# TACKLING FEDERATED UNLEARNING AS A PARAMETER ESTIMATION PROBLEM

#### Antonio Balordi

CASD - Italian Defense University Rome, Italy antonio.balordi@unicasd.it

#### Fabio Stella

Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milan, Italy fabio.stella@unimib.it

#### Lorenzo Manini

DIFA - University of Bologna Bologna, Italy lorenzo.manini@studio.unibo.it

#### Alessio Merlo

CASD - Italian Defense University Rome, Italy alessio.merlo@unicasd.it

#### **ABSTRACT**

Privacy regulations require the erasure of data from deep learning models. This is a significant challenge that is amplified in Federated Learning, where data remains on clients, making full retraining or coordinated updates often infeasible. This work introduces an efficient Federated Unlearning framework based on information theory, modeling leakage as a parameter estimation problem. Our method uses second-order Hessian information to identify and selectively reset only the parameters most sensitive to the data being forgotten. This model-agnostic approach supports categorical and client unlearning without requiring server access to raw client data after the initial information aggregation. Evaluations on benchmark datasets demonstrate strong privacy (MIA success near random, categorical knowledge erased) and high performance (Normalized Accuracy against retrained benchmarks of  $\approx 0.8$ ), while also achieving increased efficiency over complete retraining. Furthermore, in a targeted backdoor attack scenario, our framework effectively neutralizes the malicious trigger, restoring model integrity. This offers a practical solution for data forgetting in FL.

Keywords Federated Unlearning · Parameter estimation · Right To Be Forgotten · Data Privacy · Hessian computation.

# 1 Introduction

The rapid adoption of deep learning has led to models trained on massive datasets that achieve outstanding results across domains. However, such outcomes come at the cost of privacy: Neural Networks (NN) are known to memorize training data, potentially leaking sensitive information ([1], [2]). This privacy concern has led to *machine unlearning* gaining momentum, which generally refers to a set of techniques aimed at allowing organizations to meet the proper privacy constraints by selectively removing the influence of specific data from trained models. This capability has recently become crucial as privacy regulations, such as GDPR ([3], [4]) or CCPA ([5], [6]), provide the user with the "right to be forgotten". As a consequence, clients can request the complete removal of their data in any training scenario.

This work will focus on Federated Learning (FL), a learning technique that allows users to collectively obtain the benefits of shared models without the need to share the data [7]. Despite this privacy-preserving design, the need to retroactively remove a user's data contribution presents a significant challenge, leading to the development of the field of *Federated Unlearning (FU)*. Here, the *gold standard* refers to the practice of *full retraining*; that is, the model is discarded and a new one is trained from scratch without the removed client; this approach grants an *exact unlearning* ([8], [9]). However, it often leads to significant communication overhead, inefficient resource utilization, and high computational costs, rendering it impractical for many real-world federated learning (FL) deployments ([10], [8]).

Consequently, applying exact unlearning through full retraining reveals that it is often infeasible in FL scenarios with high client turnover or when dealing with large models and datasets [11]. Therefore, there is a strong need for FU methods that can approximate the guarantees provided by the gold standard while operating within feasible resource constraints and avoiding the need to retrain from scratch.

**Contributions.** This paper introduces a new federated unlearning framework that conceptualizes information leakage as a *parameter estimation problem*. This information-theoretic perspective enables the development of a high-performing unlearning method that selectively resets only those parameters identified as most informative about the data to be unlearned, utilizing second-order information (Hessian diagonals) computed post-training. Our approach is broadly applicable across diverse network architectures, supports arbitrary dataset unlearning, and integrates seamlessly into the federated setting, enabling server-side unlearning without client data access and retraining overhead.

**Paper Structure**. The remainder of the paper is organized as follows. Section 2 reviews relevant previous work on federated unlearning. Section 3 introduces the federated unlearning problem. Our proposed framework is detailed in Section 4, followed by a description of the proposed algorithm in Section 5. Experimental setup and datasets in Section 6. We present some numerical results in Section 7 and discuss them in Section 8. Section 9 concludes the paper, summarizing key findings and describing future directions.

## 2 Related Work

In centralized settings, extensive research has explored various MU techniques. Among these, methods such as SISA (sharded retraining) [8], which focus on efficient exact removal via sharding, and influence functions [12], which offer a theoretical lens on the impact of data, have been extensively explored.

Focusing on the federated learning context, we have FU methods that can be broadly categorized as follows, each with significant limitations.

- **Retraining-Based/Approximation**: These methods approximate complete retraining, for example, through knowledge distillation [13], [14]. Although faster, they still require multiple rounds of communication between clients participating in the training and significant computation. More critically, in knowledge-distillation setups, the *student* often inherits the *teacher's* membership signals, so a model can 'pass' one audit while still retaining information—making the quality of forgetting hard to measure and increasingly fragile as the forget set or model complexity grows [15, 16].
- **Gradient/Update Manipulation**: Several approaches attempt to reverse the impact of unlearned data by manipulating gradients or model updates. For example, [17] proposed methods, including gradient ascent on the forgotten client's data. However, these are often heuristic; forcing misprediction does not guarantee the removal of memorized information and can leave the model vulnerable to reconstruction or membership inference attacks, as it could still retain encoded information, as shown in [2] and in [18].
- Parameter Masking/Perturbation: These techniques alter parameters related to forgotten data, for example, pruning or adding noise [19], [20]. Identifying relevant parameters is non-trivial; currently, most methods typically fail to capture complex dependencies [21]. Determining the extent of the modification requires tuning and risk degradation of the utility [22]. Addressing this, our proposed methodology (Section 4) offers an information-theoretic strategy for the identification and resetting of the targeted parameters within this general class of methods.

## 3 Federated Unlearning

# 3.1 Objectives

In FL, some clients and a central aggregating server collaboratively train a machine learning model while maintaining the locality of the data, thus obtaining improved privacy and security.

Then, FU aims to achieve one or more of the following objectives [10]:

- Sample Unlearning: the removal of an arbitrary subset of data samples.
- **Client Unlearning:** the removal of the entire dataset of a given client. This scenario most directly corresponds to the "*Right to Be Forgotten*" (RTBF) principle.
- Class Unlearning: the removal of all data samples belonging to a specific class c, potentially affecting the data contributions of multiple clients.

It should be noted that both Client Unlearning and Class Unlearning can be viewed as specific instances of the Sample Unlearning problem. In each case, the goal is to remove a subset of samples defined by a particular characteristic (i.e., belonging to a specific client or class).

In FU, the unlearning process should provide the same level of privacy as the original federated learning process: the clients' data should remain private and not be accessible by the server or other clients.

### 3.2 Challenges

The requirements outlined in Section 3.1 pose a well-defined set of challenges, as effectively detailed in [10]. We revisit key aspects of these challenges to motivate and present the specific features of our proposed Federated Unlearning framework and methodology.

- Nondeterministic Training Process: Dynamic client participation in FL complicates the exact reproduction of past training states for unlearning, often leading to discarded knowledge. The theoretical framework presented here was developed based on the assumption of deterministic training, as demonstrated by the empirical results in Section 7. While we believe the framework may be generalizable to non-deterministic scenarios, this will require substantial additional testing. Therefore, exploring its applicability to stochastic training processes is a clear direction for future work.
- *Inscrutable Model Updates and Finite Memory*: The server's potential inability to access individual client updates and the impracticality of storing all historical model states restrict methods reliant on reversing past aggregations. To avoid this, our framework leverages the computation of second-order statistics (Hessian diagonals or derived information scores) on the final model state.
- *Inscrutable Local Datasets*: By design, in FL, client data remains private and cannot be accessed by the server or other clients for unlearning. This prevents methods that require direct access to the data. Our framework upholds this principle by requiring clients to share only second-order statistics and not the raw data itself. The subsequent retraining phase also adheres to standard FL protocols.
- *Iterative Learning Process*: The iterative nature of FL means that a naive rollback to remove the client's contribution from a previous round would invalidate subsequent aggregations. Our approach directly modifies the current model state, removing the specific imprint of the unlearned data, followed by a brief reconvergence phase with the remaining knowledge. This setting is detailed in our experiments.
- Data Heterogeneity: In an FL setting, data might often be non-IID across clients; however, due to the secrecy of the clients' datasets, assessing the data heterogeneity is not straightforward. Nevertheless, the data-agnostic model of our framework is well-suited to leverage this, as it inherently adapts to data heterogeneity without requiring prior knowledge of its extent.

