaluate class-level unlearning on CIFAR-10 by removing the *airplane* class across the federation and non-IID Dirichlet partitions with $\gamma_{\text{Dir}} \in \{0.1, 0.3, 0.5\}$. Complete results are reported in Appendices B.6 and B.8, respectively.

7 Conclusion

We introduced FedRUn, a client-side FU framework that performs post-training unlearning at the requesting client using a Rényi-entropy-based forget objective and returns only a client-computed update to the server. By operating directly from the final trained global model, FedRUn avoids storing historical training states and does not require non-requesting clients to participate during the FU procedure. The framework supports client-level, sample-level, and simultaneous multi-client unlearning, while retaining and proximal terms provide additional control when non-requested samples must be preserved or model deviation must be limited. Experiment results show that FedRUn substantially reduces post-request computation and communication compared with retraining and recovery-based baselines while maintaining useful model utility. Additional sample-level, multi-client, class-level, and non-IID evaluations further demonstrate its ability to operate across different request granularities and client distributions. In the future work, we will study FedRUn with larger models and datasets, asynchronous federated settings, and stronger empirical and formal verification of unlearning effectiveness.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the generation of synthetic data and to provide feedback on experimental design. We did not use generative AI tools to formulate or refine hypotheses, implement methods, translate content, clean or reformat datasets, or interpret experimental results. We did not use generative AI tools to formulate mathematical claims, provide critical ingredients for mathematical proofs, or assist in the writing of proofs. Qualitative or thematic data analysis is not applicable to this work.

Additionally, we used generative AI tools to improve the readability of the manuscript and assist with formatting. All AI-assisted data generation procedures and experimental design suggestions were reviewed and validated by the authors. All AI-assisted manuscript edits were also reviewed by the authors. We take responsibility for the final content of this work, including text, claims, data, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies federated machine unlearning and does not involve human participants or the collection of personally identifiable information. Experiments are conducted using standard benchmark datasets, including MNIST, FashionMNIST, CIFAR-10, and Purchase-10, under their respective usage conditions.

The goal of this work is to reduce the influence of requested training data from federated models while preserving model utility. Membership-inference attacks are used solely for evaluating residual membership information and are not applied to real users or real-world systems. All experiments are conducted in controlled research settings.

REFERENCES

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, pp. 2938–2948. PMLR, 2020.

Rob Bonta. California consumer privacy act (ccpa). *Retrieved from State of California Department of Justice: https://oag. ca. gov/privacy/ccpa*, 1, 2022.

Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pp. 141–159. IEEE, 2021.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pp. 463–480. IEEE, 2015.

Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pp. 896–911, 2021.

Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems*, 32, 2019.

Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. Federated unlearning: How to efficiently erase a client in fl? *arXiv preprint arXiv:2207.05521*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Thanh Trung Huynh, Trong Bang Nguyen, Phi Le Nguyen, Thanh Tam Nguyen, Matthias Weidlich, Quoc Viet Hung Nguyen, and Karl Aberer. Fast-fedul: A training-free federated unlearning with provable skew resilience. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 55–72. Springer, 2024.

Peter Kairouz and H Brendan McMahan. Advances and open problems in federated learning. *Foundations and trends in machine learning*, 14(1-2):1–210, 2021.

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

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th international symposium on quality of service (IWQOS)*, pp. 1–10. IEEE, 2021.

Ziyao Liu, Yu Jiang, Jiyuan Shen, Minyi Peng, Kwok-Yan Lam, Xingliang Yuan, and Xiaoning Liu. A survey on federated unlearning: Challenges, methods, and future directions. *ACM Computing Surveys*, 57(1):1–38, 2024.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. Pmlr, 2017.

Hiromi Narimatsu, Mayuko Ozawa, and Shiro Kumano. Collision probability matching loss for disentangling epistemic uncertainty from aleatoric uncertainty. In *International Conference on Artificial Intelligence and Statistics*, pp. 11355–11370. PMLR, 2023.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic learning theory*, pp. 931–962. PMLR, 2021.

Zibin Pan, Zhichao Wang, Chi Li, Kaiyan Zheng, Boqi Wang, Xiaoying Tang, and Junhua Zhao. Federated unlearning with gradient descent and conflict mitigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19804–19812, 2025.

Protection Regulation. Regulation (eu) 2016/679 of the european parliament and of the council. *Regulation (eu)*, 679(2016):10–13, 2016.

Alfréd Rényi. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*, volume 4, pp. 547–562. University of California Press, 1961.

Nicolò Romandini, Alessio Mora, Carlo Mazzocca, Rebecca Montanari, and Paolo Bellavista. Federated unlearning: A survey on methods, design guidelines, and evaluation metrics. *IEEE Transactions on Neural Networks and Learning Systems*, 36(7):11697–11717, 2024.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pp. 3–18. IEEE, 2017.

Youming Tao, Cheng-Long Wang, Miao Pan, Dongxiao Yu, Xiuzhen Cheng, and Di Wang. Communication efficient and provable federated unlearning. *arXiv preprint arXiv:2401.11018*, 2024.

Junxiao Wang, Song Guo, Xin Xie, and Heng Qi. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM web conference 2022*, pp. 622–632, 2022.

Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd international conference on artificial intelligence and statistics*, pp. 1226–1235. PMLR, 2019.

Leijie Wu, Song Guo, Junxiao Wang, Zicong Hong, Jie Zhang, and Yaohong Ding. Federated unlearning: Guarantee the right of clients to forget. *IEEE Network*, 36(5):129–135, 2022.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Lefeng Zhang, Tianqing Zhu, Haibin Zhang, Ping Xiong, and Wanlei Zhou. Fedrecovery: Differentially private machine unlearning for federated learning frameworks. *IEEE Transactions on Information Forensics and Security*, 18:4732–4746, 2023.

Kaixiang Zheng and En-Hui Yang. Knowledge distillation based on transformed teacher matching. In *International Conference on Learning Representations*, volume 2024, pp. 44963–44979, 2024.

A Algorithm and Theoretical Details

This appendix provides the complete FedRUn procedure and the proofs of the analytical results stated in Section 5.

A.1 Complete FedRUn Algorithm

Algorithm 1 summarizes the complete FedRUn procedure. For each target client, $\mathcal{D}_{\mathrm{fg}}$ and $\mathcal{D}_{\mathrm{rt}}$ denote the locally defined forget and retain sets associated with that client's unlearning request. When $\mathcal{D}_{\mathrm{rt}} = \emptyset$, as in client-level unlearning, the retain term is omitted.

