NoT: Federated Unlearning via Weight Negation

Supplementary Material

9. Co-Adaptation

The term *co-adaptation* originates from evolutionary biology and describes phenomena where living entities—from proteins to entire species—are not merely adapted in isolation but within an environment composed of other living entities. The entities are adapted to an ecological niche, which itself evolves through the adaptations of other entities. As a result, multiple living entities adapt together through co-evolution, leading to a state of co-adaptation [15].

In computer science, the term has been adopted more loosely to describe scenarios where different agents—such as human-machine interfaces [41], robots [55], software components [33], network layers [53], and artificial neurons [24, 62]—are not considered optimal in isolation but as part of a multi-agent system.

Our use of this terminology requires further clarification beyond this intuition. A typical neural network trained via gradient descent to minimize a loss function is considered optimal with respect to the given loss. This optimality is defined over the parameter space of a fixed architecture. However, in our work, we perturb models layer-wise rather than as a whole. The optimality of an individual layer is not meaningful in isolation because it does not minimize the loss independently; instead, it contributes to loss minimization in conjunction with other layers.

In our work, we manipulate parameters from multiple optimal models, introducing the following definitions:

Definition 3 (Grafting)**.** *Let $\mathcal{N}^{\theta}$ denote a model with a set of layers $\mathscr{L}$, where $\theta^{(i)}$ represents a set of parameters.*

A grafting of $\left(\mathcal{N}^{\theta^{(i)}}\right)_{i\in I}$ is a set of parameters $\overline{\theta}$ such that:

$$\forall \ell \in \mathscr{L}, \exists i \in I, \overline{\theta}_\ell = \theta_\ell^{(i)}. \tag{6}$$

Definition 4 (Co-Adapted Layers)**.** *Let $\mathcal{L}$ be a loss, $\mathcal{N}^{\theta}$ a model with layer set $\mathscr{L}$, and $\overline{\theta}$ a grafting of models.*

Layer parameters $\overline{\theta}_\ell$ for $\ell \in \mathscr{L}$ are said to be co-adapted if $\mathcal{N}^{\overline{\theta}}$ is optimal with respect to $\mathcal{L}$.

By abuse of language, we refer to the layers themselves as co-adapted if their parameters are co-adapted.

If the grafting originates from a singleton of an optimal model ($|I| = 1$), the layers are trivially co-adapted. In our study, we perturb the layers of a model to ensure layer-wise optimality is preserved, while creating a non-optimal grafting of multiple optimal models. This process leads to layers losing their co-adaptation.

10. NoT Algorithm

Algorithm 1 details our proposed unlearning method, which leverages weight negation.

Algorithm 1 NoT

Input: Initialize global model $\mathcal{N}$. Each client $k \in \mathcal{P} := \{1, ..., n\}$ has data D^k. Current round is denoted as τ, number of local iterations I, and learning rate η.
\\ Client Side
for client $k \in \mathcal{P}$ **do**
    Client k decides a target set $D_u^k \subset D^k$ and defines $D_r^k := D^k \setminus D_u^k$.
    if client k has $D_u^k \neq \emptyset$ **then**
        Send unlearning request to server.
\\ Server Side
if unlearning request from clients $\mathcal{K} \subset \mathcal{P}$ **then**
    Negate parameters of selected layers in $\mathcal{N}$.
    for communication round $\tau = 1, \cdots, \mathcal{T}$ **do**
        $\mathcal{M} = \{\}$
        for client $k \in \mathcal{P} \setminus \mathcal{K}$ **do**
            $\mathcal{M}[k] \leftarrow \text{CLIENTUPDATE}(\mathcal{N}, k, I)$.
        $\mathcal{N} \leftarrow$ Aggregate $\mathcal{M}$
\\ Client Side
function CLIENTUPDATE($\mathcal{N}, k, I$)
    for local iteration $i = 1, ..., I$ **do** ▷ Local training
        for minibatch $B^k \in$ local data D_r^k **do**
            $\mathcal{N} \leftarrow \mathcal{N} - \eta \nabla \mathcal{L}_{B^k}(\mathcal{N})$
    return $\mathcal{N}$

11. PyTorch Code

The PyTorch implementation of the main component of NoT, which applies layer-wise negation to a model.

```python
import torch

def negate_layers(model, layer_indicies=[0]):
    '''
    Negates the layer-wise parameters of a model
    by layer indicies
    Args:
        model: The model that needs to be
    unlearned
        layer_indicies: Layer indicies to be
    negated
    returns 'unlearned model'
    '''
    with torch.no_grad():
        for layer_indx, param in enumerate(model.
    parameters()):
            if layer_indx in layer_indicies:
                param.data = -1 * param.data
    return model
```