## 4 Parameter Estimation Framework

To characterize the information encoded in each of the network parameters, we consider a scenario in which a hypothetical attacker attempts to infer details about the target dataset (TD) <sup>1</sup> through interactions with the neural network. We model this attack as a *parameter estimation problem* where the parameter to estimate is whether the target dataset was used during training or not, and the observables are the NN parameters. We then leverage results from information theory to define a *Target Information Score (TIS)* that quantifies how much each parameter contributes to the estimability of information about the TD. In Section 4, we use it to define a procedure to forget the TD.

The attack scenario is defined as follows:

- The attacker has full knowledge about the training procedure. The training procedure is modelled as a function<sup>2</sup>
   T: D → B from the space of all possible datasets D to the space of all possible network parameters B.
- The attacker knows the trained network parameters  $\tilde{\boldsymbol{\theta}} \in \mathfrak{P}$ .
- The attacker is aware that only two scenarios are possible: either the target dataset (TD) was included in the training process, or it was entirely excluded. Formally, let us denote the TD as  $\tilde{D}_T \in \mathfrak{D}$ , then the two scenarios correspond to two possible training datasets

- 
$$\tilde{D}_1 \in \mathfrak{D}$$
 such that  $\tilde{D}_T \subset \tilde{D}_1$ .

<sup>&</sup>lt;sup>1</sup>A formal definition of the Target Dataset will be provided in Section 5

<sup>&</sup>lt;sup>2</sup>The training procedure is considered to be deterministic for simplicity

- 
$$\tilde{D}_0 \in \mathfrak{D}$$
 such that  $\tilde{D}_0 = \tilde{D}_1 \setminus \tilde{D}_T$ .

The information the attacker wants to infer is which of the two datasets was used; formally, she thus wants to estimate a binary parameter

$$t = \begin{cases} 0 & \text{if the training dataset was } \tilde{D}_0 \\ 1 & \text{if the training dataset was } \tilde{D}_1 \end{cases}$$
 (1)

so that we also denote the (parametric) training dataset as  $\tilde{D}_t$ .

• The attacker has an "arbitrarily good" subjective probability distribution regarding the possible training datasets. Precisely, we consider the attacker not to know the exact datasets  $\tilde{D}_t$  but that her knowledge provides her with two subjective probability distributions defined on  $\mathfrak{D}$  and peaked around  $\tilde{D}_t$ , as follows

$$\rho_t^{data}(D) = \mathcal{N}(D; \tilde{D}_t, \epsilon \, \mathbb{1}) \qquad t \in \{0, 1\}$$

where  $\mathcal{N}$  is a multivariate Gaussian distribution defined on the sample space of each dataset, and  $\epsilon$  quantifies the uncertainty about the datasets<sup>3</sup>. We thus have that in the limit of vanishing  $\epsilon$  the prior tends to a Dirac delta centered on the true datasets, i.e., exact knowledge of them. Moreover, from their definitions, it follows that

$$\rho_1^{data}(D) = \mathcal{N}(D; \tilde{D}_0, \epsilon \mathbb{1}) \mathcal{N}(D; \tilde{D}_T, \epsilon \mathbb{1})$$
(3)

$$= \rho_0^{data}(D)\,\rho_T^{data}(D) \tag{4}$$

since  $\tilde{D}_1$  is the disjoint union of  $\tilde{D}_0$  and  $\tilde{D}_T$ , and where we defined the distribution of the TD as

$$\rho_T^{data}(D) = \mathcal{N}(D; \tilde{D}_T, \epsilon \, \mathbb{1}) \tag{5}$$

From the attacker's point of view, her lack of knowledge about the precise content of the two possible datasets, quantified by  $\epsilon$ , leads to uncertainty regarding the expected trained network parameters in the two possible cases. Specifically, this is described by the fact that the training dataset D is a random outcome sampled from one of the two probability distributions  $\rho_t^{data}(D)$ . The trained network parameters are a random variable given by  $\theta = \mathcal{T}(D)$ . Thus, we will have two probability distributions for the trained parameters defined in  $\mathfrak{P}$ :

$$\rho_t^{par}(\boldsymbol{\theta}) \qquad t \in \{0, 1\} \tag{6}$$

Then, if both  $\rho_0^{par}(\tilde{\boldsymbol{\theta}}) > 0$  and  $\rho_1^{par}(\tilde{\boldsymbol{\theta}}) > 0$ , the attacker will not be able to infer with complete certainty the value of t. Still, she will only be able to assign a likelihood to the two possibilities. Formally, the procedure the attacker uses to obtain her best guess of the parameter t is an estimator  $\hat{t}:\mathfrak{P}\to\{0,1\}$ .

We now want to study the information-theoretical limits to the efficacy of the attack procedure; that is, the efficacy of any estimator  $\hat{t}$ . Our analysis will ultimately take advantage of theorems that provide a lower bound on the variance of  $\hat{t}$ . However, these theorems require the parameter to be estimated to be continuous, which implies having a one-dimensional family of probability distributions. To satisfy this hypothesis, we need a continuous embedding of t, the choice of which is not, in general, unique. To fix ideas, one might picture the embedding as choosing any smooth interpolation from  $\tilde{D}_0$  to  $\tilde{D}_1$  parametrized by  $t \in [0,1]^4$ , then by the same arguments as in the previous paragraphs this leads to a family of probability distributions of the trained parameters

$$\rho_t^{par}(\boldsymbol{\theta}) \qquad t \in [0, 1] \tag{7}$$

From now on, we will refer to t as the parameter of the continuous embedding.

We can then use the Cramér-Rao bound, stating that if we have a 1-dim family of probability distributions  $\rho_t^{par}(\boldsymbol{\theta})$ , smoothly parametrized by t, then the variance of any unbiased estimator  $\hat{t}$  is bounded as follows:

$$\operatorname{var}(\hat{t}) \ge \frac{1}{I(\tilde{t})} \tag{8}$$

where  $\tilde{t}$  is the real value of t and  $I(\tilde{t})$  is the Fisher information defined as

$$I(\tilde{t}) = \mathbb{E}_{\rho_{\tilde{t}}^{par}(\theta)} \left[ \left( \frac{\partial \log(\rho_{t}^{par}(\theta))}{\partial t} \bigg|_{t-\tilde{t}} \right)^{2} \right]$$
(9)

We thus have that I(1) quantifies the total information available to the attacker about the presence of the TD in the training dataset when the TD was used in the training.

The expectation value in Equation (9) involves integrating over the parameter space  $\mathfrak{P}$ , and we will see that, under some assumptions, this integration leads to a sum of 1-dimensional integrals over the possible values of each network parameter. This allows us to consider the contribution of each network parameter to the total information, or rather, how much information about the TD is expressed by each network parameter.

<sup>&</sup>lt;sup>3</sup>Here and after, 1 is intended to be the identity matrix, and its dimension is the same as that of the multivariate Gaussian

<sup>&</sup>lt;sup>4</sup>The continuous embedding we used is a generalization of the one presented here and is treated in Section 4.1

#### 4.1 Distribution of Network Parameters

To compute the Fisher information in Equation (9), we need the expression of  $\rho_t^{par}(\boldsymbol{\theta})$  in a neighborhood  $t \in [1 - \delta, 1]$  for some  $\delta > 0$  to compute the derivative in Equation (9).

We assume that the training procedure involves the minimization of a loss function

$$\mathcal{L}(\cdot; D): \mathfrak{P} \to \mathbb{R}$$
 where  $D \in \mathfrak{D}$  (10)

so that the training function  $\mathcal{T}$  is such that for each dataset  $\tilde{D}$  the trained parameters  $\tilde{\boldsymbol{\theta}} = \mathcal{T}(\tilde{D})$  correspond to the minimum of  $\mathcal{L}(\boldsymbol{\theta}; \tilde{D})$ , and thus

$$\left[\nabla_{\boldsymbol{\theta}} \mathcal{L}\right](\tilde{\boldsymbol{\theta}}; \tilde{D}) \equiv \left. \frac{\partial \mathcal{L}(\boldsymbol{\theta}; \tilde{D})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta} = \tilde{\boldsymbol{\theta}}} = 0 \tag{11}$$

and the Hessian is positive-semidefinite. We further assume that the Loss function is the average of a sample loss function over all the samples of the dataset. Thus, for any fixed dataset D and any partition of it  $\{D_i\}_{i=1,\dots,k}$ , the total loss function is

$$\mathcal{L}(\boldsymbol{\theta}; D) = \sum_{i=1}^{k} \frac{N_i}{N} \mathcal{L}(\boldsymbol{\theta}; D_i)$$
 (12)

where N and  $N_i$  are the number of samples in D and in  $D_i$  respectively.