For the single-client case, the generic notation above reduces to $w_{k^\star}^{(s)}$ and $\Delta_{k^\star}$ as defined in Section 4.2. For multi-client unlearning, each $k \in \mathcal{F}$ independently computes its update from the same trained global model w_T before the server combines the returned updates.

A.2 Proof of Lemma 1

We prove the properties stated in Lemma 1. The result follows from standard properties of Rényi entropy (Rényi, 1961).

Let $p = (p_1, \ldots, p_C)$ be a categorical probability distribution and let $\alpha > 1$. Recall that

$$H_\alpha(p) = \frac{1}{1-\alpha} \log \left(\sum_{i=1}^{C} p_i^\alpha \right). \tag{14}$$

Because $0 \le p_i \le 1$ and $\alpha > 1$, we have $p_i^\alpha \le p_i$. Therefore,

$$\sum_{i=1}^{C} p_i^\alpha \le \sum_{i=1}^{C} p_i = 1. \tag{15}$$

Since $1/(1-\alpha) < 0$, substituting this inequality into Eq. 14 gives

$$H_\alpha(p) \ge 0. \tag{16}$$

To establish the upper bound, note that x^α is convex for $\alpha > 1$. Jensen's inequality therefore gives

$$\frac{1}{C} \sum_{i=1}^{C} p_i^\alpha \ge \left(\frac{1}{C} \sum_{i=1}^{C} p_i \right)^\alpha. \tag{17}$$

Since $\sum_{i=1}^{C} p_i = 1$,

$$\sum_{i=1}^{C} p_i^\alpha \ge C^{1-\alpha}. \tag{18}$$

Using $1/(1-\alpha) < 0$ again yields

$$H_\alpha(p) \le \log C. \tag{19}$$

Equality is attained when

$$p_1 = \cdots = p_C = \frac{1}{C}, \tag{20}$$

i.e., when the predictive distribution is uniform.

We next relate Rényi entropy to the largest predicted class probability. Let

$$p_{\max} = \max_i p_i. \tag{21}$$

Because $p_{\max}^\alpha$ is one of the terms appearing in the sum,

$$\sum_{i=1}^{C} p_i^\alpha \ge p_{\max}^\alpha. \tag{22}$$

Algorithm 1 FedRUn: Client-Side Federated Unlearning

Require: Trained global model w_T; target-client set $\mathcal{F}$; local unlearning steps S; learning rate η; loss weights $\lambda_f, \lambda_r, \lambda_p$; Rényi order α; server-side scaling β

for all $k \in \mathcal{F}$ **do**
  Server sends w_T to target client k
  Initialize $w_k^{(0)} \leftarrow w_T$
  Initialize AdamW for $w_k^{(0)}$ with learning rate η
  for $s = 0, \ldots, S-1$ **do**
    Compute $\mathcal{L}_{\mathrm{fg}}\left(w_k^{(s)}\right)$ using Eq. 5
    if $\mathcal{D}_{\mathrm{rt}} \neq \emptyset$ **then**
      Compute $\mathcal{L}_{\mathrm{rt}}\left(w_k^{(s)}\right)$ using Eq. 6
    else
      Set $\mathcal{L}_{\mathrm{rt}}\left(w_k^{(s)}\right) \leftarrow 0$
    end if
    Compute $\mathcal{L}_{\mathrm{prox}}\left(w_k^{(s)}\right)$ using Eq. 7
    Compute $\mathcal{L}_{\mathrm{FedRUn}}\left(w_k^{(s)}\right)$ using Eq. 4
    Update $w_k^{(s+1)}$ using one AdamW step on $\mathcal{L}_{\mathrm{FedRUn}}\left(w_k^{(s)}\right)$
  end for
  Client computes $\Delta_k \leftarrow w_k^{(S)} - w_T$
  Client sends Δ_k to the server
end for
Construct

$$\widetilde{w} \leftarrow w_T + \beta \frac{1}{|\mathcal{F}|} \sum_{k \in \mathcal{F}} \Delta_k$$

return $\widetilde{w}$

Taking logarithms and multiplying by $1/(1-\alpha) < 0$ reverses the inequality, giving

$$H_\alpha(p) \leq -\frac{\alpha}{\alpha - 1} \log p_{\max}. \tag{23}$$

Rearranging,

$$p_{\max} \leq \exp\left(-\frac{\alpha - 1}{\alpha} H_\alpha(p)\right). \tag{24}$$

Therefore, increasing predictive Rényi entropy decreases an upper bound on the dominant class probability. This result provides the mathematical basis for using negative predictive Rényi entropy as the FedRUn forget loss. It characterizes the predictive behavior encouraged by the objective and does not, by itself, constitute a guarantee of exact unlearning. □

A.3 Proximal-Term Analysis

The proximal term in FedRUn is optional and limits excessive deviation of the locally optimized model from the trained global model w_T. Consistent with the implementation, we define

$$\mathcal{L}_{\mathrm{prox}}(w) = \frac{1}{d} \|w - w_T\|_2^2, \tag{25}$$

where d is the number of trainable scalar parameters.

Define

$$G(w) = \lambda_f \mathcal{L}_{\mathrm{fg}}(w) + \lambda_r \mathcal{L}_{\mathrm{rt}}(w). \tag{26}$$

Then the local FedRUn objective can be written as

$$\mathcal{L}_{\mathrm{FedRUn}}(w) = G(w) + \frac{\lambda_p}{d} \|w - w_T\|_2^2. \tag{27}$$

Proposition 1. *Let $w^\star$ be a stationary point of $\mathcal{L}_{\mathrm{FedRUn}}$ with $\lambda_p > 0$. Then*

$$w^\star - w_T = -\frac{d}{2\lambda_p}\nabla G(w^\star). \tag{28}$$

Consequently, if $\|\nabla G(w^\star)\|_2 \le B$, then

$$\|w^\star - w_T\|_2 \le \frac{dB}{2\lambda_p}. \tag{29}$$

Proof. At a stationary point $w^\star$,

$$\nabla \mathcal{L}_{\mathrm{FedRUn}}(w^\star) = 0. \tag{30}$$

Using Eq. 27,

$$\nabla G(w^\star) + \frac{2\lambda_p}{d}(w^\star - w_T) = 0. \tag{31}$$

Rearranging gives

$$w^\star - w_T = -\frac{d}{2\lambda_p}\nabla G(w^\star), \tag{32}$$

which proves Eq. 28. Taking the Euclidean norm on both sides and applying $\|\nabla G(w^\star)\|_2 \le B$ yields

$$\|w^\star - w_T\|_2 = \frac{d}{2\lambda_p}\|\nabla G(w^\star)\|_2 \le \frac{dB}{2\lambda_p}. \tag{33}$$

This result gives a direct interpretation of the optional proximal term: for fixed model dimension and forget-and-retain gradient magnitude, increasing λ_p strengthens the pull toward w_T and tightens the corresponding stationary-point deviation bound. The result characterizes only the regularizing effect of the proximal term. It does not establish retained-data preservation or an exact-unlearning guarantee. In the finalized client-level experiments, $\lambda_p = 0$, so this term is inactive. We retain the analysis because the general FedRUn formulation permits $\lambda_p > 0$ when additional control of local model deviation is desired.