12. Extra Theoretical Results and Mathematical Proofs

12.1. Unlearning Time Lower Bound

Theorem 5. *Let $\mathcal{N}^\theta$ be a model, and let (D_r, D_u) denote a pair of datasets. Given an initial parameter set $\theta^0 \in \Theta$, assume $\mathcal{N}^{\theta^0}$ is trained using Stochastic Gradient Langevin Descent (SGLD) to minimize $\mathcal{L}_{D_r}$ starting from θ^0. The parameter evolution is described as: $d\theta^t = -\nabla_{\theta^t}\mathcal{L}dt + \Sigma(\theta^t, t)\cdot dW$, where training time $t \geq 0$. Then the following holds:*

$$t \geq \frac{\mathbb{E}(\delta(\theta^t) - (\theta^0))^2}{L^2\left[|\mathcal{L}_{D_r}(\theta^0) - \mathbb{E}\mathcal{L}_{D_r}(\theta^t)| + A\right]}, \tag{7}$$

where:

$$L := \sup_{\theta_1 \neq \theta_2} \frac{|\delta(\theta_1) - \delta(\theta_2)|}{\|\theta_1 - \theta_2\|} \tag{8}$$

$$A = \frac{1}{2}\int_0^t \left|\mathrm{Tr}\left(\Sigma(\theta^s, s)^2 \cdot \nabla^2 \mathcal{L}_{D_r}(\theta^s)\right)\right| ds \tag{9}$$

Proof. To begin with, let us prove the formula in the case of Deterministic Gradient Descent ($\Sigma = 0$). We have:

$$d\theta^t = -\nabla\mathcal{L}_{D_r}(\theta^t)dt \tag{10}$$

$$d\mathcal{L}_{D_r}(\theta^t) = -\nabla\mathcal{L}_{D_r}(\theta^t)\cdot\nabla\mathcal{L}_{D_r}(\theta^t)dt \tag{11}$$

$$= -\|\nabla\mathcal{L}_{D_r}(\theta^t)\|^2 dt \tag{12}$$

$$\|\delta(\theta^t) - \delta(\theta^0)\| \leq \|\theta^t - \theta^0\|\|\delta\|_{\mathrm{Lip}} \tag{13}$$

where $\|\delta\|_{\mathrm{Lip}} = \sup_{\theta_1\neq\theta_2}\frac{|\delta(\theta_1)-\delta(\theta_2)|}{\|\theta_1-\theta_2\|}$.

Then,

$$\|\theta^t - \theta^0\| = (\theta^t - \theta^0)\cdot\frac{(\theta^t - \theta^0)}{\|(\theta^t - \theta^0)\|} \tag{14}$$

$$= \int_0^t (-\nabla\mathcal{L}_{D_r}(\theta^s))\cdot\frac{(\theta^t - \theta^0)}{\|(\theta^t - \theta^0)\|}ds \tag{15}$$

$$\leq \sqrt{\underbrace{\int_0^t \|\nabla\mathcal{L}_{D_r}(\theta^s)\|^2 ds}_{=|\mathcal{L}_{D_r}(\theta^t)-\mathcal{L}_{D_r}(\theta^0)|}\underbrace{\int_0^t \left\|\frac{\theta^t - \theta^0)}{\|(\theta^t - \theta^0)\|}\right\|^2 ds}_{=t}} \tag{16}$$

Therefore,

$$\|\theta^t - \theta^0\|^2 \leq t|\mathcal{L}_{D_r}(\theta^t) - \mathcal{L}_{D_r}(\theta^0)|, \tag{17}$$

and then:

$$\frac{\|\delta(\theta^t) - \delta(\theta^0)\|}{\|\delta\|_{\mathrm{Lip}}|\mathcal{L}_{D_r}(\theta^t) - \mathcal{L}_{D_r}(\theta^0)|} \leq t. \tag{18}$$

The third line is obtained applying Cauchy-Schwarz inequality in the Hilbert space $L^2([0,t],\mathbb{R}^n)$. The result is proved for $\Sigma = 0$. When $\Sigma \neq 0$, we have:

$$d\theta^t = -\nabla\mathcal{L}_{D_r}(\theta^t)dt + \Sigma(\theta^t, t)\cdot dW \tag{19}$$

$$d\mathcal{L}_{D_r}(\theta^t) = -\|\nabla\mathcal{L}_{D_r}(\theta^t)\|^2 dt + \nabla\mathcal{L}_{D_r}\cdot\Sigma(\theta^t,t)\cdot dW + \frac{1}{2}\mathrm{Tr}\left(\Sigma(\theta^t,t)^2\cdot\nabla^2\mathcal{L}_{D_r}(\theta^t)\right)dt \tag{20}$$

$$\|\delta(\theta^t) - \delta(\theta^0)\| \leq \|\theta^t - \theta^0\| \times \|\delta\|_{\mathrm{Lip}} \tag{21}$$

Where the second line is obtained from the first applying Itô Formula (Chapter IV in [50]). The computation then unfolds the same way, then taking the expectancy and absolute value.

