Subset-wise Sequential Machine Unlearning

Emirhan Bilgiç [1 2] **Adrian Popescu** [1 3]

Abstract

Real-world computer vision deployments often require removing the data influence of specific clients over time, not only forgetting entire classes at once. In practice, data often arrives from multiple clients, and selectively unlearning only one client's contribution is a challenging task. We formalize this setting as subset-wise sequential forgetting: unlearning requests can arrive over time, and the model must iteratively remove specified clients' influence while preserving overall performance. We propose a simple and practical pipeline to achieve this. Our method combines (i) a two-head multi task model (for class and client/subset identification) trained with a loss called the Disentanglement Loss, and (ii) parameter dampening for unlearning, guided by the diagonal Fisher Information Matrix. Experiments on CIFAR-10 and CIFAR-100 with ResNet-18 show that our approach induces targeted forgetting while maintaining generalization. Unlike aggressive pruning-based methods, it avoids large and revealing performance gaps (which indicate a specific client has been unlearned).

1. Introduction

Across AI systems, ranging from on-device models to large foundation models, there is a recurring need to remove or *forget* specific training data after deployment (e.g., consent withdrawal or right-to-be-forgotten requests). Machine unlearning seeks to remove the influence of specified data from a trained model while preserving performance on remaining data. Throughout this paper, we use the term "client" to denote a source that contributes a subset of training data, for example, an individual user in our experiments, or, in web-scale text/multimedia collections, a website or content provider. With the recent trend in LLMs, literature has emphasized LLMs (Scholten et al., 2025; Sakarvadia et al., 2025; Liu et al., 2025; 2024). On the other hand, the literature in computer vision systems is mainly concentrated on class-wise, one-class, and non-sequential forgetting (Golatkar et al., 2020; Shi et al., 2024; Foster et al., 2024; Yamashita et al., 2024). However, many deployment scenarios require multi-class and continual unlearning: client-wise forgetting over time.

[1]Université Paris-Saclay, Gif-sur-Yvette, France [2]Nokia Bell Labs, Espoo, Finland [3]CEA, LIST, Gif-sur-Yvette, France. Correspondence to: Emirhan Bilgiç <emirhan.bilgic@etu-upsaclay.fr>.

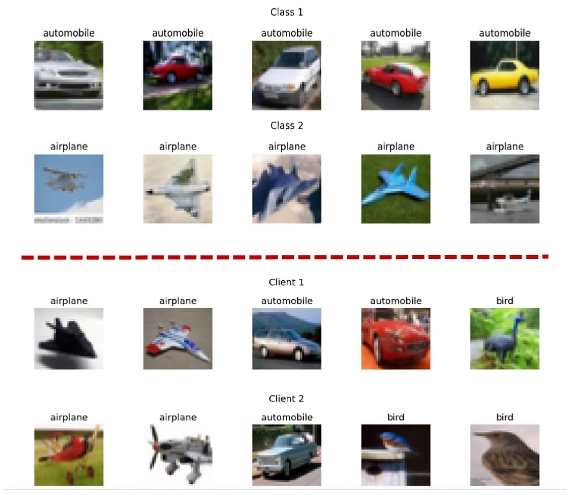
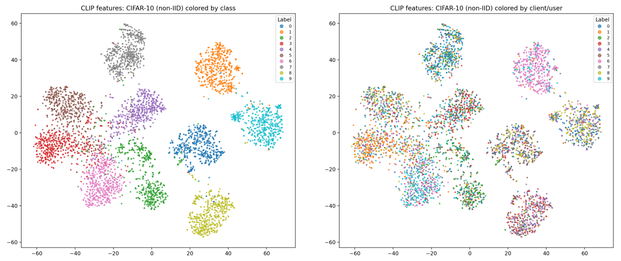


Figure 1. Problem statement: (a) class-wise setting; (b) client-wise setting in which each client contains images from multiple classes.

Problem setting. Unlearning can be instantiated along different axes and with different mechanisms. Requests may target semantic classes, data clients, or a hybrid class+client specification (e.g., "forget class k for client u"), and methods span retraining-based exact unlearning, pruning/resetting, distillation, and Fisher-guided parameter updates, the latter being the regime we investigate. Figure 1 contrasts two regimes: *(a) Class-wise forgetting* assumes the forget request targets an entire semantic class (e.g., "remove class c"). This is the dominant setup in prior work; it aligns labels and forget set and isolates the effect to a single decision boundary. By construction, classes are disjoint, so increasing loss on the target class has limited cross-talk to other classes. *(b) Client-wise (subset-wise) forgetting* targets a data client (subset) whose samples span multiple classes. Forgetting a client, therefore, requires removing the client's contribution across many class-conditional decision boundaries while preserving performance on other clients.



(a) CLIP embeddings by class (b) CLIP embeddings by client

Figure 2. Our problem is harder: t-SNE of CLIP embeddings (Radford et al., 2021) shows class clusters are more separable than client clusters, indicating client removal is more challenging than class removal due to lower feature separability.

In realistic, non-IID deployments, client distributions are heterogeneous and overlapping (cf. Figure 4), which makes client-wise forgetting substantially more challenging than class-wise forgetting.

We study the *sequential* client-wise setting: given a trained model and an ordered stream of forget requests $(c_1, c_2, \dots)$, we iteratively unlearn different clients c_t while preserving performance on the retain set and test distribution at each step t. This sequential constraint precludes retraining from scratch and penalizes methods that induce growing performance gaps across rounds.

Defining the good performance Following Chundawat et al. (2023) and Foster et al. (2024), we define good unlearning as matching a gold model which is trained from scratch on the retain set. Driving forgotten accuracy to 0 can induce revealing behavior and expose the forgetting event. Jia et al. (2023) suggests that pruning may simplify unlearning. However, in our scenario, simplification is not the goal; being close to the baseline and preserving generalization is. Membership inference attacks (MIA) inquire whether an attacker can distinguish training samples from non-training samples via a model's outputs; thus, aligning with the gold model's MIA can be considered a more faithful objective than minimizing MIA. We instantiate MIA via a simple logistic-regression attacker; see Section 5 for details.

Our approach. We address subset-wise sequential unlearning with a two-head multi task architecture (class and client heads) trained with a loss function, called Disentanglement Loss, aiming to make different client embeddings clustered together. A post-hoc, Fisher-guided unlearning method, Selective Synaptic Dampening Foster et al. (2024), is adopted to remove a target client's influence. We choose this update over aggressive pruning or distillation-heavy pipelines because it is a lightweight, retrain-free method that provides a small deviation from the gold model. Compared to prior Fisher/pruning methods, our experiments show smaller and less revealing (that a specific client has been unlearned) gaps between forgotten and retained sets, and stable performance across sequential requests. Compared to SSD without a multi task learning (MTL) pipeline, our results show competitive or better performance.

Contributions. Our contributions can be summarized as follows:

- **A realistic and challenging task design.** We formalize subset-wise, client-level unlearning under non-IID client mixtures and sequential requests, a deployment-driven setting largely absent from standard benchmarks.
- **Exploration of underexplored regimes.** We systematically study underexplored areas; *sequential* and *multi-class* forgetting in tandem, highlighting failure modes of class-wise assumptions and pruning-heavy approaches.
- **A loss shaping term for learn/unlearn.** We introduce a loss function, namely the Disentanglement Loss. We combine it with a training schedule that improves the separability of client representations, aiding both initial learning and subsequent unlearning.
- **An MTL-based unlearning pipeline with client guarantees.** We propose a two-head MTL pipeline coupled with Fisher-guided, retrain-free dampening that provides experimental guarantees of forgetting for targeted clients (via accuracy gaps and MIA), while matching or slightly exceeding no-MTL baselines that offer no user-wise guarantees.