B Experimental Details

Datasets and models. We evaluate FedRUn on MNIST LeCun et al. (1998), CIFAR-10 Krizhevsky & Hinton (2009) FashionMNIST Xiao et al. (2017) and Purchase-10 covering increasing task and model complexity. MNIST and FashionMNIST use the lightweight convolutional architectures adopted in our federated-learning setup, while CIFAR-10 uses ResNet-18 He et al. (2016), Purchase-10 uses a multilayer perceptron (MLP). For each dataset, standard federated training is first completed to obtain the final global model w_T, after which an unlearning request is issued. We evaluate three primary forms of unlearning: (i) client-level unlearning, where the complete local dataset $\mathcal{D}_{k^\star}$ is requested for forgetting. (ii) sample-level unlearning, where 30%, 50%, or 70% of the target client's samples form $\mathcal{D}_{\mathrm{fg}}$ and the remaining samples form $\mathcal{D}_{\mathrm{rt}}$ and (iii) multi-client unlearning with $|\mathcal{F}| \in \{1, 2, 3, 4\}$. We additionally evaluate class-level unlearning, CIFAR-100, and non-IID client distributions as complementary evaluations of request granularity, dataset complexity, and statistical heterogeneity. Dataset-specific federated-training and FedRUn configurations are reported in Appendix B.

Baselines. We compare FedRUn with four representative federated-unlearning methods. Retrain removes the requested data and repeats federated training from scratch, providing the reference unlearned model. FedEraser Liu et al. (2021) reconstructs the model using stored training history and performs post-request calibration with retained clients, requiring both persistent historical state and additional client participation. Fast-FedUL Huynh et al. (2024) removes the target client's estimated accumulated influence from stored historical updates and therefore avoids new client participation in the evaluated implementation at the cost of persistent history storage. PGA Halimi et al. (2022) performs constrained gradient-ascent unlearning at the requesting client followed by recovery involving non-requesting clients. Thus, FedEraser and PGA require other clients to remain available after an unlearning request, whereas FedRUn completes the request using only the requesting client and the server. All methods use the same data partition and target request within each dataset.

Evaluation protocol and metrics. Final results are reported as mean±standard deviation over three independent seeds, $\{43, 44, 45\}$. We evaluate *utility* using test accuracy and report accuracy on the requested forget set to characterize the model's behavior on the data targeted for unlearning. For sample-level requests, we additionally report retain-set accuracy to quantify preservation of $\mathcal{D}_{\mathrm{rt}}$. We assess residual membership information using membership-inference attacks (MIAs) Shokri et al. (2017), reporting attack precision, recall, and F1, with MIA F1 shown in the main tables.

B.1 Dataset, Model, and Training Configurations

All final results are reported over seeds $\{43, 44, 45\}$. For each seed, standard federated training is completed first to obtain the final global model w_T, and all unlearning methods evaluated under the same dataset and seed start from that finalized model. Table 4 summarizes the federated-training configurations used to obtain w_T.

Dataset	Model	Params.	K	Clients/ Round	T	Local Epochs	FL Batch Size
MNIST	Lightweight CNN	431,080	10	5	8	3	64
FashionMNIST	Lightweight CNN	431,080	10	5	8	3	64
CIFAR-10	ResNet-18	11,173,962	10	5	20	2	128
Purchase-10	Purchase-10 MLP	1,274,122	5	5	30	2	128

Table 4: **Dataset, model, and federated-training configurations.** K denotes the total number of clients and T the number of federated training rounds. Clients/Round gives the number of clients participating in each round.

Dataset	FL Optimizer	FL LR	Partition
MNIST	SGD	0.03	IID
FashionMNIST	SGD	0.03	IID
CIFAR-10	SGD	0.10	IID
Purchase-10	SGD	0.02	IID

Table 5: **Federated-training optimization settings.** SGD uses momentum 0.9 with Nesterov acceleration. Weight decay is 5×10^{-4} for MNIST, FashionMNIST, and CIFAR-10, and 1×10^{-4} for Purchase-10. The target client is $k^\star = 0$ in the finalized client-level experiments.

MNIST and FashionMNIST use the same lightweight convolutional architecture with 431,080 trainable parameters. CIFAR-10 uses a ResNet-18 architecture with the CIFAR-specific input stem and 11,173,962 trainable parameters, while Purchase-10 uses an MLP with 1,274,122 trainable parameters. All four finalized configurations use IID client partitions. CIFAR-10 reduces the FL learning rate by a factor of 0.1 at rounds 12 and 16, while Purchase-10 applies the same decay at rounds 20 and 25.

Dataset	Optimizer	η	α	λ_f	λ_r	λ_p	S	β
MNIST	AdamW	1×10^{-4}	2.0	0.1	0	0	15	0.75
FashionMNIST	AdamW	1×10^{-4}	2.0	0.1	0	0	15	0.75
CIFAR-10	AdamW	1×10^{-4}	1.5	0.005	0	0	8	0.10
Purchase-10	AdamW	5×10^{-4}	1.5	0.001	0	0	24	0.20

Table 6: **FedRUn hyperparameters for the finalized client-level experiments.** Here, η is the client-side unlearning learning rate, α is the Rényi order, λ_f, λ_r, and λ_p weight the forget, retain, and proximal terms, S is the number of local unlearning steps, and β controls server-side scaling of the client-computed unlearning update. For client-level requests, $\mathcal{D}_{\mathrm{rt}} = \emptyset$ and therefore $\lambda_r = 0$.