□

12.2. Activation Distance Maximization

Maximizing the distance d between activations serves as a straightforward heuristic for increasing loss. Specifically, for a layer ℓ activated by a function σ, denoting by ℓ' a replacement for ℓ, we aim to maximize $d(\sigma\ell x, \sigma\ell' x)$ while ensuring that $\|\sigma\ell x\| \simeq \|\sigma\ell' x\|$. We deem reasonable to assume that the later is approximately satisfied: for wide networks ℓx is close to a centered Gaussian distribution at initialization and our main case of interest is negation which preserves such Gaussian distributions. Let us consider specific cases for different activation functions.

1. **ReLU Activation ($\sigma : a \mapsto \max(0, a)$):** For any $y_0 \in \mathbb{R}^n_+$, if at least one coordinate of y_0 is non-zero, then
$$\max_{\|y\|=\|y_0\|, y\geq 0} d(y_0, y)^2 = 2\|y_0\|^2. \tag{22}$$
If all coordinates of y_0 are positive, then
$$\max_{\|y\|=\|y_0\|, y\geq 0} d(y_0, y)^2 = \|y_0\|^2. \tag{23}$$
The maximum is reached when y is orthogonal to y_0. Thus, the optimal scenario involves orthogonalizing the post-nonlinearity activations.
2. **Binary Step Function ($\sigma : a \mapsto \mathbf{1}_{a>0} + \frac{1}{2}\mathbf{1}_{a=0}$):** this function approximates sigmoid and similar variations. Here, the maximum distance is achieved by applying the Boolean negation to y_0. Notably, **not** $\sigma(\ell x) = \sigma(-\ell x)$, setting $\ell' = -\ell$ ensures the desired property.[7]
3. **Hyperbolic Tangent (or any Odd Activation Function):** For such activations, the maximum distance is attained by $y = -y_0$. Since $\forall x, \|\sigma(\ell x)\| = \|\sigma(-\ell x)\|$ for odd functions, setting $\ell' = -\ell$ satisfies the requirement.

Regarding the ReLU case, we have the following Lemma which implies Theorem 2:

Lemma 1. *Denote $\sigma(x) := \max(x, 0)$ and let $Y \in \mathbb{R}^n$ be a random vector. Assume $\mathbb{E}\left|\|\sigma(Y)\|^2 - \|\sigma(-Y)\|^2\right| \leq \varepsilon$*

[7] By extension of the boolean operator we define $\mathbf{not}(x) := 1 - x$

then:

$$\mathbb{E}\|\sigma(Y)-\sigma(-Y)\|^2 \geq \mathbb{E}\left[\max_{y\geq 0,\|y\|=\|\sigma(Y)\|} d(\sigma(Y),y)^2\right]-\varepsilon. \tag{24}$$

Proof.

$$\begin{aligned}\mathbb{E}\|\sigma(Y)-\sigma(-Y)\|^2 = \quad & \mathbb{E}\mathbf{1}_{Y\geq 0}\left[\|\sigma(Y)\|^2+\|\sigma(-Y)\|^2\right] \\ & +\mathbb{E}\mathbf{1}_{Y\not\geq 0}\left[\|\sigma(Y)\|^2+\|\sigma(-Y)\|^2\right]\end{aligned} \tag{25}$$