2. Related Work

Fisher-based unlearning includes Fisher Forgetting (Golatkar et al., 2020), DeepClean (Shi et al., 2024), Selective Synaptic Dampening (SSD) (Foster et al., 2024), and Unlearning with Mnemonic Code (Yamashita et al., 2024). Some other recent noteworthy works include (Jia et al., 2023) and (Kumaravelu et al., 2024).

Full Fisher Information. Golatkar et al. (2020) (using the dense FIM) has been explored but is computationally expensive and memory-intensive; most practical methods, including ours, use the diagonal FIM as a tractable curvature proxy with strong empirical performance.

DeepClean-style pruning. Shi et al. (2024) propose resetting privacy-sensitive weights using the Fisher diagonal. A pruning approach identifies *forget-sensitive* weights via specialization ratios (e.g., coordinates where $F_{\mathcal{D}_f}/F_{\mathcal{D}}$ exceeds a threshold), prunes them, and then fine-tunes on the retain set. While effective at suppressing the forget set, this often induces large gaps between forgotten and non-forgotten

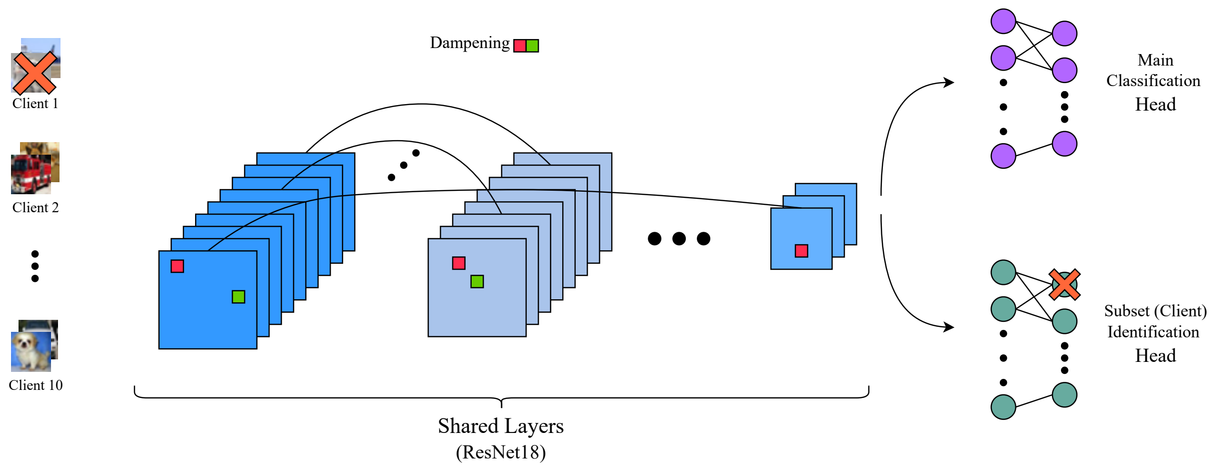


Figure 3. Illustration of the proposed sequential subset-wise unlearning pipeline. The network first learns from all clients. The network has two output heads: (i) the main classification head for predicting class labels, and (ii) the subset (client) identification head for distinguishing between different clients. The process consists of four stages: (1) Training: the model is trained on both tasks with Cross Entropy and Disentanglement Loss; (2) Dampening: SSD is applied to unlearn the features most specific to the forget client; (3) Pruning: the corresponding output neuron of the forgotten client in the subset-ID head is pruned to prevent re-learning; (4) Fine-tuning: an optional short fine-tuning step can be applied to restore accuracy on retained data. For sequential unlearning, steps (2)–(4) are repeated iteratively.

performance in our client-wise setting, which can make forgetting events more detectable and degrade generalization.

Selective Synaptic Dampening (SSD). Foster et al. (2024) As a Fisher-guided alternative to aggressive pruning, SSD uses the Fisher information of the training and forgetting data to (i) select parameters that are disproportionately important to the forget set and (ii) *dampen* these parameters by a factor proportional to their relative importance to $\mathcal{D}_f$ versus $\mathcal{D}$. This proportional shrinkage preserves parameters important to the retain distribution and typically yields smaller deviations from baseline than hard pruning.

Pruning can simplify unlearning. Jia et al. (2023) show that sparsity can make class-wise unlearning easier, suggesting a potential synergy between pruning and unlearning. However, in our subset-wise, client-level scenario, strong pruning tends to widen gaps between forgotten and non-forgotten behavior, conflicting with our objective of remaining close to the gold model.

One-shot unlearning via mnemonic codes. Yamashita et al. (2024) propose an approach for settings with no access to the original data: they generate per-class mnemonic codes, encourage the model to memorize them, and use only these codes to drive forgetting. This is well-suited to single-class forgetting but does not directly extend to sequential, client-wise requests.

Continual unlearning frameworks. UnCLe (Kumaravelu et al., 2024) is, to our knowledge, the first continual unlearning framework; it couples knowledge distillation with multiple loss terms to preserve utility while removing targeted information. Our work differs in that we pursue a retrain-free, Fisher-guided post-hoc adjustment with an MTL architecture tailored to client-wise unlearning.

3. Notation and Preliminaries

We study sequential subset-wise unlearning in a multi-client setting: a model is trained on data aggregated from multiple clients and, over time, receives one or more requests to remove the influence of specific clients. The goal is to erase the contribution of the requested client(s) while preserving accuracy on the remaining data. Our pipeline for learning and unlearning is summarized in Figure 3.

Formal setup. We denote the full training set by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with class labels $y_i \in \{0, \dots, K-1\}$. Data are partitioned across C clients, and each sample has a client (subset) label $s_i \in \{0, \dots, C-1\}$. A forget request targets either a single client c^* or a set of clients; we write the corresponding forget subset as $\mathcal{D}_f \subset \mathcal{D}$ and the retain subset as $\mathcal{D}_r = \mathcal{D} \backslash \mathcal{D}_f$. Throughout, we exemplify our contribution on CIFAR-10 (K=10) and CIFAR-100 (K=100).

3.1. Client partitions via Dirichlet non-IID split

Because benchmark datasets like CIFAR-10/100 do not include client identifiers, we synthesize client partitions to emulate a multi-client scenario. This synthetic partitioning is necessary to study client-wise unlearning when actual

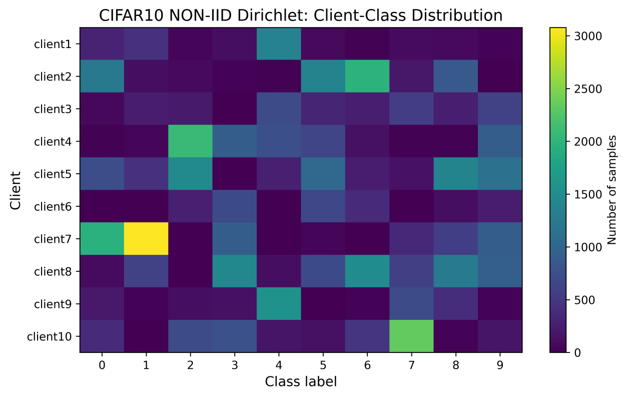


Figure 4. Client-class distribution heatmap (CIFAR-10, C=10, α=0.6). Rows (y-axis) are clients c_t and columns (x-axis) are classes; color encodes sample counts. Lower α yields sharper specialization (more non-IID), higher α approaches IID.

client metadata are unavailable. We partition the training data across C clients using a Dirichlet distribution to induce non-IID heterogeneity. For each class $c \in \{0, \dots, 9\}$ we draw proportions (Eq. (1))

$$\boldsymbol{\pi}_c \sim \text{Dirichlet}(\alpha\, \mathbf{1}_C), \tag{1}$$

and allocate indices of class c to clients according to $\boldsymbol{\pi}_c$ (via cumulative splits). We use a concentration $\alpha = 0.6$ and a fixed seed for reproducibility. This generates a realistic imbalance both in per-client class mixtures and total sample counts.