B.2 Baseline Operational Requirements

Method	Full Retrain	Hist. Storage	Unlearning Comm.	Sample-Level Unlearning
Retrain	Yes	No	High	Yes
FedEraser	No	Yes	High	No
Fast-FedUL	No	Yes	None	No
PGA	No	No	High	No
FedRUn	**No**	**No**	**Low**	**Yes**

Table 7: **Operational requirements and unlearning capabilities of federated unlearning methods.** Full Retrain indicates whether training must be repeated from scratch after excluding the requested data. Hist. Storage indicates storing the previous global models and training history during training. Unlearning Comm. summarizes the model/update communication required after the request, exact communication volumes are reported in Table 2. Sample-Level Unlearning indicates whether the method directly supports forgetting a subset of a client's data rather than removing the entire client. FedRUn requires neither full retraining nor persistent historical storage, incurs low post-request communication, and supports both client level and sample-level unlearning.

B.3 Post-Request System-Cost Accounting

We measure system cost only after an unlearning request is received and the trained global model w_T is already available, computation performed during the original federated training process is therefore excluded. We report wall-clock time, client- and server-side computation, persistent historical storage, and post-request communication.

Client-side computation. We estimate client-side computation from the actual local optimization performed after an unlearning request. For every local optimizer step, we count the computation required to process the corresponding mini-batch, including both the forward and backward passes through the model. The total client-side cost is then obtained by summing these operations over all mini-batches, local epochs, participating clients, and post-request rounds used by the method:

$$C_{\text{client}} = \sum_{u} \sum_{b \in \mathcal{B}_u} F_{\text{train}}(b), \tag{34}$$

where u indexes post-request client updates, $\mathcal{B}_u$ is the set of mini-batches processed for update u, and $F_{\text{train}}(b)$ is the FLOP cost of one forward–backward optimization step on mini-batch b.

To obtain $F_{\text{train}}(b)$ consistently across methods, we first measure the forward-pass FLOPs of each model and approximate the backward pass as twice the forward-pass cost. Thus, processing a mini-batch of size $|b|$ requires approximately three times the corresponding forward-pass computation. Importantly, each training example is counted only when it is actually processed, the dataset size is not multiplied again by the number of mini-batches.

For retraining, this cost includes all local training performed by the participating clients during the post-request retraining rounds. For FedEraser, it includes the client-side computation performed during its post-request recovery procedure. For PGA, we count both the target-client gradient-ascent computation and the subsequent recovery computation performed by participating clients. FedRUn performs computation only at the requesting client, and its cost is obtained directly from the S AdamW unlearning steps executed on $\mathcal{D}_{\text{fg}}$. Fast-FedUL performs no neural-network training at clients after the request in the evaluated implementation, so its post-request computational cost is accounted for on the server.

Server-side computation. We account for server computation separately from client-side model optimization. Since the server does not perform neural-network training in the evaluated methods, its cost consists of arithmetic operations on model parameters or stored client updates after the unlearning request. For retraining and PGA, we count the operations required to aggregate the models returned by the participating clients during each post-request round. FedEraser additionally performs its reconstruction and recovery operations using stored historical information, while Fast-FedUL performs its unlearning computation directly on previously stored client updates at the server.

For FedRUn, the server performs only the final update construction. After receiving the client-computed unlearning update $\Delta_{k^\star}$, it computes

$$\widetilde{w} = w_T + \beta \Delta_{k^\star}. \tag{35}$$

For a model with d parameters, this requires one multiplication and one addition per parameter, corresponding to approximately $2d$ floating-point operations. We sum the corresponding server-side operations over all post-request aggregation or update steps and report the resulting cost in GFLOPs.

Storage, communication, and runtime. Storage measures the persistent model information that must be retained before an unlearning request arrives. This includes the final trained global model w_T and, when required by a method, additional historical global models, intermediate checkpoints, or stored client updates. Retrain does not require previously retained model state specifically for the unlearning. PGA and FedRUn require only w_T, whereas FedEraser and Fast-FedUL require additional historical training information.

Communication is measured only after an unlearning request is received and includes all model or update transfers between the server and participating clients. FedRUn requires a single post-request exchange: the server sends w_T to the requesting client $k^\star$, and the client returns the client-computed update $\Delta_{k^\star}$. The remaining clients do not participate in the unlearning request. PGA, in contrast, performs a recovery stage after the target-client unlearning step, requiring additional communication with participating clients. Fast-FedUL performs its post-request operation from previously stored historical updates and therefore does not require new client communication in the evaluated implementation, trading communication cost for persistent storage.

Finally, wall-clock time is measured directly for the complete post-request procedure, whereas client- and server-side GFLOPs are analytical estimates of the computational work performed. We report both because runtime depends on the hardware and implementation, while FLOPs provide a hardware-independent measure of computational cost.

B.4 Additional Sample-Level Unlearning Results

For sample-level unlearning, only a subset of the target client's data is included in the forget set $\mathcal{D}_{\mathrm{fg}}$, while the remaining target-client samples form $\mathcal{D}_{\mathrm{rt}}$. We evaluate forget fractions of 30%, 50%, and 70%. Unlike the finalized client-level configuration, the sample-level setting activates the retain objective. For CIFAR-10, FedRUn uses $\alpha = 2.0$, $\lambda_f = 0.1$, $\lambda_r = 1.0$, $S = 15$, and $\beta = 0.75$.

We compare against retraining and PGA-Adapted. For retraining, only the requested samples are removed before training again. For PGA-Adapted, the gradient-ascent operation is restricted to the requested forget subset. FedEraser and Fast-FedUL are not included because the evaluated implementations operate at client-level granularity and do not provide a corresponding sample-level unlearning procedure.

Table 8: **Sample-level unlearning on CIFAR-10.** Results are mean±standard deviation over seeds $\{43, 44, 45\}$. $\mathcal{D}_{\mathrm{fg}}$ contains the requested samples and $\mathcal{D}_{\mathrm{rt}}$ contains the remaining target-client samples.