With these assumptions, we argue for the following radical approximation. In the limit of vanishing uncertainty  $\epsilon$  of the prior of the attacker, we expect  $\rho_1^{par}(\boldsymbol{\theta})$  to tend to a Dirac delta centered on  $\tilde{\boldsymbol{\theta}} = \mathcal{T}(\tilde{D}_1)$ . Thus, for small  $\epsilon$ , we expect the distribution to peak around  $\tilde{\boldsymbol{\theta}}$ . Since the trained parameters  $\tilde{\boldsymbol{\theta}}$  lie in a minimum of the loss, we can approximate the loss in a small neighborhood around  $\tilde{\boldsymbol{\theta}}$  as

$$\mathcal{L}(\boldsymbol{\theta}; \tilde{D}_1) \approx \mathcal{L}(\tilde{\boldsymbol{\theta}}; \tilde{D}_1) + \frac{1}{2} (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})^T \cdot H_1 \cdot (\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}})$$
(13)

where  $H_1$  is the Hessian of the Loss for the network parameters, computed in  $\tilde{\theta}$ . Geometrically, this amounts to approximating the loss function with an N-dimensional paraboloid, where N is the number of network parameters. Then, intuitively, we argue that small changes to the dataset will change the loss function so that the new minimum will be displaced more along the directions where the paraboloid is wide and less along the ones where the paraboloid is narrow. More formally, we consider the fact that small perturbations of the dataset will lead to a small functional perturbation of the loss function, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}; \tilde{D}_1 + \delta D) \approx \mathcal{L}(\boldsymbol{\theta}; \tilde{D}_1) + \delta \mathcal{L}(\boldsymbol{\theta})$$
(14)

Then the new minimum  $\theta'$  will be displaced by a small amount according to

$$\left[\nabla_{\boldsymbol{\theta}} \mathcal{L}\right] (\boldsymbol{\theta}'; \tilde{D}_1 + \delta D) = 0 \tag{15}$$

$$\Rightarrow H_1 \cdot (\boldsymbol{\theta}' - \tilde{\boldsymbol{\theta}}) + \left[ \nabla_{\boldsymbol{\theta}} \delta \mathcal{L} \right] (\boldsymbol{\theta}') = 0 \tag{16}$$

$$\Rightarrow \quad \boldsymbol{\theta}' - \tilde{\boldsymbol{\theta}} = -H_1^{-1} \cdot \left[ \nabla_{\boldsymbol{\theta}} \delta \mathcal{L} \right] (\boldsymbol{\theta}') \tag{17}$$

And we see that the displacement depends on the inverse of the Hessian, so that the wider the paraboloid is along some axis, the larger the displacement will be in that direction.

Then, for a peaked Gaussian distribution around  $\tilde{D}_1$ , we expect the distribution of the network parameters to be approximately

$$\rho_1^{par}(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta}; \tilde{\boldsymbol{\theta}}, C(\epsilon) H_1^{-1})$$
(18)

where  $C(\epsilon)$  is a scaling factor for the covariance matrix that depends only on  $\epsilon$  and vanishes as  $\epsilon \to 0$ .

To compute  $\rho_t^{par}(\pmb{\theta})$  in the neighborhood, we first need to fix a continuous embedding for t. To simplify computations, we use a generalization of what was presented in Section 4. We use the fact that the loss has the property of Equation (12) to carry out continuous embedding by weighting the TD on average instead of changing the dataset. We keep fixed the distribution of the dataset  $\rho_1^{data}$  of Equation (3) so that any dataset sampled can be partitioned into  $D_T$ , sampled from  $\rho_T^{data}$ , and its complement  $D_{NT}$  (non-target), which is sampled from  $\rho_0^{data}$ . Then, we redefine the loss function as follows

$$\mathcal{L}_{t}(\boldsymbol{\theta}; D) = \frac{N_{NT} \mathcal{L}(\boldsymbol{\theta}; D_{NT}) + t N_{T} \mathcal{L}(\boldsymbol{\theta}; D_{T})}{N}$$
(19)

where  $N_{NT}$  and  $N_{T}$  are the numbers of non-target and target samples, respectively. Notice that  $\mathcal{L}_{1}(\boldsymbol{\theta};D) = \mathcal{L}(\boldsymbol{\theta};D)$  and  $\mathcal{L}_{0}(\boldsymbol{\theta};D) = \mathcal{L}(\boldsymbol{\theta};D_{NT})$  so that in the two extremes, one recovers the original loss if the distribution of the underlying dataset was  $\rho_{1}^{data}$  and  $\rho_{0}^{data}$ , respectively. By linearity, it follows that the Hessian of the weighted loss evaluated at  $\tilde{\boldsymbol{\theta}}$  is

$$H_t = \frac{N_{NT} H(\tilde{D}_{NT}) + t N_T H(\tilde{D}_T)}{N}$$
(20)

Finally, in a small neighborhood of t = 1, we approximate  $\mathcal{T}_t(\tilde{D}) \approx \mathcal{T}(\tilde{D}) = \tilde{\boldsymbol{\theta}}$  so that, by the same arguments as in the previous paragraph, we extend Equation (18) to a small neighborhood around  $\tilde{\boldsymbol{\theta}}$ 

$$\rho_t^{par}(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta}; \tilde{\boldsymbol{\theta}}, C^2(\epsilon) H_t^{-1}) \quad \text{for } t \in [1 - \delta, 1]$$
 (21)

The expression of Equation (21) requires the Hessian to be non-degenerate to be well-defined.

To allow computation, we further approximated the Hessian with its diagonal and ignored the parameters corresponding to vanishing diagonal elements, thus considering<sup>5</sup>

$$H_t = \text{diag}(h_1(t), \dots, h_{N_p}(t))$$
  $h_i > 0 \text{ for } i = 0, \dots, N_p$  (22)

where  $N_p$  is the number of network parameters after excluding those with vanishing diagonal elements. These approximations are used as working assumptions and are ultimately validated by experiments.

The final form of  $\rho_t^{par}$  is thus:

$$\rho_t^{par}(\boldsymbol{\theta}) \approx \prod_{i=1}^N \mathcal{N}(\theta_i; \tilde{\theta}_i, \sigma_i(t)) \quad \text{where} \quad \sigma_i(t) = \frac{C(\epsilon)}{\sqrt{h_i(t)}}$$
(23)

#### 4.2 Fisher Information

Starting from the results of the previous section, we derive an explicit expression for the Fisher information of Equation (9), and define the Target Information Score.

The Fisher information about the presence of the TD, in the case where it was used during the training, is:

$$I(1) = \mathbb{E}_{\rho_1^{par}(\boldsymbol{\theta})} \left[ \left( \frac{\partial \log(\rho_t^{par}(\boldsymbol{\theta}))}{\partial t} \bigg|_{t=1} \right)^2 \right]$$
 (24)

In Appendix 1, we show that substituting the expression of Equation (23) into the previous one yields

$$I(1) \propto \sum_{i=1}^{N} \left(\frac{h_i'}{h_i}\right)^2 \quad \text{with} \quad h_i' \equiv \left. \frac{\partial h_i(t)}{\partial t} \right|_{t=1} \quad h_i \equiv h_i(1)$$
 (25)

where the proportionality constant only depends on  $\epsilon$ . Then from Equation (20) we find:

$$h_i(t) = \frac{1}{N} \left[ N_{NT} \frac{\partial^2 \mathcal{L}(\tilde{D}_{NT})}{\partial \theta_i^2} + t N_T \frac{\partial^2 \mathcal{L}(\tilde{D}_T)}{\partial \theta_i^2} \right]$$
 (26)

So that:

$$h_i = \frac{\partial^2 \mathcal{L}(\tilde{D})}{\partial \theta_i^2} \qquad h_i' = \frac{N_T}{N} \frac{\partial^2 \mathcal{L}(\tilde{D}_T)}{\partial \theta_i^2}$$
 (27)

The total Fisher information of Equation (24) is a sum over the network parameters; we define the Target Information Score as

$$TIS(\theta_i) \equiv \left(\frac{\frac{\partial^2 \mathcal{L}(\tilde{D}_T)}{\partial \theta_i^2}}{\frac{\partial^2 \mathcal{L}(\tilde{D})}{\partial \theta_i^2}}\right)^2 \propto \left(\frac{h_i'}{h_i}\right)^2$$
(28)

where the proportionality constant only depends on  $\epsilon$ , N,  $N_T$  and is therefore independent of i.

# 5 Proposed Unlearning Algorithm

The framework presented in Section 4 is used to implement an FU algorithm.

