Empirical Study of Federated Unlearning: Efficiency and Effectiveness - Supplementary Material

Thai-Hung Nguyen 20HUNG.NT@VINUNI.EDU.VN
Hong-Phuc Vu 20PHUC.VH@VINUNI.EDU.VN
College of Engineering & Computer Science, VinUniversity, Hanoi, Vietnam
Dung Thuy Nguyen DUNG.NT2@VINUNI.EDU.VN
VinUni-Illinois Smart Health Center, VinUniversity, Hanoi, Vietnam
Tuan Minh Nguyen TUAN.NM@VINUNI.EDU.VN
Khoa D Doan DOANKHOADANG@GMAIL.COM
Kok-Seng Wong WONG.KS@VINUNI.EDU.VN
VinUni-Illinois Smart Health Center, VinUniversity, Hanoi, Vietnam
College of Engineering & Computer Science, VinUniversity, Hanoi, Vietnam

Editors: Berrin Yanıkoğlu and Wray Buntine

In this supplementary material, we provide the pseudocode of two unlearning methods, Projected Gradient Ascent (PGA) (Halimi et al., 2022) and FedEraser (Liu et al., 2021) in Section A. Next, we provide the experimental setting and results of our experiments on CIFAR-100 in Section B.

Appendix A. FedUnlearn Method

A.1. Projected Gradient Ascent (PGA)

The natural approach to unlearning involves updating the model parameters in the opposite direction of the loss function gradient. However, simply maximizing the loss using gradient ascent can pose problems, particularly when the loss is unbounded, as is often the case with, for instance, the cross-entropy loss. In situations with an unbounded loss, each gradient step leads toward a model that increases the loss, and after several steps, it may converge to an arbitrary, random-like model.

To tackle this challenge, PGA (Halimi et al., 2022) projects the updated model parameters onto a ball centered at the reference model parameters. These reference model parameters are calculated as the average of the model parameters of all clients except the removed client. The projection ensures that the updated model parameters remain close to the reference model parameters. The pseudocode for PGA can be found in Algorithm 1.

Algorithm 1 Unlearning via Projected Gradient Ascent (Halimi et al., 2022)

Unlearning Parameters: learning rate η_u, batch size B_u, number of epochs E_u, clipping radius δ, and early stopping threshold τ
Split dataset D_i into train set D_i^{tr} and validation set D_i^{val}
Set $\mathbf{w}_{ref} \leftarrow \frac{1}{N-1}\left(N\mathbf{w}^T - \mathbf{w}_i^{T-1}\right) = \frac{1}{N-1}\sum_{i\neq j}\mathbf{w}_j^{T-1}$
Define $\mathcal{P}(\mathbf{w})$ as the projection of $\mathbf{w} \in \mathbb{R}^d$ onto the ℓ_2-norm ball $\Omega = \{\mathbf{v} \in \mathbb{R}^d : ||\mathbf{v} - \mathbf{w}_{ref}|| \leq \delta\}$
Initialize unlearning model as $\mathbf{w} \leftarrow \mathbf{w}_i^{T-1}$
$\mathcal{D}_i \leftarrow$ (split D_i^{tr} into batches of size B_u)
for each local epoch $e = 1$ **to** E_i **do do**
  for batch b in $\mathcal{D}_i$ **do do**
    $\mathbf{w} \leftarrow \mathcal{P}\left(\mathbf{w} + \eta_n \nabla F_i^u(w; b)\right)$
    if Accuracy$\left(\mathbf{w}; D^{val}\right) < \tau$ **then then**
      Set $\mathbf{w}_i^u \leftarrow \mathbf{w}$ and return $\mathbf{w}_i^u$ to server
    end if
  end for
end for
Set $\mathbf{w}_i^u \leftarrow \mathbf{w}$ and return $\mathbf{w}_i^u$ to server

A.2. FedEraser

To efficiently eliminate the influence of a target client's data from the trained global model without retraining from scratch, the central server in FedEraser (Liu et al., 2021) retains client updates at regular intervals during the global model's training process. These retained updates include the index of the corresponding round. Subsequently, the server uses this information to calibrate the retained updates, enabling the reconstruction of the unlearned global model. FedEraser follows a four-step process:

- **Step 1: Calibration Training** — This step is executed on the client's side. Each client initializes its local model with the retained updates and subsequently trains it for a few epochs using its own dataset.
- **Step 2: Update Calibrating** — On the central server, the retained updates undergo calibration. This process utilizes the calibrated local models from each client to refine the retained updates.
- **Step 3: Calibrated Update Aggregating** — The server aggregates the calibrated updates, forming the basis for constructing the unlearned global model.
- **Step 4: Unlearned Model Updating** — The unlearned global model is updated based on the aggregated calibrated updates.