Forget Ratio	Method	Test Acc.	Forget Acc.	Retain Acc.	MIA F1
30%	Original FL	0.568 ± 0.006	0.709 ± 0.021	0.710 ± 0.007	0.843 ± 0.006
	Retrain	0.562 ± 0.008	0.572 ± 0.007	0.714 ± 0.007	0.521 ± 0.097
	PGA-Adapted	0.562 ± 0.011	0.682 ± 0.028	0.715 ± 0.009	0.610 ± 0.007
	FedRUn	$\mathbf{0.572 \pm 0.005}$	$\mathbf{0.714 \pm 0.024}$	$\mathbf{0.721 \pm 0.009}$	$\mathbf{0.641 \pm 0.009}$
50%	Original FL	0.568 ± 0.006	0.706 ± 0.012	0.714 ± 0.009	0.846 ± 0.005
	Retrain	0.554 ± 0.013	0.541 ± 0.013	0.707 ± 0.021	0.463 ± 0.104
	PGA-Adapted	0.565 ± 0.007	0.686 ± 0.014	0.724 ± 0.012	0.610 ± 0.005
	FedRUn	$\mathbf{0.571 \pm 0.005}$	$\mathbf{0.712 \pm 0.013}$	$\mathbf{0.729 \pm 0.008}$	$\mathbf{0.640 \pm 0.008}$
70%	Original FL	0.568 ± 0.006	0.707 ± 0.007	0.718 ± 0.015	0.846 ± 0.003
	Retrain	0.560 ± 0.005	0.564 ± 0.012	0.722 ± 0.010	0.523 ± 0.095
	PGA-Adapted	0.562 ± 0.009	0.686 ± 0.014	0.732 ± 0.021	0.601 ± 0.008
	FedRUn	$\mathbf{0.571 \pm 0.005}$	$\mathbf{0.711 \pm 0.007}$	$\mathbf{0.735 \pm 0.015}$	$\mathbf{0.652 \pm 0.012}$

Table 8 provides the complete numerical results corresponding to Fig. 2. Across all three forget ratios, FedRUn changes test accuracy by only $+0.003$–$+0.004$ relative to the original global model and increases retain accuracy by 0.011–0.017. At the same time, MIA F1 decreases by 0.194–0.206. Retraining produces a larger reduction in MIA F1, whereas FedRUn better preserves the original model's utility, illustrating the forgetting–preservation trade-off in sample-level requests.

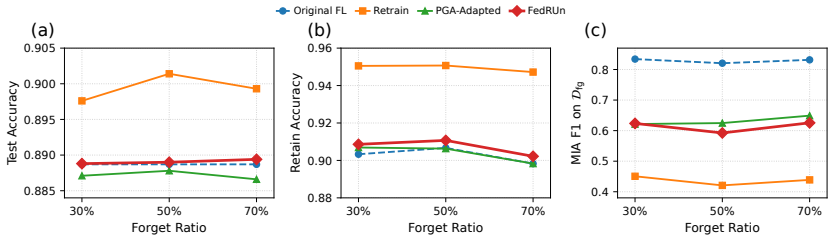

Figure 4: **Additional sample-level results on FashionMNIST.** Test accuracy, retain accuracy on $\mathcal{D}_{\mathrm{rt}}$, and MIA F1 on $\mathcal{D}_{\mathrm{fg}}$ for 30%, 50%, and 70% forget ratios. The trends provide an additional evaluation of the retain-aware FedRUn objective beyond the CIFAR-10 setting used in the main paper.

Figure 4 shows the corresponding experiment on FashionMNIST. FedRUn again maintains nearly unchanged global accuracy while preserving the non-requested target-client samples and reducing MIA F1 relative to the original model.

B.5 Additional Multi-Client Unlearning Results

Table 9: **Multi-client unlearning on CIFAR-10.** Results are mean $\pm$ standard deviation over seeds $\{43, 44, 45\}$. $\mathcal{F}$ denotes the set of clients simultaneously requesting unlearning.

$\lvert\mathcal{F}\rvert$	Method	Test Acc.	Forget Acc.	MIA F1	Time (s)	Comm. (MB)
1	Original FL	0.681 ± 0.025	0.705 ± 0.029	0.852 ± 0.002	0	0
	Retrain	0.686 ± 0.004	0.693 ± 0.008	0.514 ± 0.102	291 ± 1	6826
	FedRUn	$\mathbf{0.674 \pm 0.032}$	$\mathbf{0.698 \pm 0.035}$	$\mathbf{0.579 \pm 0.006}$	$\mathbf{0.31 \pm 0.01}$	**85**
2	Original FL	0.681 ± 0.025	0.706 ± 0.030	0.854 ± 0.006	0	0
	Retrain	0.681 ± 0.009	0.680 ± 0.015	0.577 ± 0.002	224 ± 5	5119
	FedRUn	$\mathbf{0.674 \pm 0.031}$	$\mathbf{0.699 \pm 0.037}$	$\mathbf{0.576 \pm 0.008}$	$\mathbf{0.59 \pm 0.02}$	**128**
3	Original FL	0.681 ± 0.025	0.706 ± 0.032	0.875 ± 0.004	0	0
	Retrain	0.654 ± 0.027	0.657 ± 0.035	0.564 ± 0.124	152 ± 1	3413
	FedRUn	$\mathbf{0.675 \pm 0.031}$	$\mathbf{0.700 \pm 0.037}$	$\mathbf{0.634 \pm 0.005}$	$\mathbf{0.85 \pm 0.01}$	**171**
4	Original FL	0.681 ± 0.025	0.705 ± 0.030	0.885 ± 0.005	0	0
	Retrain	0.541 ± 0.076	0.542 ± 0.081	0.669 ± 0.002	84 ± 2	1706
	FedRUn	$\mathbf{0.676 \pm 0.030}$	$\mathbf{0.700 \pm 0.033}$	$\mathbf{0.670 \pm 0.005}$	$\mathbf{1.13 \pm 0.02}$	**213**

B.6 Class-Level Unlearning Results

We additionally evaluate class-level unlearning on CIFAR-10 by requesting the removal of all samples belonging to the *airplane* class. Unlike client-level unlearning, where the complete local dataset of one client is forgotten, this setting requires removing the same semantic class across the federated training population. Each participating target client starts independently from the same trained global model w_T, performs the local unlearning procedure on its locally available airplane samples, and returns a client-computed unlearning update to the server. The server aggregates the returned updates using the multi-client construction in Section 4.4. Because the class is distributed across the participating CIFAR-10 clients, each client containing *airplane* samples is treated as a target client for this request.

FedRUn uses $\alpha = 1.1$. We compare against the original global model, full retraining after removing the target class, and PGA.

Table 10: **Class-level unlearning on CIFAR-10.** The *airplane* class is requested for forgetting across the federation. Forget-Class Acc. reports accuracy on airplane samples and Retain-Class Acc. reports accuracy on all remaining classes.