Briefly, the Dirichlet distribution on the probability simplex Δ^{C-1} with parameter vector $\boldsymbol{\alpha} \in \mathbb{R}^C_{>0}$ has density (Eq. (2))

$$p(\boldsymbol{\pi}; \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{C} \pi_k^{\alpha_k - 1}, \tag{2}$$

where $B(\boldsymbol{\alpha})$ is the multivariate Beta function (Bishop, 2006; Murphy, 2012). Writing $\alpha_0 = \sum_{k=1}^{C} \alpha_k$, its moments are (Eq. (3))

$$\begin{aligned} \mathbb{E}[\pi_k] &= \frac{\alpha_k}{\alpha_0}, \\ \text{Var}[\pi_k] &= \frac{\alpha_k(\alpha_0 - \alpha_k)}{\alpha_0^2(\alpha_0 + 1)}, \\ \text{Cov}(\pi_k, \pi_\ell) &= -\frac{\alpha_k \alpha_\ell}{\alpha_0^2(\alpha_0 + 1)}. \end{aligned} \tag{3}$$

In the symmetric case used here ($\boldsymbol{\alpha} = \alpha\, \mathbf{1}_C$), $\alpha < 1$ yields sparse, highly heterogeneous proportions (clients specialize), $\alpha = 1$ is uniform over the simplex, and $\alpha > 1$ concentrates near uniform allocations. We choose $\alpha = 0.6$ to intentionally induce non-IID heterogeneity.

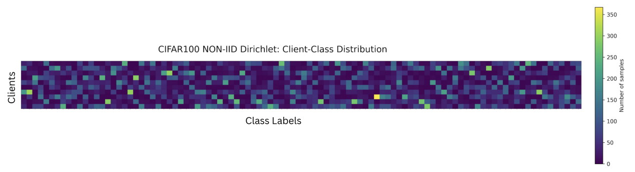


Figure 5. Client-class distribution heatmap (CIFAR-100, C=10, α=0.6). Rows (y-axis) are clients c_t and columns (x-axis) are classes; color encodes sample counts. Lower α yields sharper specialization (more non-IID), higher α approaches IID.

Client	CIFAR-10 Samples	CIFAR-100 Samples
client1	2575	5259
client2	5913	5105
client3	3263	4306
client4	5538	4391
client5	6895	4995
client6	2359	3820
client7	7847	5442
client8	7200	5365
client9	3226	5281
client10	5184	6036

Table 1. Per-client sample counts under the Dirichlet split (α=0.6) for both CIFAR-10 and CIFAR-100 datasets.

4. Method

Objective and rationale. Our goal is subset-wise sequential unlearning: iteratively remove a target client's influence while preserving performance on retained data and staying close to a gold model retrained on the retain set. We use a two-head multi task design so that the subset/client-ID head encourages representations that disentangle client-specific factors from class semantics, making client footprints more controllable for unlearning. For the actual unlearning step, we adopt a Fisher-guided update (SSD, Foster et al. (2024)) that leverages a curvature-weighted view of the loss to selectively dampen parameters specialized to the forget set while protecting those important to the overall training distribution (see the Taylor/Fisher justification in Section 4.3).

4.1. MTL task design

Backbone: ResNet-18. Heads: (1) main class head; (2) subset/client-ID head. For a minibatch $\{(x_i, y_i, s_i)\}_{i=1}^{N}$ the training objective is (Eq. (4))

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{main}} + \lambda_2 \mathcal{L}_{\text{aux}} + \lambda_{\text{dis}} \mathcal{L}_{\text{disentangle}}, \tag{4}$$

with cross-entropy (CE) for both main and auxiliary (subset identification) tasks. The disentanglement loss (DL) term encourages within-subset compactness and between-subset separation in the embedding space z_i (Eqs. (5)–(7)):

$$\mathcal{L}_{\text{pull}} = \frac{1}{|S|} \sum_{s \in S} \frac{1}{|I_s|} \sum_{i \in I_s} \|z_i - \bar{z}_s\|^2, \tag{5}$$

$$\mathcal{L}_{\text{push}} = \frac{1}{|\mathcal{P}|} \sum_{(s,s') \in \mathcal{P}} \big[\max(0,\, m - \|\bar{z}_s - \bar{z}_{s'}\|)\big]^2, \tag{6}$$

$$\mathcal{L}_{\text{disentangle}} = \lambda_{\text{pull}} \mathcal{L}_{\text{pull}} + \lambda_{\text{push}} \mathcal{L}_{\text{push}}. \tag{7}$$

Here, S is the set of subset IDs in the minibatch; I_s is the index set of samples with subset label s; z_i is the backbone embedding of sample i; $\bar{z}_s = \frac{1}{|I_s|} \sum_{i \in I_s} z_i$ is the centroid for subset s; $\mathcal{P} = \{(s, s') \in S \times S :\ s \neq s'\}$ is the set of distinct subset pairs; $m > 0$ is the margin; and $\|\cdot\|$ denotes the Euclidean norm. If $|S| = 1$, then $\mathcal{L}_{\text{push}} = 0$.

The loss elements $\mathcal{L}_{\text{main}}$ and $\mathcal{L}_{\text{aux}}$ are Cross Entropy Loss and their coefficients λ_1 and λ_2 are chosen as 1. The $\mathcal{L}_{\text{aux}}$'s coefficient λ_{dis} is chosen as 0.1 to hinder a big effect on the main learning task. λ_{pull} and λ_{push} are chosen as 1.

4.2. Unlearning via SSD

Functioning principle. SSD "forgets" by first finding parameters that matter much more for the forget set than for the overall data (measured with diagonal Fisher via squared gradients). It then shrinks only those specialized parameters *in proportion to their specialization*, which raises loss on the forget set while keeping the rest of the model stable.

Let $F_{\mathcal{D}}$ and $F_{\mathcal{D}_f}$ denote the diagonal Fisher information over $\mathcal{D}$ and $\mathcal{D}_f$ respectively (computed by squared gradients). SSD selects parameters specialized to $\mathcal{D}_f$ (Eq. (8)):

$$\mathcal{S} = \{i :\ F_{\mathcal{D}_f,i} > \alpha\, F_{\mathcal{D},i}\}, \tag{8}$$

then dampens them proportionally (Eq. (9)):

$$\beta_i = \min\left(\frac{\lambda F_{\mathcal{D},i}}{F_{\mathcal{D}_f,i}}, 1\right), \qquad \theta'_i = \begin{cases} \beta_i \theta_i, & i \in \mathcal{S}, \\ \theta_i, & \text{otherwise.} \end{cases} \tag{9}$$

The *proportionality* part is important because it differs from other Fisher-based methods. For example, in Yamashita et al. (2024), important parameters are perturbed, while in Shi et al. (2024), full pruning is applied to these parameters.