We consider n clients with datasets  $D_c$  for  $c=1,\ldots,n$ , and assume that a model  $\boldsymbol{\theta}$  has been trained cooperatively. Suppose that the server receives an unlearning request that requires it to forget a certain *target dataset*  $D_T$ . Interestingly, we do not need  $D_T$  to be composed of samples from the same client; instead, we assume it to be a generic subset of the whole training dataset; thus, each client dataset will be decomposed into  $D_c = D_c^{(f)} + D_c^{(r)}$ , where  $D_c^{(f)} = D_c \cap D_T$  and  $D_c^{(r)} = D_c \setminus D_c^{(f)}$ . This allows us to address the Sample Unlearning task as described in Section 3.1.

All clients are assumed to be collaborative and are required to compute the Hessian diagonal of the loss averaged over the samples  $D_c^{(f)}$  and  $D_c^{(r)}$ , respectively, thus obtaining

$$\boldsymbol{h}_{c}^{(f)} = \frac{\partial^{2} \mathcal{L}(D_{c}^{(f)})}{\partial \boldsymbol{\theta}^{2}} \quad \boldsymbol{h}_{c}^{(r)} = \frac{\partial^{2} \mathcal{L}(D_{c}^{(r)})}{\partial \boldsymbol{\theta}^{2}} \quad \text{for } c = 1, \dots, n$$
(29)

<sup>&</sup>lt;sup>5</sup>From now on, all the derivations in the network parameter space are intended to be evaluated at the trained parameters  $\theta$ 

These are then collected and aggregated by the server, allowing it to compute the expressions of Equation (27) as follows:

$$\boldsymbol{h} = \frac{1}{N} \sum_{c=1}^{n} \left( N_c^{(f)} \boldsymbol{h}_c^{(f)} + N_c^{(r)} \boldsymbol{h}_c^{(r)} \right) \quad \boldsymbol{h}' = \frac{1}{N_T} \sum_{c=1}^{n} N_c^{(f)} \boldsymbol{h}_c^{(f)}$$
(30)

where  $N_c^{(f)}$  and  $N_c^{(r)}$  are the numbers of target and non-target samples in  $D_c$ , respectively. Finally, the TIS for each parameter is calculated as in Equation (28).

Then, the parameters deemed "the most informative" for  $D_T$  are identified and reset: we introduce the hyperparameter removal percentage  $\gamma_{removal}$ , quantifying the percentage of information to be removed. For each layer, we compute its total TIS by adding up the TIS of every parameter within it. We then select the smallest set of parameters whose total TIS amounts to  $\gamma_{removal}$  percent of the layer TIS. These parameters are reset to their initial (pre-training) values. The procedure can also be visualized as, for each layer, ordering the parameters in descending value of TIS and then resetting them one by one until  $\gamma_{removal}$  percent of the total layer TIS has been reset.

## 5.1 Time and Space Overheads

#### **Time Overheads**

The core of this routine involves iterating once through the complete dataset of all participating clients (i.e., processing a total of  $B_{\rm total}$  batches across all clients). For each batch, the diagonal of the Hessian matrix (or Generalized Gauss-Newton matrix, GGN) is computed. Crucially, leveraging the BackPACK library [23] for this step, the calculation for each batch is highly efficient. The time complexity to calculate the diagonal Hessian (or GGN) for a single batch using BackPACK is approximately  $k \cdot C_{bwd}$ , where  $C_{bwd}$  is the cost of a standard gradient backpropagation pass for that batch, and k is a small constant factor. Therefore, this dominant step has a complexity of approximately  $O(B_{\rm total} \cdot C_{bwd})$ .

While this step is efficient, its cost can be further reduced by employing an approximation. In particular, we can leverage BackPACK to compute a Monte-Carlo approximation of the Generalized Gauss-Newton (GGN) matrix diagonal. This approach significantly lowers the time complexity, making the unlearning process more scalable at the cost of a small loss in precision.

Subsequent steps in the routine primarily involve  $O(L \cdot N_c \cdot M_p)$  operations, where L is the number of parameter blocks (e.g., layers),  $N_c$  is the number of clients, and  $M_p$  is the cost of elementwise operations on tensors of a parameter block's size.

Once the information scores are computed, the unlearning process itself involves the following.

- Identifying parameters to reset based on these precomputed scores (negligible time complexity, typically involving thresholding).
- 2. Reset these identified parameters (also in negligible time).

This design ensures that the most computationally intensive part (Hessian diagonal computation) can be performed efficiently on a single basis as soon as the training ends. At the same time, the unlearning execution upon request remains ready.

## **Space Overheads**

In terms of space overhead, the benefit of our model is related to the fact that it does not require the saving of any information during training; nonetheless, just after training (or as soon as the unlearning request is received), the algorithm needs the computation of the elements previously described and the saving of these.

Each client will have to compute the diagonal of the Hessian and store it until the server requires it to proceed with the unlearning. This element has approximately a space complexity equal to O(P) where P is the total number of parameters.

Other elements required during training have negligible overhead space.

## 6 Evaluation

### 6.1 Experimental setup

#### **Dataset and models**

We designed our experiments on three standard datasets: MNIST [24], FashionMNIST [25] and CIFAR-10 [26]. Each dataset consists of 10 distinct classes; we consider a scenario involving five clients participating in the training process. The datasets are split into training and test sets (with a ratio of 70/30, respectively). Client-specific datasets are constructed by allocating samples from the training set according to two distinct label distribution strategies. The strategies hereby presented have been implemented following the Sample Unlearning and the Class Unlearning objectives, as outlined in Section 3:

- Random Setting. In this setting, the original dataset is randomly partitioned into 10 client subsets, corresponding to the scenario in which clients have independent and identically distributed (IID) datasets.
- Preferential Setting (Categorical Unlearning Simulation). In this setting, we model a scenario of highly heterogeneous (non-IID) distributions of the client's datasets. For each dataset, the samples of five classes were distributed evenly among the five clients, while for each of the remaining five classes, all the samples were assigned to a different client. Therefore, all clients share five classes, but each one also brings a class exclusive to that client. This setup corresponds to scenarios where individual clients hold some specialized, near-unique knowledge; in fact, unlearning any client also requires completely unlearning its preferential class.
- Dirichlet Distribution Setting. In this setting, we partition data using a Dirichlet distribution. This method is widely used to simulate heterogeneous and imbalanced client datasets. We specifically use a concentration parameter of  $\alpha = 0.8$  to allocate training samples to the five clients. A low value of  $\alpha$  corresponds to high data heterogeneity. Our chosen value ensures that each client's local data distribution is unique and imbalanced, yet it is probable that every client still holds samples from most classes. This setup effectively simulates a moderate level of data heterogeneity, placing it between the perfect IID case and the disjoint, near-unique knowledge of the Preferential Setting.

#### 6.1.1 Unlearning Scenario: Backdoor Attack

As an additional scenario in which our proposed FU algorithm can be applied, we considered a backdoor attack [27]. This scenario is a practical and challenging test for data forgetting, as the unlearning algorithm must remove some deliberately inserted malicious behaviour while restoring the performance.

The target client acts maliciously by inserting a backdoor trigger into its training samples. Specifically, we use the Adversarial Robustness Toolbox (ART) [28] to insert a 3x3 white pixel pattern in the corner of the images. For every sample now containing this trigger, the target client changes its label to a single predetermined target class, regardless of its original class. When these poisoned data are used to train the global model, the model learns to associate the trigger pattern with the target label. In this case, we inserted the backdoor in the MNIST dataset, making the model misclassify every such sample as the number "9", following [17]. A successful unlearning process must therefore not only neutralize the influence of this backdoor but also ensure a correct identification of the image with the correct (non-poisoned) label.

#### **6.1.2** Training details

Regarding the models used for the experiments, we used a Convolutional Neural Network (CNN) for the tests on MNIST, composed of two convolutional layers, and we used ResNet18 ([29]) for the other two datasets.

The models were trained for 40 epochs using the ResNet18 architecture and for six epochs with CNN. Training used SGD as an optimizer, with a learning rate  $\eta_{lr}$  = 0.01 and a momentum of 0.9. The cross-entropy loss function was used, and a batch size of 32 was used consistently in all experiments.

All experimental evaluations were conducted on a server equipped with an NVIDIA L40S GPU of 48 GB and an AMD 32-core CPU with 768 GB of RAM.

## **6.2** Evaluation Metrics

We assess the efficacy of our proposed unlearning algorithm considering three main dimensions [10]: the *efficiency* of the algorithm, the *performance* of the unlearned model, and the *forgetting* of the target dataset. The metrics used to

evaluate these aspects involve the comparison of: (1) the unlearned model  $\theta^u$ , (2) the model before unlearning  $\theta^i$ , (3) the model retrained from scratch without the target dataset  $\theta^r$  (gold standard).