Method	Test Acc.	Forget Acc.	Retain Acc.	MIA P	MIA R	MIA F1
Original FL	0.6900	0.6850	0.6906	0.9331	0.6230	0.7467
Retrain	0.6601	0.0000	0.7334	0.5043	0.6860	0.5435
PGA	0.6615	0.0000	0.7350	0.5467	0.2237	0.3120
FedRUn	0.6430	0.1067	0.7026	0.5183	0.6417	0.5570

Table 10 shows that FedRUn substantially changes the model's behavior on the requested class while preserving useful performance on the remaining classes. Forget-class accuracy decreases from 0.6850 in the original global model to 0.1067 after FedRUn, while retain-class accuracy remains at 0.7026. FedRUn achieves 0.6430 overall test accuracy compared with 0.6601 for retraining. Its MIA F1 decreases from 0.7467 in the original model to 0.5570, which is in the same numerical range as retraining (0.5435). PGA obtains a lower MIA F1 but also drives forget-class accuracy to zero and requires additional post-request client participation.

Table 11: **System cost for class-level unlearning on CIFAR-10.** Post-request runtime, client/server computation, persistent storage, and communication.

Method	Runtime (s)	Client GFLOPs	Server GFLOPs	Storage	Comm. (MB)
Retrain	418.4 ± 103.5	$1{,}504{,}931 \pm 1{,}315$	1.3409 ± 0.0000	–	8,939.62
PGA	162.1 ± 0.7	$760{,}590 \pm 0$	1.2291 ± 0.0000	44.70 MB	8,939.62
FedRUn	16.5 ± 0.3	$32{,}095 \pm 0$	0.0223 ± 0.0000	44.70 MB	446.69

B.7 Additional Client-Level Results on CIFAR-100

Table 12 reports client-level unlearning results on CIFAR-100. FedRUn achieves 0.500 ± 0.002 test accuracy, compared with 0.489 ± 0.006 for Retrain and 0.512 ± 0.003 for PGA. Its MIA F1 is 0.600 ± 0.004, lower than PGA (0.616 ± 0.004) but higher than the Retrain reference (0.520 ± 0.142).

The system-cost difference is substantially larger. FedRUn completes the request in 1.41 ± 0.76 s with only 86 MB of post-request communication, compared with 621 ± 240 s and 8568 MB for Retrain, and 94 ± 4 s and 3513 MB for PGA. FedEraser and Fast-FedUL obtain lower MIA F1 values than FedRUn, but with substantially lower test accuracy and, in the case of Fast-FedUL, considerably higher communication. These results further illustrate the utility–unlearning–system-cost trade-off considered in FedRUn.

Table 12: Client-level unlearning on CIFAR-100.

Method	Test	MIA F1	Time	Comm.
Retrain	0.489	0.520	621	8568
FedEraser	0.163	0.438	42	5402
Fast-FedUL	0.411	0.497	0.40	10710
PGA	0.512	0.616	94	3513
FedRUn	0.500	0.600	1.41	86

B.8 Additional Non-IID Results

We evaluate client-level unlearning under heterogeneous client distributions generated using Dirichlet partitions with $\gamma_{\mathrm{Dir}} \in \{0.1, 0.3, 0.5\}$, where smaller γ_{Dir} corresponds to stronger statistical heterogeneity. All methods use the same client partition.

Figure 5 reports the CIFAR-10 results. FedRUn maintains test accuracy of 0.379, 0.556, and 0.608 as γ_{Dir} increases from 0.1 to 0.5, remaining close to the corresponding pre-unlearning model while

avoiding the larger utility degradation observed for FedEraser and Fast-FedUL. FedRUn also reduces MIA F1 relative to the original global model across all three heterogeneity levels, obtaining values between 0.505 and 0.521.

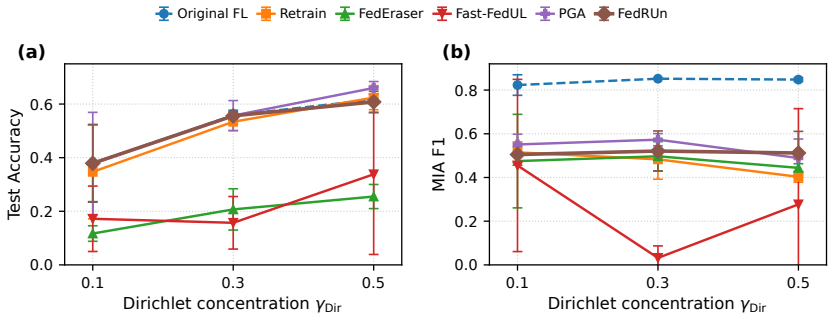

Figure 5: **Robustness to non-IID client distributions on CIFAR-10.** (a) Test accuracy and (b) MIA F1 under Dirichlet client partitions with $\gamma_{\text{Dir}} \in \{0.1, 0.3, 0.5\}$, where smaller γ_{Dir} corresponds to stronger statistical heterogeneity. Points report the mean over seeds $\{43, 44, 45\}$ and error bars indicate standard deviation. All methods use the same client partition. Original FL denotes the pre-unlearning global model.

We further evaluate the same non-IID setting on MNIST and FashionMNIST. Figures 6 and 7 show that FedRUn consistently reduces MIA F1 relative to the corresponding original global model across all evaluated values of γ_{Dir}. Test accuracy generally improves as γ_{Dir} increases, reflecting the reduced statistical heterogeneity, while FedRUn preserves useful global utility across both datasets.

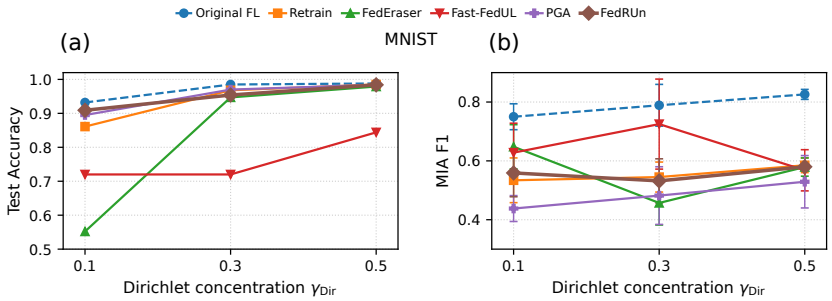

Figure 6: **Non-IID client-level unlearning on MNIST.** (a) Test accuracy and (b) MIA F1 under Dirichlet client partitions with $\gamma_{\text{Dir}} \in \{0.1, 0.3, 0.5\}$. Smaller γ_{Dir} corresponds to stronger statistical heterogeneity. Test-accuracy curves report mean values, while MIA F1 error bars denote standard deviation. Original FL denotes the pre-unlearning global model.