It is important to highlight that the first step is executed by individual clients (excluding the removed clients), while the subsequent stages are conducted on the server. For a more comprehensive understanding of the FedEraser implementation, please consult the pseudocode outlined in Algorithm 2.

Algorithm 2 Unlearning via FedEraser (Liu et al., 2021)

Require: Initial global model $\mathcal{M}^1$; retained client updates U
Require: Target client index k_n
Require: Number of global calibration round T
Require: Number of local calibration training epoch E_{cali}
Central server executes:
for each round $R_{t_j}, j \in \{1, 2, ..., T\}$ **do**
for each client $C_{k_c}, k_c \in \{1, 2, ..., K\}$ k_u **in parallel do**
$\widehat{U}_{k_c}^{t_j} \leftarrow CaliTrain\left(C_{k_c}, \widetilde{\mathcal{M}}_{k_c}^{t_j}, E_{cali}\right)$
$\widetilde{U}_{k_c}^{t_j} \leftarrow |U_{k_c}^{t_j}| \frac{\widehat{U}_{k_c}^{t_j}}{\|\widehat{U}_{k_c}^{t_j}\|}$; ▷ `Update Calibrating`
end for
$\widetilde{U}^{t_j} \leftarrow \frac{1}{(K-1)\sum w_{k_c}} \sum_{k_c} w_{k_c} \widetilde{U}_{k_c}^{t_j}$; ▷ `Update Aggregating`
$\widetilde{M}^{t_{j+1}} \leftarrow \widetilde{M}^{t_j} + \widetilde{U}^{t_j}$; ▷ `Model Updating`
end for
CaliTrain $\left(C_{k_c}, \widetilde{\mathcal{M}}_{k_c}^{t_j}, E_{cali}\right)$: // Run on client C_{k_c}
for each local training round j from 1 to E_{cali} **do**
$\widetilde{M}_{k_c}^{t_j}|_{j+1} \leftarrow Train\left(\widetilde{M}_{k_c}^{t_j}|_j, D_{k_c}\right)$
end for
$\widehat{U}_{k_c}^{t_j} \leftarrow$ Calculation Update$\left(\widetilde{M}_{k_c}^{t_j}|_{E_{cali}}, \widetilde{M}_{k_c}^{t_j}|_1\right)$
return $\widehat{U}_{k_c}^{t_j}$ to the central server

Appendix B. Additional experiments on CIFAR-100

B.1. Experimental setup

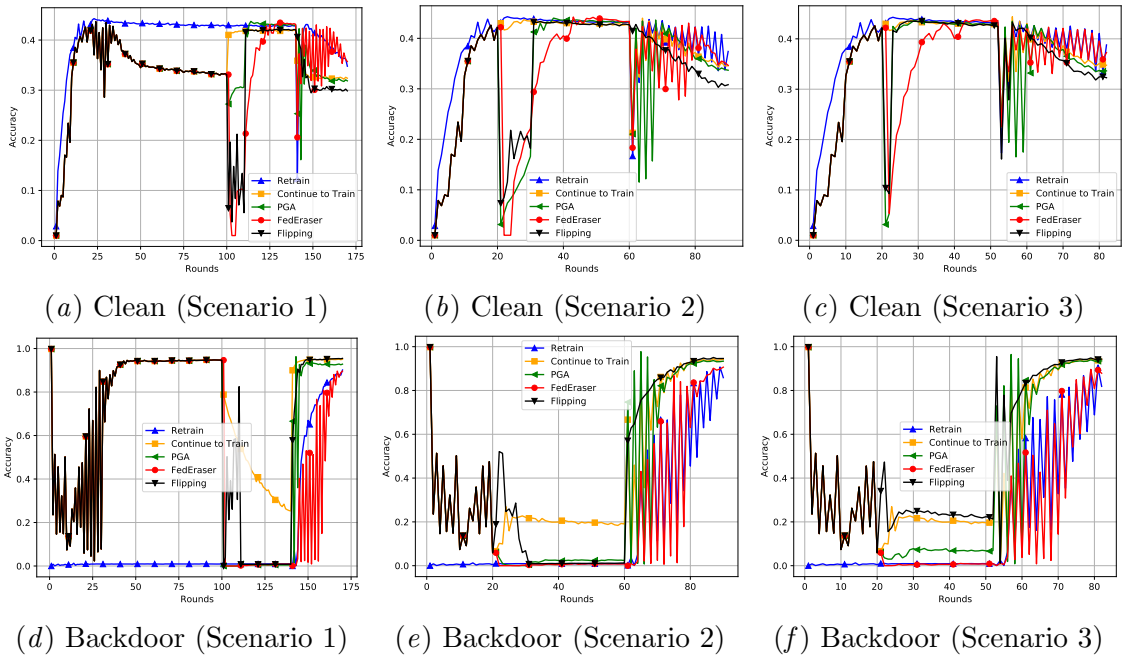
Additional experiments were conducted using CIFAR-100, applying the same experimental setup as for the MNIST and CIFAR-10 datasets. The federated learning (FL) training configuration comprised five clients with an independent and identically distributed (IID) data distribution. For these experiments, backdoor samples were introduced into the training data of the removed client (client 0) at a backdoor ratio of 0.9. The trigger pattern used consisted of a 3x3 square with fixed pixel values in the bottom right corner. The backdoor label was set to `99` (with a label set of 0-99). We provided the detailed hyperparameters in Table 1.