4.3. Taylor/Fisher view (a justification for Fisher-guided unlearning)

Quadratic increase of loss under weight perturbations. The one-dimensional Taylor expansion of a function f around a point a is (Eq. (10))

$$f(x) = f(a) + f'(a)(x-a) + \tfrac{1}{2} f''(a)(x-a)^2 + O\big((x-a)^3\big). \tag{10}$$

For a multivariate loss $\mathcal{L}(w)$, expanding around optimized weights w^* with perturbation δ, we obtain (Eq. (11))

$$\mathcal{L}(w^* + \delta) \approx \mathcal{L}(w^*) + \nabla \mathcal{L}(w^*)^\top \delta + \tfrac{1}{2}\, \delta^\top H\, \delta + O(\|\delta\|^3), \tag{11}$$

where H is the Hessian of $\mathcal{L}$ at w^*. At a stationary point w^*, the gradient vanishes (Eq. (12)):

$$\nabla \mathcal{L}(w^*) = 0. \tag{12}$$

Hence, the expansion simplifies to (Eq. (13)):

$$\mathcal{L}(w^* + \delta) \approx \mathcal{L}(w^*) + \tfrac{1}{2}\, \delta^\top H\, \delta. \tag{13}$$

If H is diagonalized (e.g., in an eigenbasis), the expression becomes (Eq. (14))

$$\mathcal{L}(w^* + \delta) \approx \mathcal{L}(w^*) + \tfrac{1}{2} \sum_i \lambda_i\, \delta_i^2, \tag{14}$$

where λ_i are the eigenvalues of the Hessian. Thus, along directions with positive curvature ($\lambda_i > 0$), any nonzero perturbation increases the loss to second order.

Remark. The higher-order term $O(\|\delta\|^3)$ is omitted in this quadratic approximation. Locally approximating the loss by a quadratic form.

Under standard regularity conditions at an optimum, the Fisher Information Matrix equals the Hessian in expectation(Bishop, 2006; Murphy, 2012). Using its (tractable) diagonal, we arrive at the curvature-weighted approximation in Eq. (16), which we leverage to increase loss on a target set while constraining loss on others for unlearning. Let $\mathcal{L}_S(\theta)$ denote the empirical loss over a dataset S. For a model trained to (a local) optimum θ^* on $\mathcal{D}$, we have $\nabla \mathcal{L}_{\mathcal{D}}(\theta^*) = 0$. A second-order expansion around θ^* gives, for a small perturbation δ,

$$\mathcal{L}_S(\theta^* + \delta) \approx \mathcal{L}_S(\theta^*) + \tfrac{1}{2}\, \delta^\top H_S\, \delta, \tag{15}$$

where H_S is the Hessian of $\mathcal{L}_S$ at θ^*. Under standard regularity conditions, the Fisher Information Matrix (FIM) satisfies $\mathbb{E}[\nabla \ell\, \nabla \ell^\top] = H$ at the optimum, and in practice we use its diagonal $F_S = \operatorname{diag}(\text{FIM}_S)$ as a tractable curvature proxy. With a diagonal approximation, Eq. (15) becomes

$$\Delta \mathcal{L}_S(\delta) \triangleq \mathcal{L}_S(\theta^* + \delta) - \mathcal{L}_S(\theta^*) \approx \tfrac{1}{2} \sum_i F_{S,i}\, \delta_i^2. \tag{16}$$

Forgetting as a constrained quadratic objective. Unlearning aims to increase the loss on the forget set $\mathcal{D}_f$ while limiting the loss increase on the original training distribution $\mathcal{D}$. Using Eq. (16), a natural formulation is

$$\max_{\delta} \quad \tfrac{1}{2} \sum_i F_{\mathcal{D}_f,i}\, \delta_i^2 \tag{17}$$

$$\text{s.t.} \quad \tfrac{1}{2} \sum_i F_{\mathcal{D},i}\, \delta_i^2 \ \le\ \varepsilon \tag{18}$$

for a small budget $\varepsilon > 0$ controlling deviation from the trained solution. Since Eqs. (17)–(18) are separable in coordinates, the optimal allocation prioritizes indices with the largest specialization ratio

$$\rho_i \;=\; \frac{F_{\mathcal{D}_f,i}}{F_{\mathcal{D},i}}. \tag{19}$$

This immediately motivates SSD's *selection rule*: update only coordinates with $F_{\mathcal{D}_f,i} > \alpha\, F_{\mathcal{D},i}$ (i.e., $\rho_i > \alpha$) for a threshold $\alpha > 0$.

Dampening magnitude. The Lagrangian for Eqs. (17)–(18) with diagonal curvature is $\mathcal{L}(\delta,\lambda) = \frac{1}{2}\sum_i F_{\mathcal{D}_f,i}\,\delta_i^2 - \lambda\big(\frac{1}{2}\sum_i F_{\mathcal{D},i}\,\delta_i^2 - \varepsilon\big)$. Stationarity in each coordinate yields $\delta_i^2 \propto \lambda\, F_{\mathcal{D},i}/F_{\mathcal{D}_f,i}$. Translating a perturbation budget into a *multiplicative* parameter shrinkage suggests

$$\theta_i' \;=\; \beta_i\,\theta_i, \qquad \beta_i \;=\; \min\Big(\frac{\lambda\, F_{\mathcal{D},i}}{F_{\mathcal{D}_f,i}},\, 1\Big), \tag{20}$$

which matches SSD's dampening rule: coordinates more specialized to $\mathcal{D}_f$ (large $F_{\mathcal{D}_f,i}$) receive stronger shrinkage, while coordinates important to $\mathcal{D}$ (large $F_{\mathcal{D},i}$) are protected. The cap at 1 prevents growth for large multipliers.

Takeaway. The second-order (Taylor) view with a diagonal FIM links unlearning to a separable, constrained quadratic problem. The solution structure precisely recovers SSD's two components: (i) *selection* via a ratio test $F_{\mathcal{D}_f,i} > \alpha F_{\mathcal{D},i}$, and (ii) *dampening* with factor β_i in Eq. (20). Thus, Fisher-guided selection and proportional dampening are not ad hoc; they are the natural consequence of maximizing loss on $\mathcal{D}_f$ under a curvature-weighted budget that preserves performance on $\mathcal{D}$.

Sequential forgetting (MTL). We apply SSD and fine-tuning iteratively; for already-forgotten clients we prune their subset-ID neurons during evaluation and fine-tuning.

Sequential forgetting (no-MTL). No output neuron pruning is needed; Fisher is computed on the main classification task and the retain loss excludes the subset term.

4.4. Which head's loss to use during FIM calculation?

A practical question in MTL is which head's loss to use when estimating the Fisher Information Matrix.

- **FIM from the subset head.** Using the subset/client-ID head to compute the FIM did not open up the desired gap relative to the MTL-free baseline. In practice, the no-MTL pipeline yielded better main classification accuracies.

Table 2. Main-task accuracy and subset-ID accuracy under classification-head FIM, without output subset-ID neuron pruning.

	Main task accuracy		Subset ID Accuracy			
	Client 1	Others	After SSD		After 1 Epoch	
			Client 1	Others	Client 1	Others
Values	0.9020	0.9610	<u>0.2711</u>	0.6551	<u>0.4223</u>	0.7327

- **FIM from the main classification head.** Using the main class head to compute the FIM introduces a different issue: the subset-ID head is not fully forgotten and can begin to re-learn during fine-tuning. For example, the target subset's subset-ID accuracy is not fully driven down and rebounds after even a single epoch of fine-tuning, while the *Others* subset-ID accuracy also increases (Table 2).