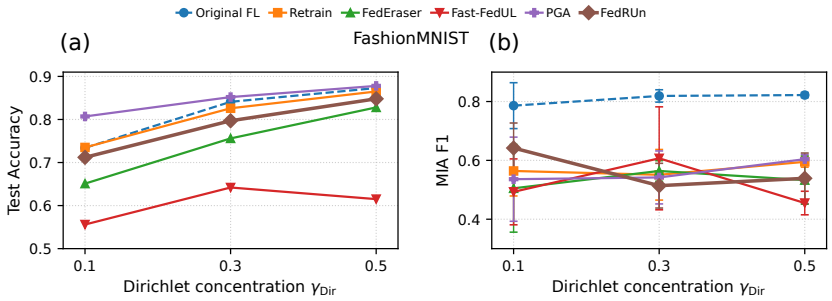

Figure 7: **Non-IID client-level unlearning on FashionMNIST.** (a) Test accuracy and (b) MIA F1 under Dirichlet client partitions with $\gamma_{\text{Dir}} \in \{0.1, 0.3, 0.5\}$. Smaller γ_{Dir} corresponds to stronger statistical heterogeneity. Test-accuracy curves report mean values, while MIA F1 error bars denote standard deviation. Original FL denotes the pre-unlearning global model.

Across both datasets, FedRUn lowers MIA F1 relative to Original FL at every evaluated heterogeneity level while maintaining substantially higher utility as the client distributions become less skewed.

B.9 Hyperparameter Sensitivity

We analyze the sensitivity of FedRUn to the Rényi order α, the forget-loss weight λ_f, the server scaling factor β, and the number of local unlearning steps S.

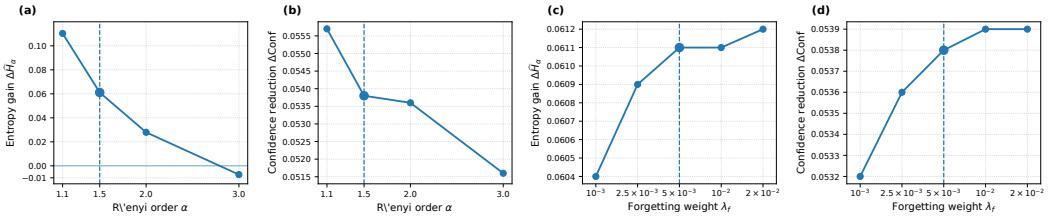

Figure 8: **Rényi-entropy sensitivity on CIFAR-10.** (a)–(b) Entropy gain and confidence reduction versus α. (c)–(d) the same diagnostics versus λ_f. Dashed lines mark the selected settings $\alpha = 1.5$ and $\lambda_f = 0.005$.

As shown in Fig. 8, the Rényi order α directly affects the forgetting mechanism. The normalized entropy gain decreases from 0.1103 at $\alpha = 1.1$ to 0.0611 at the selected setting $\alpha = 1.5$, becoming slightly negative at $\alpha = 3.0$, while confidence reduction changes more gradually from 0.0557 to 0.0516. These results indicate that α controls the strength of the Rényi-entropy response on the forget set.

In contrast, FedRUn is relatively insensitive to λ_f over the evaluated range. The normalized entropy gain remains between 0.0604 and 0.0612, while confidence reduction varies only from 0.0532 to 0.0539. This stability indicates that the forgetting mechanism remains consistent under moderate changes around the selected value $\lambda_f = 0.005$.

Retain-loss weight λ_r. We examine the effect of the retain-loss weight in sample-level unlearning for forget ratios of 30%, 50%, and 70%. As shown in Fig. 9, introducing retain supervision improves both global test accuracy and retain-set accuracy, with most of the gain occurring between $\lambda_r = 0$ and $\lambda_r = 1$. At the same time, MIA F1 on $\mathcal{D}_{\text{fg}}$ generally increases as λ_r becomes larger, revealing a trade-off between preserving non-requested samples and suppressing membership evidence on the forget set. We therefore use $\lambda_r = 1$ as the finalized sample-level setting.

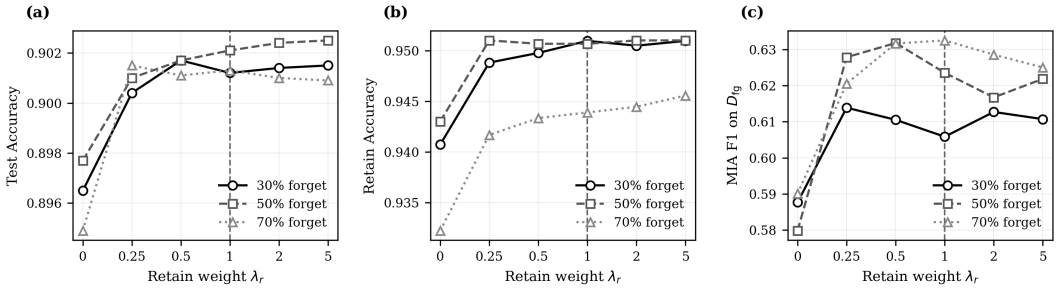

Figure 9: **Sensitivity to the retain-loss weight λ_r in sample-level unlearning.** (a) Test accuracy, (b) retain-set accuracy, and (c) MIA F1 on $\mathcal{D}_{\text{fg}}$ for 30%, 50%, and 70% forget ratios. The dashed line marks the finalized setting $\lambda_r = 1$.

Rényi order α. We first examine the sensitivity of FedRUn to the Rényi order α on CIFAR-10 while keeping the remaining client-level configuration fixed. Figure 10 reports the resulting test accuracy and MIA F1. Test accuracy remains nearly unchanged across the evaluated values of α, indicating that global utility is relatively insensitive to this parameter in the tested range. In contrast,

MIA F1 changes more noticeably: $\alpha = 1.1$ produces the lowest attack F1, while the values for $\alpha \geq 1.5$ are comparatively similar. This result shows that α primarily influences the forgetting behavior rather than the retained global utility.