Define I_1, I_2 the terms on the right hand side. We have:

$$\begin{aligned} I_1 &= \mathbb{E}\mathbf{1}_{Y\geq 0}\left[\|\sigma(Y)\|^2+\|\sigma(-Y)\|^2\right] && (26)\\ &= \mathbb{E}\mathbf{1}_{Y\geq 0}\|\sigma(Y)\|^2 && (27)\\ &= \mathbb{E}\mathbf{1}_{Y\geq 0}\max_{\|y\|=\|\sigma(Y)\|,y\geq 0} d(\sigma(Y),y)^2 && (28)\\ & && (29)\\ I_2 &= \mathbb{E}\mathbf{1}_{Y\not\geq 0}\left[\|\sigma(Y)\|^2+\|\sigma(-Y)\|^2\right] && (30)\\ &= \mathbb{E}\mathbf{1}_{Y\not\geq 0}\left[\|\sigma(-Y)\|^2-\|\sigma(Y)\|^2\right] \\ &\quad +\mathbb{E}\mathbf{1}_{Y\not\geq 0}2\|\sigma(Y)\|^2 && (31)\\ &\geq \mathbb{E}\mathbf{1}_{Y\not\geq 0}\max_{\|y\|=\|\sigma(Y)\|,y\geq 0} d(\sigma(Y),y)^2-\varepsilon && (32)\end{aligned}$$

Substituting Equation (29) and Equation (32) in Equation (25) yields the desired result. □

Alternative perturbations approaches:

• **Orthogonal Linear Transformations:** Applying an orthogonal linear transformation to ℓ post-activation seems promising. However, ensuring the transform works well across all activation values ℓx is generally infeasible if the activation distribution contains a basis of the output vector space.

• **Weight Randomization:** Randomizing the weights of a dense or convolution layer (ℓ) produces ℓ'. If X being the random variable input to the layer, the random variables ℓX and $\ell' X$ have low correlation, especially in high dimensions. Therefore, we expect the expected distance $\mathbb{E}d(\sigma\ell x,\sigma\ell' x)^2$ to be close to $\mathbb{E}d(\sigma Y_1,\sigma Y_2)^2$, where Y_1 and Y_2 are independent. However, this approach is less effective, as demonstrated in the following lemma:

Lemma 2. *Denote $\sigma(x) := \max(x,0)$ and let Y_1, Y_2 be IID standard Gaussian random arrays.*

$$\mathbb{E}\|\sigma(Y_1)-\sigma(Y_2)\|^2 = \alpha\mathbb{E}\left[\max_{y\geq 0,\|y\|=\|\sigma(Y_1)\|} d(\sigma(Y_1),y)^2\right], \tag{33}$$

with $\alpha = \left(1-\frac{1}{\pi}\right) \in [0,1[$.

12.3. Jacobian Bound

In this section, we only consider feedforward models consisting of dense layers, concatenations, splits and non-trainable activation functions. By feedforward, we mean that the computational graph is Directed Acyclic, not necessarily sequential.

The assumption on layers is not very restrictive: one may formally rewrite most common architectures using only dense layers with tied weights. We write bounds assuming non-tied weights, but they may be generalized, noticing that the gradient with respect to some tied weights is the sum of the tied layers of the untied gradients.

Let E, F be finite dimensional normed spaces. Let us recall that the Sobolev space $W^{k,\infty}(E,F)$ is the set of function $E\to F$ whose derivatives up to order k are L^∞. Define $W_1^{k,\infty}(E,F)$ the set of functions whose differential is in $W^{k-1,\infty}(E,E^*\otimes F))$. For functions defined on a convex open subset of a finite-dimensional normed space E to another F, we define the semi-norms:

$$\|f\|_{\mathrm{Lip}} := \sup_{x\neq y}\frac{\|f(x)-f(y)\|}{\|x-y\|}, \tag{34}$$

For functions in $\mathrm{Lip}+L^\infty$ we define:

$$\|f\|_{\mathrm{Lip},\infty} := \inf_{g+h=f}\max\left(\|g\|_{\mathrm{Lip}};2\|h\|_{L^\infty}\right). \tag{35}$$

All models are considered as a function of the variable x, so Lipchitz and L^∞ norms are computed with respect to the x input, not the parameters θ. Let us begin with some elementary bounds.

Lemma 3. *Let E, F be two finite dimensional normed spaces and let $\mathcal{U}$ be an open subset of E. Let $f:\mathcal{U}\to\mathbb{F}$ and let $\mathcal{P}(\mathcal{U}),\mathcal{P}(F)$ be the space of probability distributions having a first moment on $\mathcal{U}$ and F respectively (for the Borel σ-algebra). Then $f^*:\mathcal{P}(\mathcal{U})\to\mathcal{P}(F),\mu\mapsto f\#\mu$ is $\|f\|_{\mathrm{Lip}}$-Lipchitz for the metric $\mathcal{W}$ and 1-Lipchitz for the metric* TV.