In addition to comparing our  $\theta^u$  with the gold standard  $\theta^r$ , we introduce a more direct baseline to specifically isolate the contribution of our parameter selection criterion. Our method utilizes a removal percentage ( $\gamma_{\text{removal}}$ ) to identify and reset a certain number of parameters in each layer based on their descending TIS, as described in Section 4. To create a fair comparison, we design a **Random Reset** baseline. First, we measure the exact count of parameters that our primary method resets in each layer. Then, the baseline resets the **identical number of parameters per layer**, but the selection is performed **uniformly at random**, without considering any information score. By comparing our model's performance against this baseline, we can effectively measure the added value of our information-theoretic approach. We hypothesize that our method will outperform the Random Reset baseline across both performance and forgetting metrics, proving that *which* parameters are reset is as critical as *how many*.

## **6.2.1** Efficiency Metrics

Efficiency consists of the time and computational complexity of the unlearning process. To assess efficiency in our experiments, we use the **Recovery Time Ratio** (**RTR**). This is defined as the ratio of the time taken by our unlearning algorithm to the time required to retrain  $\theta^r$  from scratch:

$$RTR = \frac{\text{Unlearning Time (s)}}{\text{Retraining Time (s)}}$$
(31)

#### **6.2.2** Performance Metrics

Model performance refers to the retained performance of the model after unlearning has occurred. In an optimal scenario,  $\theta^u$  would achieve test accuracy identical to that of  $\theta^r$ . The difference between the accuracy of  $\theta^i$  and the accuracy of  $\theta^r$  quantifies the performance impact attributable solely to the removal of the unlearned data, independent of the efficiency of the unlearning algorithm.

To further facilitate comparison, we also report a **Normalized Test Accuracy (NTA)**. This metric expresses the test accuracy of the unlearned model ( $\theta^u$ ) as a fraction of the benchmark retrained model's ( $\theta^r$ ) test accuracy:

Normalized Test Accuracy = 
$$\frac{\text{Accuracy}(\boldsymbol{\theta^u}, D_{\text{test}})}{\text{Accuracy}(\boldsymbol{\theta^r}, D_{\text{test}})}$$
(32)

A value approaching 1.0 for this normalized accuracy indicates that the unlearned model has successfully recovered the performance level of an ideally retrained model. NTA can also exceed 1—especially with small forget/retain sets or under non-IID FL—because the constrained update space induces pruning-like regularization, improving generalization over a full retrain.

## **6.2.3** Forgetting Metrics

Forgetting metrics quantify the removal of the influence of the target data ( $D_{\mathrm{target}}$ ) from the unlearned model.

The primary measure for assessing this, especially in non-IID client dataset scenarios or for categorical unlearning, is the **Target Dataset Accuracy**. We compare this accuracy across  $\theta^u$ ,  $\theta^r$ , and  $\theta^i$ . For different client datasets, we usually expect Accuracy( $\theta^i$ ,  $D_T$ ) > Accuracy( $\theta^r$ ,  $D_T$ ), as  $\theta^i$  was trained on  $D_T$  while  $\theta^r$  was not. The optimal scenario is for  $\theta^u$  to achieve the accuracy of the target identical to  $\theta^r$ . As client datasets become more IID, the difference between Accuracy( $\theta^i$ ,  $D_T$ ) and Accuracy( $\theta^r$ ,  $D_T$ ) may decrease, making this absolute measure less discriminative for certain settings of unlearning.

To provide a comparative measure of how effectively the unlearning process bridges the gap in knowledge between the original and the ideally retrained model, we introduce a **Normalized Forgetting Score** (**NFS**) based on target accuracy:

$$NFS_{TargetAcc} = \frac{Accuracy(\boldsymbol{\theta^i}, D_T) - Accuracy(\boldsymbol{\theta^u}, D_T)}{Accuracy(\boldsymbol{\theta^i}, D_T) - Accuracy(\boldsymbol{\theta^r}, D_T)}$$
(33)

An NFS value approaching 1.0 indicates that the unlearned model  $\theta^u$ 's performance on the target data is very close to that of the ideally retrained model  $\theta^r$ , relative to the initial 'vulnerability' demonstrated by  $\theta^i$ . A value near zero would suggest  $\theta^u$  still behaves much like  $\theta^i$  on the target data.

In scenarios with IID client data, we also employ Membership Inference Attacks (MIA), as described in [1]. We measure the **MIA Accuracy**, which reflects the attacker's ability to distinguish target dataset samples from test dataset samples based on model outputs. Ideally, MIA\_Acc( $\theta^r$ )  $\approx 0.5$  (random guessing), while MIA\_Acc( $\theta^i$ ) may be

significantly higher. The goal is for MIA\_Acc( $\theta^u$ ) to also approach 0.5. Analogous to the target accuracy, we can define a **Normalized Forgetting Score based on MIA Accuracy (NFS<sub>MIA</sub>)**:

$$NFS_{MIA} = \frac{MIA\_Acc(\boldsymbol{\theta^i}) - MIA\_Acc(\boldsymbol{\theta^u})}{MIA\_Acc(\boldsymbol{\theta^i}) - MIA\_Acc(\boldsymbol{\theta^r})}$$
(34)

Again, a value approaching 1.0 signifies that the unlearned model's vulnerability to MIA has been reduced to the level of the ideally retrained model, relative to the original model's vulnerability.

## **6.2.4** Backdoor Accuracy

For the backdoor scenario, we introduce two specific metrics to provide a complete picture of both the attack's potency and the unlearning's effectiveness:

- 1. Attack Success Rate (ASR): This metric measures the effectiveness of the backdoor attack. It is calculated as the accuracy of the model on test images containing the trigger, where the labels are set to the poisoned target label (e.g., class '9'). A high ASR in the initially trained model ( $\theta^z$ ) indicates a successful attack. The goal of unlearning is to reduce the ASR of the unlearned model ( $\theta^u$ ) to near-random guessing (e.g., 10% for a 10-class problem).
- 2. **Backdoor Accuracy:** We use this metric to evaluate the model's ability to correctly classify triggered examples according to their original true labels. A low Backdoor Accuracy for the  $\theta^z$  model confirms that it is susceptible to the trigger. A high Backdoor Accuracy for the unlearned  $\theta^u$  model demonstrates that it has successfully ignored the trigger and recovered the samples' true class, signifying successful neutralization of the backdoor  $\theta^z$ .

## 7 Results

In this section, we present the empirical evaluation of our proposed federated unlearning framework. The core of our analysis is a direct comparison of our information-theoretic method (Reset) against a Random Reset baseline and the gold-standard Retrain from scratch benchmark. We aim to demonstrate that our targeted approach provides a superior trade-off between model performance and privacy (forgetting the target data).

We experimentally assess the proposed framework by repeating each test 20 times to allow for more precise statistics. The results are presented, indicating the mean and standard deviation after the 20 tests.

## 7.1 Privacy Performance Trade-Off

To visualize the effectiveness of our unlearning strategy, we plot the trade-off between model performance and privacy across a range of unlearning percentages. Figure 1 presents this analysis for the CIFAR-10 and FashionMNIST datasets under all the different configurations: Random Class (IID-like), Preferential Class (non-IID) settings, and the Dirichlet Distribution.

The optimal outcome is represented by the "Retrain from scratch" point at (1, 1), and an unlearning method is considered superior if its curve lies closer to it. In the preferential and Dirichlet settings, our proposed Reset method (blue curve) consistently achieves a more favourable privacy-performance trade-off than the Random Reset baseline (orange curve), while in the random setting, the two have performances compatible with each other.

## 7.2 Analysis of Unlearning Scenarios

The trade-offs visualized in the figures are supported by the quantitative data presented in Tables 1 through 5. The  $\gamma_{\text{removal}}$  was chosen for each test as the one corresponding to the lowest Euclidean distance of (NTA, NFS) from (1,1).

# 7.2.1 Preferential Class (Categorical Unlearning)

In the non-IID setting where clients hold unique classes, Table 1 shows the Reset model achieves a Test Accuracy of 60.15% for CIFAR-10 and 69.83% for FashionMNIST, resulting in NTA values of 0.82 and 0.84, respectively. Forgetting metrics in Table 2 show a reduction in Target Accuracy to 10.51% (CIFAR-10) and 12.32% (FashionMNIST). This corresponds to NFS values of 0.85 and 0.88. As reported in Table 3, these results were obtained by resetting approximately 4.27% (CIFAR-10) and 6.99% (FashionMNIST) of the total model parameters.

<sup>&</sup>lt;sup>6</sup>Backdoor Accuracy is computed exclusively on triggered images whose original label was not the target class (class 9), thereby testing only the model's capacity to ignore the malicious pattern.