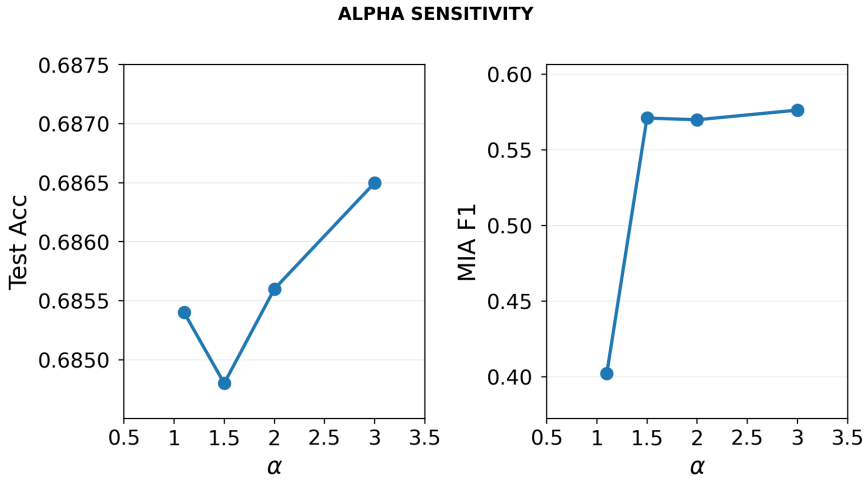

Figure 10: **Sensitivity to the Rényi order α on CIFAR-10.** Test accuracy remains nearly unchanged across the evaluated values of α, whereas MIA F1 varies more substantially, indicating that α mainly controls the forgetting response.

Overall, the sweep indicates that α affects the strength of forgetting more strongly than global utility. The finalized client-level configuration uses $\alpha = 1.5$. This sensitivity analysis characterizes the behavior of the Rényi objective rather than selecting an optimal value.

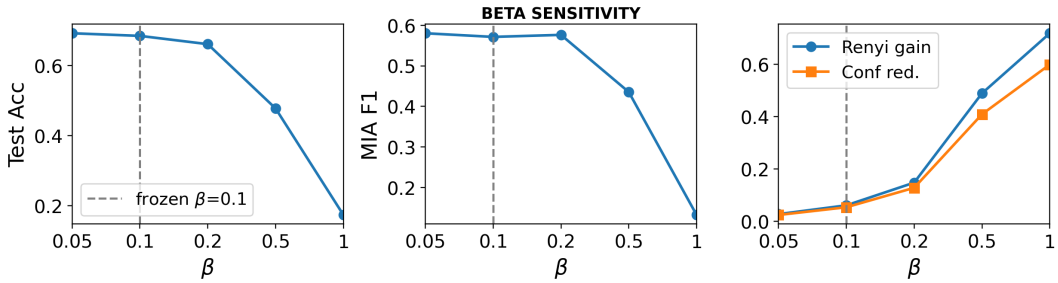

Figure 11: **Sensitivity to the server scaling parameter β on CIFAR-10.** Increasing β strengthens the applied client-computed update, reducing MIA F1 but also decreasing test accuracy. The corresponding Rényi-entropy gain and confidence reduction increase with β. The dashed line marks the finalized setting $\beta = 0.1$.

Figure 11 shows that β controls the utility–unlearning trade-off. Larger values apply a stronger client-computed update, increasing entropy and reducing predictive confidence on the forget set, but eventually degrading global test accuracy. The finalized $\beta = 0.1$ therefore uses a conservative update magnitude that preserves utility while still producing a measurable forgetting effect.

Forget-loss weight λ_f. We next vary the weight of the Rényi-entropy forget objective while keeping the remaining CIFAR-10 client-level configuration fixed. Figure 12 shows that both test accuracy and MIA F1 change only marginally over the evaluated range. Test accuracy remains approximately 0.685, while MIA F1 varies by less than 10^{-3}. This indicates that FedRUn is relatively insensitive to moderate changes in λ_f around the finalized setting $\lambda_f = 0.005$.

Number of local unlearning steps S. We next vary the number of local FedRUn optimization steps on CIFAR-10 while keeping the remaining client-level configuration fixed. As shown in Fig. 13, increasing S strengthens the forgetting effect, with MIA F1 decreasing substantially as more

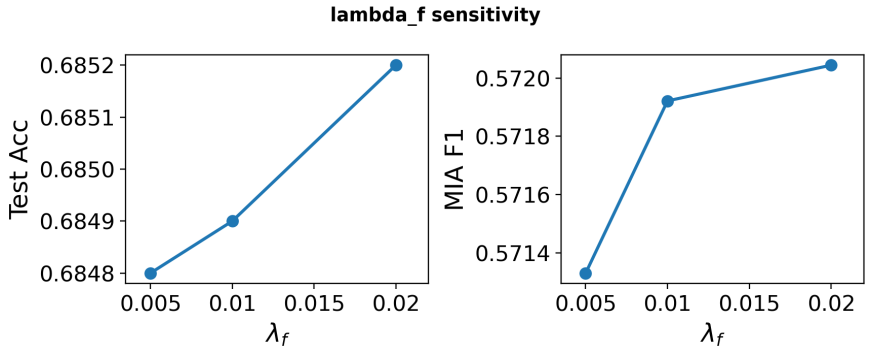

Figure 12: **Sensitivity to the forget-loss weight λ_f on CIFAR-10.** Test accuracy and MIA F1 remain nearly unchanged across the evaluated values of λ_f, indicating limited sensitivity around the finalized setting $\lambda_f = 0.005$.

local steps are performed. This comes with a gradual reduction in test accuracy and a corresponding increase in client-side computation. Thus, S directly controls the trade-off between unlearning strength, model utility, and post-request computational cost. The finalized client-level configuration uses $S = 8$, providing a conservative operating point that preserves utility while keeping the local unlearning procedure short.

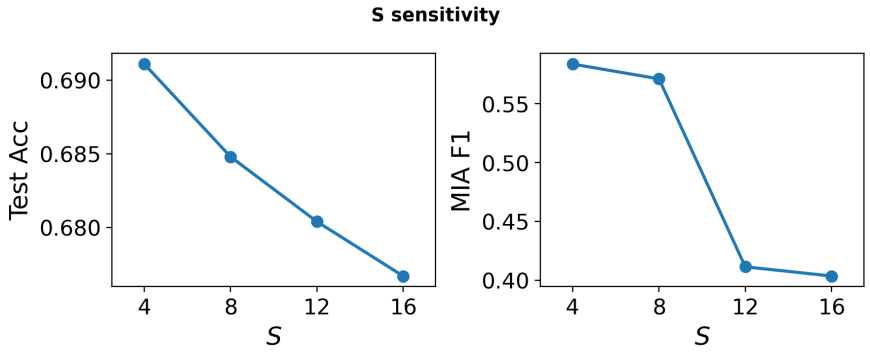

Figure 13: **Sensitivity to the number of local unlearning steps S on CIFAR-10.** Increasing S reduces MIA F1 but gradually decreases test accuracy, illustrating the utility–unlearning trade-off.