Proposed solution. We compute the FIM from the classification head to maintain performance comparable to or better than the no-MTL setting, and we prune the forgotten subset's output neuron in the subset-ID head to prevent its accuracy from increasing during fine-tuning and evaluation.

5. Evaluation

We include full tables for CIFAR-10 and CIFAR-100 on ResNet-18 for retain/forget/test accuracies, MIA, and delta-to-gold score. We evaluate both MTL and no-MTL baselines and sequential multi-client forgetting. In the following tables, CE refers to Cross Entropy loss and DL refers to Disentanglement Loss.

Implementation details. We train for a total of $T = 200$ epochs. For epochs $t = 1, \ldots, 20$, we optimize only the main classification head with cross entropy (i.e., λ_2=0 and λ_{dis}=0). For epochs $t = 21, \ldots, 200$, we enable the subset/client-ID head and the disentanglement loss, using λ_1=1, λ_2=1, λ_{dis}=0.1, and λ_{pull}=λ_{push}=1. This schedule yields high subset-ID accuracy (approximately $\sim 99\%$) and well-separated client representations that are later exploited by Fisher-guided unlearning (cf. Tables 3 and 4).

Metrics. We evaluate with: (i) main classification accuracy on the forgotten subset vs. other subsets; (ii) subset-ID accuracy; (iii) test accuracy; and (iv) a train-only logistic-regression MIA that distinguishes retain from forget samples using the model's softmax outputs.

Membership inference attack (MIA) and our regression attacker. Membership inference attacks probe whether a model's outputs leak information about whether a sample was in the training set. Following prior work (Chundawat et al., 2023), we adopt a simple and strong attacker based on multinomial logistic regression:

Algorithm 1 Sequential subset-wise unlearning (MTL)

```
1: Input: baseline model $M^{(0)}$, client order $\mathcal{C} = [c_1, \dots]$, precomputed $F_{\mathcal{D}}$, fine-tune grids for epochs and $(\lambda_{\text{class}}, \lambda_{\text{subset}})$
2: $\mathcal{F} \leftarrow \emptyset$    (set of forgotten clients)
3: for $t = 1, 2, \dots$ do
4:     SSD: Given $M^{(t-1)}$ and current client $c_t$, compute $F_{\mathcal{D}_f}(c_t)$ (on subset task) and apply Eq. (20) to obtain $\tilde{M}^{(t)}$.
5:     Prune subset-ID output neuron: In $\tilde{M}^{(t)}$, suppress output neurons for all clients in $\mathcal{F} \cup \{c_t\}$.
6:     Fine-tune (Optional): From $\tilde{M}^{(t)}$, grid-search epochs $\in \{1, 2, 3\}$ and $(\lambda_{\text{class}}, \lambda_{\text{subset}})$ using
           $$\mathcal{L} = \underbrace{\mathrm{CE}(\hat{y}_r, y_r) + \lambda_{\text{subset}}\,\mathrm{CE}(\hat{s}_r, s_r)}_{\text{retain}} \; \underbrace{-\left(\lambda_{\text{class}}\, \max\{0, a_f - a_f^{\text{base}}\}\right) \mathrm{CE}(\hat{y}_f, y_f)}_{\text{adversarial on forget}},$$
       and select the checkpoint minimizing deviation from per-round baselines to get $M^{(t)}$.
7:     $\mathcal{F} \leftarrow \mathcal{F} \cup \{c_t\}$
8: end for
9: Output: $M^{(t)}$ after the final round.
```

Algorithm 2 Sequential subset-wise unlearning (no-MTL)

```
1: Input: baseline model $M^{(0)}$, client order $\mathcal{C} = [c_1, \dots]$, precomputed $F_{\mathcal{D}}$, fine-tune grids for epochs and $\lambda_{\text{class}}$
2: for $t = 1, 2, \dots$ do
3:     SSD: Given $M^{(t-1)}$ and current client $c_t$, compute $F_{\mathcal{D}_f}(c_t)$ (on main classification task) and apply Eq. (20) to obtain $\tilde{M}^{(t)}$.
4:     Fine-tune (Optional): From $\tilde{M}^{(t)}$, grid-search epochs $\in \{1, 2, 3\}$ and $\lambda_{\text{class}}$ using
           $$\mathcal{L} = \underbrace{\mathrm{CE}(\hat{y}_r, y_r)}_{\text{retain}} \; \underbrace{-\left(\lambda_{\text{class}}\, \max\{0, a_f - a_f^{\text{base}}\}\right) \mathrm{CE}(\hat{y}_f, y_f)}_{\text{adversarial on forget}},$$
       and select the checkpoint minimizing deviation from per-round baselines to get $M^{(t)}$.
5: end for
6: Output: $M^{(t)}$ after the final round.
```

Features: For MTL, we concatenate the main-head and subset-head softmax probability vectors into a single feature vector per sample. For no-MTL, we use the class-head softmax only.

Labels: Outputs computed on retain data are labelled 0; outputs computed on the (current) forget set are labelled 1.

Attacker training: We fit a multinomial logistic regression classifier with class_weight=balanced and 5-fold stratified cross-validation, reporting cross-validated predictions. The metric is overall classification accuracy (in percent).

Intuitively, if the unlearned model still behaves differently on $\mathcal{D}_f$ than on $\mathcal{D}_r$, the attacker will separate the two and achieve high accuracy. Our goal is for the MIA score to match that of a gold model trained from scratch on $\mathcal{D}_r$. This avoids both extremes: high MIA (forgetting failed) and suspiciously low MIA coupled with pathological behavior (e.g., deliberately wrong predictions), which can reveal forgetting events. Logistic regression is data-efficient, fast, and sufficient here because the feature space (probability vectors) already captures the separability induced by leakage.

Experimental setup. Our evaluation follows a two-step protocol: first, we test with optional fine-tuning (applying fine-tuning only if it improves performance), then we test without any fine-tuning. We present results for CIFAR-10 experiments first, followed by CIFAR-100 experiments. We also report other FIM-based methods like DeepClean and Lottery Ticket-style pruning, as well as only pruning and fine-tuning retain-sensitive weights.

Each unlearning round yields a checkpoint; we refer to these as *Stage 1–3*: Stage 1 (after unlearning Client 1), Stage 2 (after unlearning Client 2 from a model that previously unlearned Client 1), and Stage 3 (after unlearning Client 3 from a model that previously unlearned Client 1 and Client 2). Results with optional fine-tuning for CIFAR-10 are summarized in Tables 5–7 and results without fine-tuning for CIFAR-10 are summarized in Tables 8–10. Results with optional fine-tuning for CIFAR-100 are summarized in Tables 11–13 and results without fine-tuning for CIFAR-100 are summarized in Tables 14–16.

Table 3. Learning Phase performance on CIFAR-10. "-" indicates not applicable.

Method	Main Task Train Accuracy	Subset ID Accuracy	Main Task Test Accuracy
Without MTL (SSD)	0.9801	–	0.9064
MTL with CE (Ours)	0.9795	0.9767	0.8987
MTL with CE + DL (Ours)	**0.9828**	**0.9936**	**0.9111**

Table 4. Learning Phase performance on CIFAR-100. "-" indicates not applicable.