B.2. Experimental Analysis and Discussion

In this section, we provide experiment results on CIFAR-100. In general, the results are consistent with our main analysis with MNIST and CIFAR-10.

Table 1: Hyperparameters for Three Scenarios in the FedUnlearn Setup on CIFAR-100

Hyperparameter	Scenario 1	Scenario 2	Scenario 3
FL communication rounds	100	20	20
Unlearning rounds	10	10	2
Post-unlearning rounds	30	30	30
Returning rounds	30	30	30



(a) Clean (Scenario 1) (b) Clean (Scenario 2) (c) Clean (Scenario 3)

(d) Backdoor (Scenario 1) (e) Backdoor (Scenario 2) (f) Backdoor (Scenario 3)

Figure 1: The effectiveness of unlearning methods compared to the Retrain (baseline) in three scenarios.

RQ1: What are the impacts of different unlearning methods on CIFAR-100 dataset?

In particular, Figs 1(a) and 1(d) depict the impact of various unlearning methods on CIFAR-100. Notably, the PGA, FedEraser, and Flipping methods demonstrate their ability to significantly reduce backdoor accuracy while maintaining a strong performance on clean data before the beginning of Post-unlearning rounds. Considering the detailed accuracy of the global model on both clean and backdoor test sets, as presented in Table 2, it becomes evident that the "Continue to Train" method is not as effective. It exhibits a slower reduction in backdoor accuracy when compared to other methods such as FedEraser, PGA, and Flipping.

Table 2: The effectiveness of four unlearning methods compared to Retrain (baseline).

Dataset	Method	BU		SU		FU		SPU		FPU		SR		FR	
		CA	BA	CA	BA	CA	BA	CA	BA	CA	BA	CA	BA	CA	BA
CIFAR-100	Retrain	**0.428**	0.009	0.429	0.009	0.429	0.009	0.428	0.009	0.428	0.009	0.123	0.000	0.347	0.901
	ConTrain	0.331	0.946	0.410	0.788	0.419	**0.559**	0.418	0.538	0.419	0.253	0.358	0.899	0.323	0.950
	PGA	0.331	0.946	0.272	**0.000**	0.305	0.000	**0.418**	0.009	0.430	0.004	0.253	0.665	0.318	**0.928**
	FedEraser	0.331	0.946	0.331	0.946	0.099	0.000	**0.213**	0.002	0.431	0.008	0.205	0.000	0.357	**0.896**
	Flipping	0.331	0.946	0.064	**0.000**	0.055	0.824	**0.415**	0.007	0.420	0.007	0.406	0.577	0.298	**0.954**

ConTrain: Continue to Train
CA: Clean Accuracy
BA: Backdoored Accuracy
BU: Before Unlearning
SU: Start Unlearning
FU: Finish Unlearning
SPU: Start Post Unlearning
FPU: Finish Post Unlearning
SR: Start Returning
FR: Finish Returning

Table 3: Comparative Analysis of Unlearning Time (in seconds) among Five Methods across Three Scenarios on CIFAR-100 dataset

Scenario	1	2	3
Retrain	6719.94(1.00x)	1376.75(1.00x)	1369.64(1.00x)
Continue to Train	1276.34(5.27x)	1306.08(1.05x)	185.82(7.37x)
PGA	**218.51(30.75x)**	**207.39(6.64x)**	**23.62(57.98x)**
FedEraser	1220.09(5.51x)	1229.05(1.12x)	157.61(8.69x)
Flipping	1595.61(4.21x)	**1610.73(0.85x)**	324.55(4.22x)

1: Late unlearning 2: Early unlearning
3: Early unlearning with small unlearning rounds

RQ2: How early and late unlearning affect the general performance of the unlearning methods?