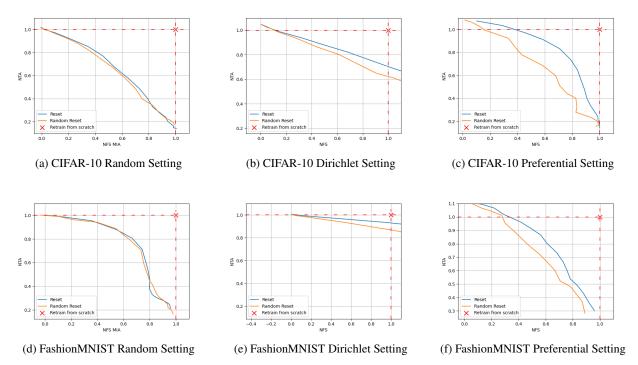


Figure 1: The privacy-performance trade-off across multiple datasets and data distribution settings. Our proposed method (**Reset**, blue curve) is compared against the **Random Reset** baseline (orange curve). The y-axis shows the NTA metric, and the x-axis shows the  $NFS_{MIA}$  score. The ideal point, representing retraining from scratch, is the red 'x' at (1, 1).

Performance Dataset Distribution **Test Accuracy** (%) ↑ NTA ↑ Original Benchmark Reset Reset CIFAR-10 Preferential  $79.66 \pm 0.54$  $73.21 \pm 0.42$  $60.15 \pm 6.09$  $0.82 \pm 0.08$ **FMNIST**  $93.48 \pm 0.24$  $83.43 \pm 0.20$  $69.83 \pm 13.36$  $0.84 \pm 0.16$  $0.73 \pm 0.14$ CIFAR-10 Dirichlet  $74.36 \pm 1.17$  $70.65 \pm 0.88$  $51.23 \pm 9.44$ **FMNIST**  $(\alpha = 0.8)$  $91.82 \pm 1.25$  $91.19 \pm 0.93$  $85.29 \pm 2.89$  $0.94 \pm 0.03$ CIFAR-10  $79.71 \pm 0.49$  $77.82 \pm 0.75$  $48.94 \pm 5.34$  $0.63 \pm 0.07$ Random **FMNIST**  $93.53 \pm 0.24$  $76.05 \pm 6.16$  $93.17 \pm 0.19$  $0.82 \pm 0.07$ 

Table 1: Performance Metrics

#### 7.2.2 Dirichlet Distribution Setting

For the moderate non-IID case simulated with a Dirichlet distribution, the Reset model produces an NTA of 0.73 on CIFAR-10 and a strong 0.94 on FashionMNIST (Table 1). The privacy metrics in Table 2 report NFS scores of 0.90 (CIFAR-10) and 0.93 (FashionMNIST) based on Target Accuracy. The parameter reset percentage for these experiments, shown in Table 3, was 5.13% for CIFAR-10 and 4.56% for FashionMNIST.

## 7.2.3 Random Class (Client Unlearning)

In the IID client unlearning scenario, Table 1 reports that the Reset model's performance corresponds to NTA values of 0.63 for CIFAR-10 and 0.82 for FashionMNIST. The privacy evaluation for this setting is based on Membership Inference Attacks (MIA), as shown in Table 2. The Reset model yields an MIA Accuracy of 55.01% (CIFAR-10) and 55.92% (FashionMNIST), which results in NFS<sub>MIA</sub> scores of 0.65 and 0.88. Table 3 indicates that this was achieved by resetting 5.54% and 6.82% of the parameters for CIFAR-10 and FashionMNIST, respectively.

Table 2: Privacy Metrics

Privacy							
Dataset	Distribution	Tar	NFS ↑				
		Original	Benchmark	Reset	Reset		
CIFAR-10 FMNIST CIFAR-10 FMNIST	Preferential  Dirichlet $(\alpha = 0.8)$	$69.47 \pm 3.16$ $98.73 \pm 0.24$ $99.94 \pm 0.07$ $100.00 \pm 0.01$	$0.00 \pm 0.00$ $0.00 \pm 0.00$ $62.78 \pm 5.05$ $89.80 \pm 1.35$	$10.51 \pm 9.63$ $12.32 \pm 15.19$ $66.71 \pm 6.19$ $90.41 \pm 3.58$	$0.85 \pm 0.13$ $0.88 \pm 0.15$ $0.90 \pm 0.16$ $0.93 \pm 0.28$		
		MIA Accuracy (%)			NFS <sub>MIA</sub> ↑		
		Original	Benchmark	Reset	Reset		
CIFAR-10 FMNIST	Random	$64.42 \pm 0.45$ $59.92 \pm 0.28$	$49.83 \pm 0.49 \\ 55.36 \pm 0.08$	$55.01 \pm 1.24$ $55.92 \pm 0.51$	$\begin{array}{c c} 0.65 \pm 0.09 \\ 0.88 \pm 0.12 \end{array}$		

Table 3: Removal parameters used in the experiments and corresponding percentage of parameters reset

Distribution	$\gamma_{ m removal}$	Reset parameters (%)
Preferential	$29.50 \pm 13.87$	$4.27 \pm 1.22$
	$54.00 \pm 21.01$	$6.99 \pm 4.41$
Dirichlet	$43.75 \pm 19.42$	$5.13 \pm 1.62$
$(\alpha = 0.8)$	$25.25 \pm 13.46$	$4.56 \pm 2.48$
Random	$39.50 \pm 12.24$ $36.75 \pm 24.76$	$5.54 \pm 1.32$ $6.82 \pm 4.21$
	Preferential  Dirichlet $(\alpha = 0.8)$	$\begin{array}{c} \text{Preferential} & 29.50 \pm 13.87 \\ & 54.00 \pm 21.01 \\ \text{Dirichlet} & 43.75 \pm 19.42 \\ (\alpha = 0.8) & 25.25 \pm 13.46 \\ \text{Random} & 39.50 \pm 12.24 \\ \end{array}$

#### 7.2.4 Backdoor Attack

Table 4 presents the results of the backdoor neutralization test on the MNIST dataset; the attack has been performed following the discussion in Section 6.1.1. The Reset method reduces the Attack Success Rate (ASR) from 100% to 10.14% and restores the Backdoor Accuracy from 0% to 57.41%. It is worth highlighting that the results for our Reset method are characterized by relatively high variance across different runs; still, the results show a clear improvement with respect to the original model.

## 8 Discussion

The experimental results provide insight into the capabilities of the theoretical framework under different unlearning settings.

The central takeaway from our experiments, visualized in the trade-off curves of Figure 1, is the correct functioning of our TIS-based **Reset** method over the **Random Reset** baseline in all the different scenarios except for the IID distribution. The **Reset** curve consistently lies closer to the ideal (1, 1) point, signifying that for any given level of performance (NTA), our method achieves a higher degree of forgetting (higher NFS). This finding validates our core hypothesis: among all the possible parameters, only a subset of these is fundamentally more important. A targeted approach based on an information-theoretic score is significantly more efficient at erasing data influence than a naive, random perturbation.

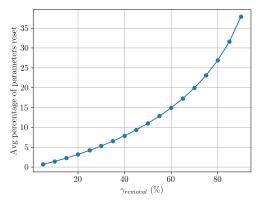
The advantage of our method is most pronounced in the non-IID settings (Preferential Class and Dirichlet). In these scenarios, clients possess specialized or unique data. We theorize that this specialized knowledge is encoded in a more concentrated and identifiable subset of the model's parameters. Our TIS-based method is more effective at pinpointing

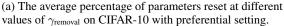
Table 4: Backdoor attack metrics on MNIST

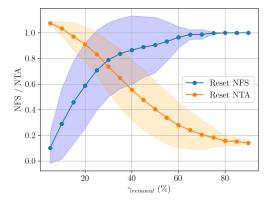
Metric	Original	Benchmark	Reset
ASR	$99.99 \pm 0.02$	$0.11 \pm 0.08$	$10.14 \pm 10.35 57.41 \pm 21.33$
Backdoor Accuracy	$0.01 \pm 0.02$	$99.14 \pm 0.16$	

Table 5: Efficiency metrics

Dataset	Retrain time (s)	Reset time (s)	RTR↓
CIFAR-10	651	191	0.29
FMNIST	778	229	0.30







(b) NFS and NTA scores measured for different values of  $\gamma_{\text{removal}}$  on CIFAR-10 with preferential setting. Shaded areas indicate one standard deviation from the average.

this subset of parameters, allowing for the precise and effective removal of the target information with minimal collateral damage to the model's general performance.