Method	Main Task Train Accuracy	Subset ID Accuracy	Main Task Test Accuracy
Without MTL (SSD)	0.9898	–	0.6726
MTL with CE (Ours)	0.9993	0.9974	0.6724
MTL with CE + DL (Ours)	**0.9996**	**0.9984**	**0.6920**

5.1. Ablation: SSD hyperparameter search

Following recommendations on Foster et al. (2024), we performed a small Optuna search over SSD hyperparameters per round. On CIFAR-10, we ran 100 trials; on CIFAR-100, we ran 50 trials. We used log-uniform suggestions over the following ranges:

$$\alpha \sim \text{log-uniform}(0.1,\ 100.0),$$
$$\lambda \sim \text{log-uniform}(0.01,\ 10.0).$$

We report the main results with the best-performing checkpoints under our deviation-to-baseline criterion; detailed ablations are consistent with the overall trends.

Values in parentheses denote divergence from the baseline; smaller is better (closer to the gold model).

6. Discussion

Learning-phase (both for CIFAR-10 and CIFAR-100). Table 3 and 4 show that adding Disentanglement Loss yields the best subset-ID accuracy and increases overall classification accuracy in the learning phase.

Pruning vs. dampening. Threshold-based pruning plus retraining (DeepClean: Shi et al. (2024)-like) removes many weights and induces large gaps between forgotten and non-forgotten performance, making forgetting detectable. SSD selects only specialized parameters and dampens them proportionally, reducing deviation from baseline and preserving generalization.

Why not pruning? As explained above, DeepClean or Lottery Ticket-style pruning (Frankle & Carbin, 2019) can enlarge the gap between forgotten and non-forgotten performance, but typically at the cost of drifting farther from the gold model. Conversely, when pruning only a small fraction of the network to remain close to the baseline, we observe substantial drops in subset-ID accuracy; restoring it to a reasonable level (e.g., $> 70\%$) can require more than 50 epochs of fine-tuning, well beyond our regime (no fine-tuning or at most 3 epochs). Detailed results for DeepClean and Lottery Ticket-style pruning are provided in the tables (see Table 18 along with the baseline Table 17); as shown there, either long fine-tuning (50+ epochs) is needed to recover subset-ID accuracy, or the main classification accuracies remain far from the gold model. In other words, either the target subset is not forgotten or the subset-ID accuracy takes long fine-tuning to recover.

MTL can help improve control. The subset-ID head and disentanglement term might yield clearly separated client embeddings, sharpening Fisher specialization and enabling more precise SSD. Results in the tables in between 5 and 16 show that MTL design usually have a closer performance to the baseline.

Limits of post-perturbation fine-tuning. In our pipeline, dampening is a targeted perturbation applied near a local optimum. Subsequent fine-tuning often yields limited gains or even regressions: once optimization has converged, small-step updates around the perturbed basin tend to restore pre-perturbation behavior only partially while also eroding the desired increase in forget-set loss. Empirically, we observe that combining Fisher-guided dampening with additional fine-tuning does not consistently improve the separation we seek, especially for CIFAR-10 (see the Tables 6 and 7). It aligns with the intuition that local post-hoc updates cannot reliably re-navigate to a better basin after a deliberate curvature-weighted perturbation.

Is fine-tuning necessary? We noticed that, for CIFAR-100, fine-tuning is necessary, especially for test set accuracy; even just 3 epochs is very useful (see Tables 11, 12, and 13). However, as explained in the previous paragraph, fine-tuning was generally not needed for CIFAR-10, showing that an increase in the number of classes can make the training-free unlearning further challenging.

CIFAR-100 nuance on fine-tuning selection. On CIFAR-100, we further observed that post-hoc fine-tuning can bias our checkpoint selection in favor of the fine-tuned models even when per-client ("unlearned") accuracies are not actually better. The deciding factor is the *Others* main-task accuracy, which tends to saturate under fine-tuning; this single component then dominates the selection criterion. In contrast, without fine-tuning, the unlearned accuracies are typically closer to the baseline (gold) model. In short, only the *Others* accuracy term meaningfully shifts with fine-tuning, which can mask that the unlearned metrics are better without it. See the comparison of fine-tuned CIFAR-100 Tables 11, 12, 13 and not fine-tuned CIFAR-100 Tables 14, 15, 16.

7. Conclusion

We have formalized and addressed the challenging problem of subset-wise sequential machine unlearning, where forget requests target data clients spanning multiple classes rather than single classes. Our approach combines a two-head multi task learning architecture with disentanglement loss and Fisher Information Matrix-guided Selective Synaptic Dampening to achieve targeted forgetting while preserving performance on retained data.

Key findings from our experiments on CIFAR-10 and

Table 5. CIFAR-10 Approximate Unlearning in Stage 1. Values in parentheses denote divergence from the baseline.

	Main Task Accuracy			Subset ID Accuracy		MIA	Is Fine-tuning Better?
Method	Client 1	Others	Test	Client 1	Others		
Without MTL (SSD)	0.9010 (0.0017)	0.9584 (0.0415)	0.8719 (0.0342)	-	-	74.0440 (**0.0113**)	Yes (3 Epochs)
MTL with CE (Ours)	0.9049 (0.0074)	0.9630 (0.0365)	0.8822 (0.0220)	0	0.8027 (**0.1882**)	78.3640 (0.0139)	No
MTL with CE + DL (Ours)	0.9033 (**0.0004**)	0.9702 (**0.0272**)	0.8914 (**0.0216**)	0	0.7529 (0.2445)	77.8700 (0.0181)	Yes (1 Epoch)

Table 6. CIFAR-10 Approximate Unlearning in Stage 2. Values in parentheses denote divergence from the baseline.

	Main Task Accuracy				Subset ID Accuracy			MIA	Is Fine-tuning Better?
Method	Client 1	Client 2	Others	Test	Client 1	Client 2	Others		
Without MTL (SSD)	0.8967 (**0.0016**)	0.8853 (0.0069)	0.9279 (0.0721)	0.8508 (**0.0423**)	-	-	-	70.3279 (0.0860)	No
MTL with CE (Ours)	0.8979 (0.0023)	0.8973 (0.0130)	0.9302 (0.0693)	0.8475 (0.0511)	0	0	0.7872 (**0.2068**)	72.0865 (**0.0481**)	No
MTL with CE + DL (Ours)	0.8993 (0.0114)	0.8955 (**0.0010**)	0.9367 (**0.0632**)	0.8508 (0.0544)	0	0	0.7487 (0.2498)	74.5000 (0.0563)	No

CIFAR-100 include: (1) MTL pipeline performs comparable to or better than the no-MTL setting in maintaining proximity to gold models, (2) Fisher-guided dampening provides more controlled forgetting compared to aggressive pruning methods, and (3) our approach handles sequential forget requests without catastrophic degradation.

Our method addresses a realistic deployment scenario where machine learning systems must accommodate evolving privacy requirements and client-specific data removal requests. By maintaining performance close to the gold model while avoiding computationally expensive full training, our approach offers a practical solution for continual unlearning in multi-client environments.

Limitations

Several limitations should be acknowledged in our work. First, our evaluation is restricted to image classification tasks on CIFAR-10 and CIFAR-100 with ResNet-18. Generalizability to other domains, architectures, and larger-scale datasets with different Dirichlet distributions and number of clients remains to be demonstrated.

Additionally, experiments were conducted on Kaggle P100 GPUs; a more robust evaluation can be performed by running multiple seeds on more powerful GPUs and reporting the average results.

Impact Statement

This work aims to provide efficient unlearning in a realistic task design, which includes client-wise settings and sequential forget requests. Risks include misuse to mask data removal or regulatory gaming, or superficial compliance, which gives the appearance of following regulations without achieving their intended purpose; we mitigate by targeting similarity to a gold model and avoiding revealing gaps that telltale forgetting.