Figs. 1(b), 1(e), 1(c), and 1(f) demonstrate that the performance of the unlearning methods can be influenced by the communication round at which we initiate the unlearning process, as well as the number of unlearning rounds. Specifically, while the PGA and Flipping methods exhibit a reduction in backdoor accuracy during unlearning rounds, they are not as effective as the FedEraser method in all scenarios. Regarding clean accuracy, the PGA, Flipping, and FedEraser methods can maintain clean accuracy after the unlearning process.

RQ3: How effectively does each method facilitate returning clients?

As depicted in Fig. 1, only PGA and Flipping methods enable rapid recovery, while FedEraser requires more training rounds to achieve comparable backdoor accuracy.

Efficiency Analysis

Unlearning Time. Regarding efficiency, Table 3 reflects the same findings as our experiments with MNIST and CIFAR-10. In all scenarios, PGA is the fastest method, while unlearning with Flipping can be slower than the baseline method. This reduced speed can be attributed to the small number of FL training rounds and the significant number of unlearning rounds.

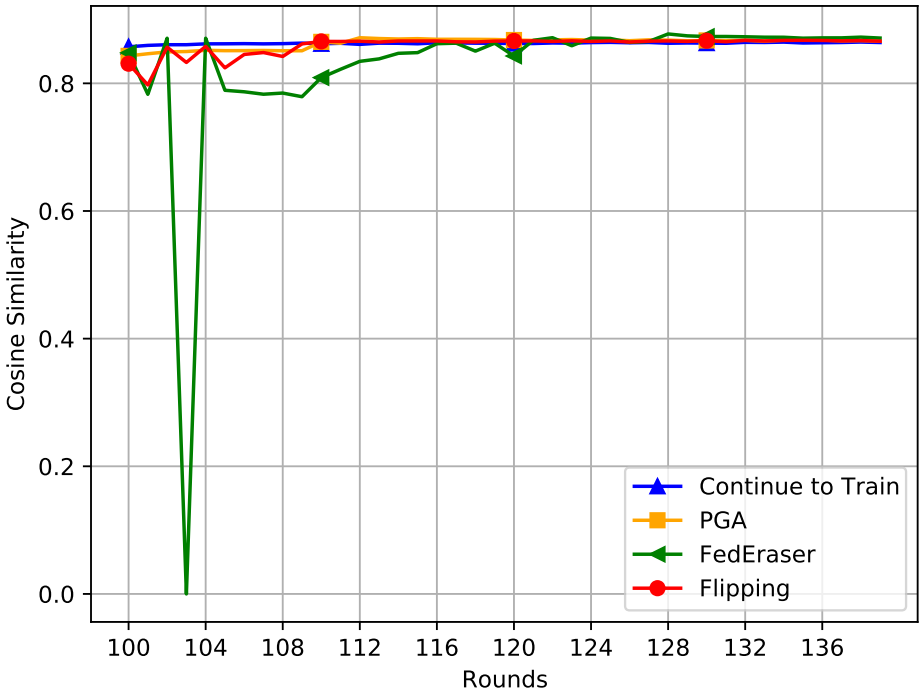


Figure 2: Cosine Similarity between the unlearned and baseline model in Scenario 1 on CIFAR-100 dataset

Cosine Similarity

Examining the Cosine Similarity between the unlearned and baseline models, as illustrated in Fig. B.2, during the unlearning rounds, FedEraser causes the unlearned model's behavior to diverge significantly from that of the baseline model. This divergence occurs because FedEraser updates the global model based on historical client models at interval rounds. However, starting from the post-unlearning rounds, PGA, Flipping, and FedEraser methods maintain a stable Cosine Similarity around 0.9. This suggests their convergence to align the unlearned model's behavior with that of the baseline model, consistent with our MNIST and CIFAR-10 results.

References

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

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)*, pages 1–10. IEEE, 2021.