Conversely, in the IID Random Class setting, the performance gap between the **Reset** and **Random Reset** methods narrows, as seen in Figures 1a and 1d. We argue that, in an IID environment, a client's data is more redundant and similar to the overall data distribution. Consequently, the information is likely stored more diffusely across the network and is thus more difficult to target.

Similarly to the case of non-IID settings, in the backdoor attack scenario, a client possesses some highly unique features in the data. Our method can greatly reduce the efficacy of the backdoor attack and partially restore the normal performance of the model on images with the backdoor pattern. This suggests that TIS can pinpoint the parameters related to the backdoor pattern.

In a real case scenario, one of the most important elements is the efficiency of the algorithm with respect to retraining from scratch. Table 5 shows that our **Reset** method achieves effective unlearning more than three times faster than full retraining, as seen in the RTR metric. This computational advantage makes our approach a viable alternative for real-world federated systems, where the cost of retraining from scratch is often prohibitive.

Finally, to highlight the ability of the model and how it can differ among various runs, it is interesting to observe the non-uniform distribution of information within the network's parameters. Table 3 provides context for these results. It demonstrates that effective unlearning is achieved by resetting a small fraction of the total model parameters, typically between 4% and 7%.

## 8.1 Sensitivity Analysis

To further understand the mechanics of our unlearning framework, in Figure 2a we analyzed the relationship between  $\gamma_{\text{removal}}$  and the actual percentage of model parameters that are reset as a result. A key observation is that the relationship is markedly non-linear and convex. This indicates that the target information, as quantified by our Target Information Score, is highly concentrated within a relatively small subset of parameters. For example, removing the first 50% of the target information requires resetting only about 11% of the model's total parameters. In contrast, to remove the subsequent 40% of information (from 50% to 90%), a much larger fraction of parameters (approximately an additional 28%) must be reset.

Further, Figure 2b shows how the NTA and NFS metrics are dependent on  $\gamma_{\text{removal}}$ . We observe that increasing  $\gamma_{\text{removal}}$  increases the measured forgetting score but has the side effect of degrading test accuracy. A key observation

is the asymmetry in their rates of change. Initially, the NFS improves sharply, increasing by approximately 0.8 for  $\gamma_{\text{removal}} = 30\%$ , while the NTA only decreased by approximately 0.2.

## 9 Conclusion

In this paper, we have addressed the challenge of efficient and effective data erasure in federated learning by introducing a principled, information-theoretic framework for federated unlearning. Our approach models the deletion of a target dataset as a parameter-estimation problem; by leveraging second-order statistics, we can quantify the contribution of each parameter to data memorization and selectively reset only the most influential parameters. This targeted strategy yields privacy outcomes and performance levels that closely approximate those of complete retraining while being three times more efficient than retraining from scratch.

Our empirical evaluations demonstrate the optimal results of our methods and the importance of identifying the correct subset of parameters to perform a focused reset; the comparison is performed against a Random Reset of the same percentage of parameters to demonstrate the effectiveness of our method. The *Reset* of the parameters found with the highest TIS score leads to significant metrics related to the forgotten data, confirming the neutralization of influential parameters. This principle extends beyond privacy-motivated data removal to critical security applications; our experiments show that the framework can neutralize backdoor attacks by identifying and resetting the parameters corrupted by malicious training data, thus restoring model integrity. To perform such a reset, the removal percentage serves as a key hyperparameter to navigate optimal performance based on specific application needs, as we have shown in Section 8.1.

Furthermore, our framework exhibits several desirable properties for real-world deployment: it is both data- and architecture-agnostic (the algorithm works with arbitrary datasets) and, significantly, it operates without the need to store historical model updates or extensive checkpoint snapshots. It is also, based on our current experiments, three times more efficient than retraining from scratch. These characteristics make it a well-suited and scalable solution for implementing "right-to-be-forgotten" mandates in federated systems.

#### **Future Work**

Future research will focus on key extensions to our framework and on addressing its current limitations.

We argue that a limitation of our framework is our reliance on a diagonal approximation for the Hessian computation, where we ignore parameters corresponding to vanishing diagonal elements. Future work could explore the utility of richer, potentially more precise second-order information beyond Hessian diagonals (e.g., block-diagonal or approximations of the full Hessian). We also aim to investigate adaptive hyperparameter tuning for the unlearning percentages and analyze the resulting privacy-performance trade-offs.

Future work will also focus on validating our framework on real-world datasets. Scaling our method from benchmark datasets to real-world scenarios introduces significant challenges, including greater data scale, higher complexity, and more pronounced client heterogeneity. Investigating the algorithm's performance under these conditions is a crucial next step for assessing its practical viability.

Finally, in this work, the scenario of clients entering and leaving the training at different times was not explored, and we believe it could be an interesting topic for future research.

# **Appendix**

## Appendix 1

We start by computing

$$\frac{\partial}{\partial t} \log(\rho_t^{par}(\boldsymbol{\theta})) = \sum_{i=1}^N \frac{\partial}{\partial t} \log(\mathcal{N}(\theta_i; \tilde{\theta}_i, \sigma_i(t)))$$
(35)

$$= \sum_{i=1}^{N} \left[ \frac{\partial}{\partial \sigma_i} \log(\mathcal{N}(\theta_i; \tilde{\theta}_i, \sigma_i(t))) \right] \frac{\partial \sigma_i}{\partial t}$$
 (36)

A simple computation shows that

$$\frac{\partial}{\partial \sigma_i} \log(\mathcal{N}(\theta_i; \tilde{\theta}_i, \sigma_i(t))) \tag{37}$$

$$= \frac{1}{\mathcal{N}(\theta_i; \tilde{\theta}_i, \sigma_i(t))} \frac{\partial}{\partial \sigma_i} \left( \frac{1}{\sqrt{2\pi} \sigma_i(t)} \exp\left[ \frac{(\theta_i - \tilde{\theta}_i)^2}{2\sigma_i(t)^2} \right] \right)$$
(38)

$$= -\frac{1}{\sigma_i(t)} + \frac{(\theta_i - \tilde{\theta}_i)^2}{\sigma_i(t)^3}$$
(39)

and

$$\frac{\partial \sigma_i}{\partial t} = -\frac{1}{2}C(\epsilon) \left(h_i(t)\right)^{-\frac{3}{2}} \frac{\partial h_i(t)}{\partial t} \tag{40}$$

$$= -\frac{1}{2} \frac{\sigma_i(t)^3}{C(\epsilon)^2} \frac{\partial h_i(t)}{\partial t}$$
(41)

so that we find

$$\frac{\partial}{\partial t} \log(\rho_t^{par}(\boldsymbol{\theta})) = \frac{1}{2C(\epsilon)^2} \sum_{i=1}^N \left(\sigma_i(t)^2 - (\theta_i - \tilde{\theta}_i)^2\right) \frac{\partial h_i(t)}{\partial t}$$
(42)

Then we get

$$\left(\frac{\partial \log(\rho_t^{par}(\boldsymbol{\theta}))}{\partial t}\Big|_{t=1}\right)^2$$

$$= \frac{1}{4C^4(\epsilon)} \left\{ \sum_{i\neq j} h_i' h_j' \left[\sigma_i(1)^2 \sigma_j(1)^2 - \sigma_i(1)^2 (\theta_j - \tilde{\theta}_j)^2\right] \right\}$$
(43)

$$-(\theta_i - \tilde{\theta}_i)^2 \sigma_j (1)^2 + (\theta_i - \tilde{\theta}_i)^2 (\theta_j - \tilde{\theta}_j)^2 \right]$$

$$+\sum_{i=1}^{N} \left[ \sigma_i (1)^4 - 2(\theta_i - \tilde{\theta}_i)^2 \sigma_i (1)^2 + (\theta_i - \tilde{\theta}_i)^4 \right]$$
 (44)

where we wrote  $h_i' \equiv \frac{\partial h_i(t)}{\partial t}\Big|_{t=1}$ . We can then compute the expectation value

$$I(1) = \mathbb{E}_{\rho_1^{par}(\boldsymbol{\theta})} \left[ \left( \left. \frac{\partial \log(\rho_t^{par}(\boldsymbol{\theta}))}{\partial t} \right|_{t=1} \right)^2 \right]$$
 (45)

$$= \frac{1}{2C^4(\epsilon)} \sum_{i=1}^{N} \sigma_i(1)^4 h_i^{\prime 2}$$
 (46)

$$=\frac{1}{2C^4(\epsilon)}\sum_{i=1}^N \left(\frac{h_i'}{h_i}\right)^2 \tag{47}$$

where we wrote  $h_i \equiv h_i(1)$  and we used the Gaussian moments

$$\mathbb{E}_{\rho_1^{par}(\boldsymbol{\theta})}\left[\left(\theta_i - \tilde{\theta}_i\right)^2\right] = \sigma_i(1)^2 \tag{48}$$

$$\mathbb{E}_{\rho_1^{par}(\boldsymbol{\theta})}\left[\left(\theta_i - \tilde{\theta}_i\right)^4\right] = 3\sigma_i(1)^4 \tag{49}$$

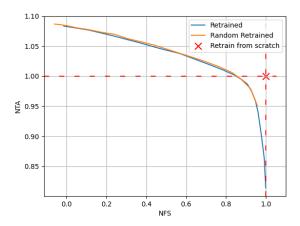


Figure 3: Privacy-performance trade-off on CIFAR-10 with preferential setting after an epoch of retrain. Our method **Retrained** (blue curve) is compared with the **Random Retrained** baseline (orange curve).