References

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

Chundawat, V. S., Tarun, A. K., Mandal, M., and Kankanhalli, M. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7210–7217, 2023.

Foster, J., Schoepf, S., and Brintrup, A. Fast machine unlearning without retraining through selective synaptic dampening. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Golatkar, S., Achille, A., and Soatto, S. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Jia, J., Liu, J., Ram, P., Yao, Y., Liu, G., Liu, Y., and Liu, S. Model sparsity can simplify machine unlearning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51584–51605, 2023.

Kumaravelu, V., Adhikari, S., and Srijith, P. K. Uncle: An unlearning framework for continual learning. OpenReview preprint, 2024. Submitted to ICLR 2025; version 2025-02-05.

Liu, C., Wang, Y., Flanigan, J., and Liu, Y. Large language model unlearning via embedding-corrupted prompts. In *Advances in Neural Information Processing Systems*, volume 37, pp. 118198–118266, 2024.

Liu, K., Choquette-Choo, C. A., Jagielski, M., Kairouz, P., Koyejo, S., Liang, P., and Papernot, N. Language

Table 7. CIFAR-10 Approximate Unlearning in Stage 3. Values in parentheses denote divergence from the baseline.

	Main Task Accuracy					Subset ID Accuracy				MIA	Is Fine-tuning Better?
Method	Client 1	Client 2	Client 3	Others	Test	Client 1	Client 2	Client 3	Others		
Without MTL (SSD)	0.8672 (0.0085)	0.8918 (0.0173)	0.9038 (0.0126)	0.9255 (0.0744)	0.8465 (0.0454)	-	-	-	-	60.1609 (0.0889)	No
MTL with CE (Ours)	0.8804 **(0.0043)**	0.8816 **(0.0051)**	0.9160 (0.0177)	0.9265 (0.0716)	0.8437 **(0.0432)**	0	0	0	0.8087 **(0.1648)**	65.248 (0.1439)	No
MTL with CE + DL (Ours)	0.8750 (0.0256)	0.8869 (0.0076)	0.9065 **(0.0083)**	0.9360 **(0.0636)**	0.8483 (0.0542)	0	0	0	0.7754 (0.2212)	63.5550 **(0.0333)**	No

Table 8. CIFAR-10 Approximate Unlearning in Stage 1 (No fine-tuning). Values in parentheses denote divergence from the baseline.

	Main Task Accuracy			Subset ID Accuracy		MIA
Method	Client 1	Others	Test	Client 1	Others	
Without MTL (SSD)	0.9146 (0.0153)	0.9703 **(0.0296)**	0.8872 **(0.0189)**	-	-	73.9060 (0.099)
MTL with CE (Ours)	0.9021 **(0.0046)**	0.9629 (0.0366)	0.8822 (0.0220)	0	0.8038 **(0.1871)**	75.88 **(0.03)**
MTL with CE + DL (Ours)	0.9099 (0.0062)	0.9454 (0.0520)	0.8755 (0.0375)	0	0.6688 (0.3286)	74.71 (0.045)

models may verbatim complete text they were not explicitly trained on. In *International Conference on Machine Learning*, 2025.

Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Sakarvadia, M., Ajith, A., Khan, A. M., Hudson, N. C., Geniesse, C., Chard, K., Yang, Y., Foster, I., and Mahoney, M. W. Mitigating memorization in language models. In *International Conference on Learning Representations*, 2025.

Scholten, Y., G"unnemann, S., and Schwinn, L. A probabilistic perspective on unlearning and alignment for large language models. In *International Conference on Learning Representations*, 2025. Preprint 2024.

Shi, J., Ghalyan, N., Gourgoulias, K., Buford, J., and Moran, S. Deepclean: Machine unlearning on the cheap by resetting privacy sensitive weights using the fisher diagonal. In *European Conference on Computer Vision Workshops (ECCV Workshops)*, 2024. URL https://arxiv.org/abs/2311.10448. U&Me Workshop.

Yamashita, T., Yamada, M., and Shibata, T. One-shot machine unlearning with mnemonic code. In *Proceedings of The 16th Asian Conference on Machine Learning*, volume 260 of *Proceedings of Machine Learning Research*. PMLR, 2024.

Table 9. CIFAR-10 Approximate Unlearning in Stage 2 (No fine-tuning). Values in parentheses denote divergence from the baseline.

	Main Task Accuracy				Subset ID Accuracy			MIA
Method	Client 1	Client 2	Others	Test	Client 1	Client 2	Others	
Without MTL (SSD)	0.9064 (0.0113)	0.8901 (0.0117)	0.9295 (0.0705)	0.8398 (0.0533)	-	-	-	68.7106 (0.0697)
MTL with CE (Ours)	0.8905 **(0.0062)**	0.8904 (0.0061)	0.9304 **(0.0691)**	0.8475 (0.0511)	0	0	0.7877 **(0.2063)**	64.11 (0.0165)
MTL with CE + DL (Ours)	0.9017 (0.009)	0.8957 **(0.0012)**	0.9273 (0.0726)	0.8615 **(0.0437)**	0	0	0.6613 (0.3372)	64.00 **(0.007)**

Table 10. CIFAR-10 Approximate Unlearning in Stage 3 (No fine-tuning). Values in parentheses denote divergence from the baseline.

	Main Task Accuracy					Subset ID Accuracy				MIA
Method	Client 1	Client 2	Client 3	Others	Test	Client 1	Client 2	Client 3	Others	
Without MTL (SSD)	0.8637 (0.0120)	0.8874 (0.0129)	0.8964 (0.0052)	0.9246 (0.0753)	0.8338 (0.0581)	-	-	-	-	59.4792 (0.0157)
MTL with CE (Ours)	0.8579 (0.0182)	0.8747 **(0.0018)**	0.9078 (0.0095)	0.9122 (0.0859)	0.9122 **(0.0253)**	0	0	0	0.7793 **(0.1942)**	61.32 (0.0105)
MTL with CE + DL (Ours)	0.9115 **(0.0109)**	0.8999 (0.0054)	0.9166 **(0.0018)**	0.9277 **(0.0719)**	0.8612 (0.0413)	0	0	0	0.6989 (0.2977)	64.201 **(0.0002)**

Table 11. CIFAR-100 Approximate Unlearning in Stage 1. Values in parentheses denote divergence from the baseline.

	Main Task Accuracy			Subset ID Accuracy		MIA	Is Fine-tuning Better?
Method	Client 1	Others	Test	Client 1	Others		
Without MTL (SSD)	0.8365 (0.2497)	0.9172 (0.0827)	0.6231 **(0.0025)**	-	-	68.2460 (0.0415)	Yes (1 Epoch)
MTL with CE (Ours)	0.8882 (0.2468)	0.9472 **(0.0525)**	0.6607 (0.0084)	0	0.8506 (0.1488)	68.8940 **(0.0177)**	Yes (3 Epochs)
MTL with CE + DL (Ours)	0.8863 **(0.2428)**	0.9431 (0.0567)	0.6648 (0.0260)	0	0.8932 **(0.1063)**	69.4560 (0.0214)	Yes (3 Epochs)

Table 12. CIFAR-100 Approximate Unlearning in Stage 2. Values in parentheses denote divergence from the baseline.