## Appendix 2

In the main body of this work, we focused on the direct impact of our information-theoretic parameter reset. However, in a practical unlearning scenario, one might apply a fine-tuning step to recover the utility of the model on the remaining data. We investigated this by performing a single epoch of federated retraining on only the reset parameters, using the TRIM module described in algorithms 1 and 2.

Figure 3 shows the privacy-utility trade-off curves for our method and the Random Reset baseline after this retraining step has been applied.

These results show the unexpected effectiveness of the Random Reset baseline when coupled with retraining. While our TIS-based method still maintains a slight edge in some cases, the performance gap between the two methods is significantly reduced compared to the reset-only scenario.

This suggests that the single-epoch fine-tuning is a dominant factor in the unlearning process. The subsequent fine-tuning efficiently adapts the model to the remaining data, masking the initial differences in the reset strategy. Consequently, what we term a "minimal retrain" is, in practice, a highly potent optimization step. However, its impact is likely magnified by the scale of the datasets and models used in our study, where a single epoch of training represents a significant update.

For this reason, this remains an open research area, where future works might bring a significant contribution through the adoption of real world datasets.

## **Algorithm 1** Efficient Targeted Retraining using the TRIM wrapper.

```
1: Module State:
 2: \mathcal{F} {The underlying (frozen) model architecture/logic.}
 3: BUFFERS {Map: ParamName → Tensor; stores the non-trainable parameter base values.}
 4: TRAINABLE_PARAMS {ParameterDict: ParamName → nn.Parameter; holds the trainable elements.}
 5: INDICES {Map: ParamName → Tensor; stores indices specifying trainable element locations.}
 6: function INITIALIZE(\Theta_{\text{base}}, M_{\text{info}})
       Input:
 7:
           \Theta_{\text{base}} {Parameters of the original, trained base model.}
 8:
 9:
           M_{\rm info} {Map {ParamName \rightarrow Tensor of indices for retraining}.}
10:
        Output: Initialized BUFFERS, TRAINABLE PARAMS, INDICES
11:
12:
13:
        Store underlying model architecture \mathcal{F}.
14:
       BUFFERS ← {}
        TRAINABLE_PARAMS ← {}
15:
       INDICES \leftarrow \{\}
16:
17:
18: for each (name, \theta) in \Theta_{\text{base}} do
19:
           \theta_{\text{fixed}} \leftarrow \theta.\text{clone}() {Copy original parameter values into a "fixed" buffer.}
20:
       if name \in M_{info} then
21:
                 indices \leftarrow M_{info}[name]
22:
                 k \leftarrow \text{length(indices)}
23:
           if k > 0 then
                    \theta_{\text{train}} \leftarrow \text{New 'nn.Parameter' of size } k \text{ Only these } k \text{ entries will be trainable.} 
24:
25:
                    Initialize \theta_{\text{train}} (e.g. with zeros or original values).
26:
                    TRAINABLE_PARAMS[name] \leftarrow \theta_{\text{train}}
27:
                    INDICES[name] \leftarrow indices
                    \theta_{\text{fixed}}[\text{indices}] \leftarrow 0 \{\text{Zero-out base values where training will occur.}\}
28:
29:
           end if
       end if
30:
31:
           Register \theta_{\text{fixed}} into BUFFERS[name] {Store as a non-trainable buffer.}
32: end for
33: end function
```

## Algorithm 2 Forward pass for the TRIM wrapper.

```
1: function FORWARD(X)
 2:
       Input: X {Input data batch.}
 3:
       Output: Model output using reconstructed parameters.
 4:
 5:
       \Theta_{\text{live}} \leftarrow \{\} {Construct current parameters for this forward pass.}
 6:
 7: for each parameter name 'name' stored in BUFFERS do
 8:
              \theta_p \leftarrow \text{BUFFERS[name]}.clone() {Start with the fixed (zeroed-out) base parameter.}
9:
       if name ∈ TRAINABLE PARAMS then
                 Retrieve indices ← INDICES[name]
10:
11:
                 Retrieve \theta_{train} \leftarrow TRAINABLE\_PARAMS[name]
                 \theta_p[\text{indices}] \leftarrow \theta_p[\text{indices}] + \theta_{\text{train}} \{\text{Add trainable values into their correct positions.}\}
12:
       end if
13:
              \Theta_{\text{live}}[\text{name}] \leftarrow \theta_p
14:
15: end for
        Y_{\text{out}} \leftarrow \mathcal{F}(X; \text{ params} = \Theta_{\text{live}}) {Run the underlying architecture using reconstructed live parameters.}
16:
        return Y_{out}
18: end function
```

## Acknowledgments

Computational resources provided by hpc-ReGAInS@DISCo,which is a MUR Department of Excellence Project within the Department of Informatics, Systems and Communication at the University of Milan-Bicocca (https://www.disco.unimib.it).

## **Author contributions**

A.B. and L.M. designed the model and the computational framework and analysed the data. A.B. and L.M. carried out the implementation. A.B. and L.M. wrote the manuscript with input from all authors. F.S. and A.M. provided critical feedback and were in charge of overall direction and planning. All authors discussed the results and commented on the manuscript.

## **Data availability**

Code for this paper is available at: https://github.com/lorenzomanini/FedUnlearn-PE

## References

- [1] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2017.
- [2] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks, 2019.
- [3] Paul Voigt and Axel Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer Cham, 01 2017.
- [4] Rik Crutzen, Gjalt-Jorn Ygram Peters, and Christopher Mondschein. Why and how we should care about the general data protection regulation. *Psychology & Health*, 34(11):1347–1357, 2019. PMID: 31111730.
- [5] California Civil Code. Assembly bill no. 1008, 2023.
- [6] Preston Bukaty. *The California Privacy Rights Act (CPRA) An implementation and compliance guide*. IT Governance Publishing, 2021.
- [7] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.
- [8] 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), pages 141–159. IEEE, 2021.
- [9] 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, 2021.
- [10] 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*, page 1–21, 2024.
- [11] Shangchao Su, Bin Li, and Xiangyang Xue. One-shot federated learning without server-side training. *Neural Networks*, 164:203–215, July 2023.
- [12] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.
- [13] Chen Wu, Sencun Zhu, and Prasenjit Mitra. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441*, 2022.
- [14] Guanhua Ye, Tong Chen, Quoc Viet Hung Nguyen, and Hongzhi Yin. Heterogeneous decentralised machine unlearning with seed model distillation. *CAAI Transactions on Intelligence Technology*, 9(3):608–619, 2024.
- [15] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.
- [16] Matthew Jagielski, Milad Nasr, Christopher Choquette-Choo, Katherine Lee, and Nicholas Carlini. Students parrot their teachers: Membership inference on model distillation, 2023.
- [17] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. Federated unlearning: How to efficiently erase a client in fl?, 2023.
- [18] Vitaly Feldman. Does learning require memorization? A short tale about a long tail. *CoRR*, abs/1906.05271, 2019.

- [19] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9304–9312, 2020.
- [20] Zhuo Ma, Yang Liu, Ximeng Liu, Jian Liu, Jianfeng Ma, and Kui Ren. Learn to forget: Machine unlearning via neuron masking. *IEEE Transactions on Dependable and Secure Computing*, 20(4):3194–3207, 2022.
- [21] Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. What is the state of neural network pruning? *CoRR*, abs/2003.03033, 2020.
- [22] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P), pages 303–319. IEEE, 2022.
- [23] Felix Dangel, Frederik Kunstner, and Philipp Hennig. Backpack: Packing more into backprop. *CoRR*, abs/1912.10985, 2019.
- [24] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [25] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.
- [26] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.
- [27] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain, 2019.
- [28] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian M. Molloy, and Ben Edwards. Adversarial robustness toolbox v1.0.0, 2019.
- [29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.