	Main Task Accuracy				Subset ID Accuracy			MIA	Is Fine-tuning Better?
Method	Client 1	Client 2	Others	Test	Client 1	Client 2	Others		
Without MTL (SSD)	0.7678 **(0.2158)**	0.7980 (0.2581)	0.9415 (0.0583)	0.6177 (0.0175)	-	-	-	68.7535 (0.0858)	Yes (3 Epochs)
MTL with CE (Ours)	0.8562 (0.2401)	0.8872 (0.2800)	0.9602 (0.0396)	0.6489 (0.0081)	0	0	0.8894 (0.1099)	70.0409 **(0.0457)**	Yes (3 Epochs)
MTL with CE + DL (Ours)	0.8528 (0.2287)	0.8668 **(0.2517)**	0.9618 **(0.0380)**	0.6614 **(0.0072)**	0	0	0.9247 **(0.0749)**	69.9537 (0.4662)	Yes (3 Epochs)

Table 13. CIFAR-100 Approximate Unlearning in Stage 3. Values in parentheses denote divergence from the baseline.

	Main Task Accuracy					Subset ID Accuracy				MIA	Is Fine-tuning Better?
Method	Client 1	Client 2	Client 3	Others	Test	Client 1	Client 2	Client 3	Others		
Without MTL (SSD)	0.7201 **(0.1829)**	0.7295 **(0.2071)**	0.7534 **(0.2543)**	0.9300 (0.0698)	0.6067 (0.0323)	-	-	-	-	74.2835 (0.1521)	Yes (2 Epochs)
MTL with CE (Ours)	0.8464 (0.2474)	0.8568 (0.2597)	0.8488 (0.2868)	0.9470 (0.0528)	0.6419 (0.0060)	0	0	0	0.9133 (0.0862)	74.9924 **(0.0800)**	Yes (2 Epochs)
MTL with CE + DL (Ours)	0.8395 (0.2310)	0.8511 (0.2476)	0.8558 (0.2782)	0.9602 **(0.0396)**	0.6491 **(0.0004)**	0	0	0	0.9363 **(0.0633)**	75.5374 (0.0820)	Yes (3 Epochs)

Table 14. CIFAR-100 Approximate Unlearning in Stage 1 (No fine-tuning). Values in parentheses denote divergence from the baseline.

	Main Task Accuracy			Subset ID Accuracy		MIA
Method	Client 1	Others	Test	Client 1	Others	
Without MTL (SSD)	0.5872 **(0.0004)**	0.8180 (0.1819)	0.5371 (0.0835)	-	-	66.5520 (0.0245)
MTL with CE (Ours)	0.6828 (0.0414)	0.8828 **(0.1169)**	0.6353 **(0.0338)**	0	0.8435 (0.1559)	65.1000 (0.0013)
MTL with CE + DL (Ours)	0.6406 (0.0029)	0.8612 (0.1386)	0.6174 (0.0734)	0	0.8439 **(0.1556)**	65.09 **(0.0004)**

Table 15. CIFAR-100 Approximate Unlearning in Stage 2 (No fine-tuning). Values in parentheses denote divergence from the baseline.

	Main Task Accuracy				Subset ID Accuracy			MIA
Method	Client 1	Client 2	Others	Test	Client 1	Client 2	Others	
Without MTL (SSD)	0.5054 **(0.0466)**	0.5397 **(0.0002)**	0.7618 (0.2380)	0.4910 **(0.1092)**	-	-	-	63.1971 (0.030)
MTL with CE (Ours)	0.5685 (0.0476)	0.6670 (0.0598)	0.7773 **(0.2225)**	0.5423 (0.1147)	0	0	0.7773 (0.2220)	64.0111 (0.0135)
MTL with CE + DL (Ours)	0.5575 (0.0666)	0.6239 (0.0088)	0.7674 (0.2324)	0.5362 (0.1324)	0	0	0.8356 **(0.1640)**	61.0150 **(0.0002)**

Table 16. CIFAR-100 Approximate Unlearning in Stage 3 (No fine-tuning). Values in parentheses denote divergence from the baseline.

	Main Task Accuracy					Subset ID Accuracy				MIA
Method	Client 1	Client 2	Client 3	Others	Test	Client 1	Client 2	Client 3	Others	
Without MTL (SSD)	0.4493 (0.0879)	0.4964 (0.0260)	0.5016 **(0.0025)**	0.7206 (0.2792)	0.4522 **(0.1222)**	-	-	-	-	66.3488 (0.0723)
MTL with CE (Ours)	0.5193 (0.0797)	0.6065 **(0.0094)**	0.5506 (0.0140)	0.7464 **(0.2534)**	0.5073 (0.1340)	0	0	0	0.7969 (0.2026)	63.1161 (0.0209)
MTL with CE + DL (Ours)	0.5372 **(0.0713)**	0.5806 (0.0229)	0.5915 (0.1300)	0.7439 (0.2559)	0.5102 (0.1393)	0	0	0	0.8360 **(0.1636)**	61.2266 **(0.0103)**

Table 17. CIFAR-10 Stage 1 baseline (gold model) metrics prior to pruning-based unlearning.

Main Task Accuracy			Subset ID Accuracy		MIA
Client 1	Others	Test	Client 1	Others	
0.9056	0.9998	0.9130	0	0.9974	75.44

Table 18. Pruning ablations on CIFAR-10 Stage 1. Selected weight percentages are over ResNet-18; heads are included in fine-tuning. Results illustrate the trade-off: recovering subset-ID often requires long fine-tuning (50+ epochs), otherwise main classification accuracies deviate from the gold model.

Selected Weights' %		Heads in FT	Method	Main Task Accuracy			Subset ID Accuracy		MIA
Forget	Retain			Client 1	Others	Test	Client 1	Others	
0.1	0.1	True	lottery ticket	0.9738	0.9899	0.9021	0	0.8821	76.34
0.1	0.1	False	lottery ticket	0.9814	0.9873	0.9035	0	0.8769	77.18
0.2	0.2	True	lottery ticket	0.9139	0.9672	0.8839	0	0.7055	73.96
0.2	0.2	False	lottery ticket	0.8949	0.9591	0.8788	0	0.6865	74.34
0.4	0.2	True	lottery ticket	0.8349	0.9135	0.8416	0	0.4423	73.85
0.4	0.2	False	lottery ticket	0.8173	0.9000	0.8321	0	0.4227	76.47
0.1	0.1	True	retain-sensitive	0.9795	0.9924	0.9059	0	0.8965	78.25
0.1	0.1	False	retain-sensitive	0.9776	0.9898	0.9022	0	0.9093	78.96
0.2	0.2	True	retain-sensitive	0.9439	0.9782	0.8855	0	0.8140	76.91
0.2	0.2	False	retain-sensitive	0.9315	0.9727	0.8890	0	0.8194	76.03
0.4	0.2	True	retain-sensitive	0.8844	0.9430	0.8643	0	0.5723	73.84
0.4	0.2	False	retain-sensitive	0.8763	0.9433	0.8652	0	0.5663	75.84
0.1	–	True	DeepClean	0.9924	0.9954	0.9097	0	0.9308	77.80
0.1	–	False	DeepClean	0.9910	0.9945	0.9099	0	0.9315	80.79
0.2	–	True	DeepClean	0.9729	0.9827	0.8974	0	0.8206	77.07
0.2	–	False	DeepClean	0.9700	0.9837	0.8985	0	0.8169	77.70
0.4	–	True	DeepClean	0.8863	0.9524	0.8759	0	0.5792	74.11
0.4	–	False	DeepClean	0.8868	0.9454	0.8676	0	0.5748	74.04