Proof. On the one hand, for any coupling π of μ,ν we have:

$$\begin{aligned}\mathcal{W}(f\#\mu,f\#\nu) &\leq \mathbb{E}_{(x,y)\sim\pi}\|f(x)-f(y)\| && (36)\\ &\leq \|f\|_{\mathrm{Lip}}\mathbb{E}_{(x,y)\sim\pi}\|x-y\| && (37)\end{aligned}$$

We may take π so that $\mathbb{E}_{(x,y)\sim\pi}\|x-y\| = \mathcal{W}(\mu,\nu)$.

On the other hand, for any coupling π of μ,ν:

$$\begin{aligned}\mathrm{TV}(f\#\mu,f\#\nu) &\leq \mathbb{E}_{(x,y)\sim\pi}\mathbf{1}_{f(x)\neq f(y)} && (38)\\ &\leq \mathbb{E}_{(x,y)\sim\pi}\mathbf{1}_{x\neq y} && (39)\end{aligned}$$

We may take π so that $\mathbb{E}_{(x,y)\sim\pi}\mathbf{1}_{x\neq y} = \mathrm{TV}(\mu,\nu)$. □

Lemma 4. *Let $f\in L^\infty(\mathcal{U},F)$ for some $\mathcal{U}\subset E$ open. Then*

$$f^*:(\mathcal{P}(\mathcal{U}),\mathrm{TV})\to(\mathcal{P}(F),\mathcal{W}) \tag{40}$$

is $2\|f\|_\infty$-Lipchitz.

Proof. See proof of Lemma 7. □

Definition 5. *Let $(A, \leq)$ be a partially ordered set (poset). Define the length of A as the longest chain of $(A, \leq)$:*

$$\mathrm{len}(A) := \max\{|\gamma| \ : \ \textit{totally ordered } \gamma \subset A\}. \tag{41}$$

We also define the diameter of A as the number of distinct maximal chains in A:

$$\begin{aligned} &\Lambda(A) := |\Gamma| \quad ; \\ &\Gamma(\mathscr{L}) := \{\gamma \subset A \mid \gamma \textit{ totally ordered and maximal}\}. \end{aligned} \tag{42}$$

Definition 6. *Let $\mathcal{N}^\theta$ be a feedforward neural net of layer poset $(\mathscr{L}, \leq)$. Define*

$$\|\theta\|_{2,\mathscr{L}} := \max_{\gamma \in \Gamma(\mathscr{L})} \left(\prod_{i=1}^{|\gamma|} \|\theta_{\gamma_i}\|_2 \right)^{1/|\gamma|}. \tag{43}$$

Lemma 5. *Let $\mathcal{N}^\theta$ be a feedforward neural net of layer poset $(\mathscr{L}, \leq)$ and activation function set $\Sigma \subset \mathrm{Lip}$. Assume that layers are either dense, split, or concatenation, then*

$$\|\mathcal{N}^\theta\|_{\mathrm{Lip}} \leq \Lambda(\mathscr{L}) \max\left(\|\Sigma\|_{\mathrm{Lip}} \|\theta\|_{2,\mathscr{L}}, 1\right)^{\mathrm{len}(\mathscr{L})} \tag{44}$$

with $\|\Sigma\|_{\mathrm{Lip}} := \max_{\sigma \in \Sigma} \|\sigma\|_{\mathrm{Lip}}$.

Proof. Under these assumptions, the output of network may be rewritten as a projection of the concatenation of the output of sequential networks following the layers along maximal chains. Therefore, with Γ the set of maximal chains of $\mathscr{L}$, for any $x \in \mathbb{R}^{d_{\mathrm{in}}}$

$$\|\mathcal{N}^\theta\|_{\mathrm{Lip}} \leq \sum_{\gamma \in \Gamma} \|\sigma_{\gamma_{|\gamma|}} \theta_{\gamma_{|\gamma|}} \sigma_{\gamma_{|\gamma|-1}} \theta_{\gamma_{|\gamma|-1}} \cdots \sigma_{\gamma_1} \theta_{\gamma_1}\|_{\mathrm{Lip}} \tag{45}$$

$$\begin{aligned} &\leq \sum_{\gamma \in \Gamma} \big(\|\sigma_{\gamma_{|\gamma|}}\|_{\mathrm{Lip}} \|\theta_{\gamma_{|\gamma|}}\|_{\mathrm{Lip}} \|\sigma_{\gamma_{|\gamma|-1}}\|_{\mathrm{Lip}} \\ &\quad \times \|\theta_{\gamma_{|\gamma|-1}}\|_{\mathrm{Lip}} \cdots \|\sigma_{\gamma_1}\|_{\mathrm{Lip}} \|\theta_{\gamma_1}\|_{\mathrm{Lip}} \big) \end{aligned} \tag{46}$$

$$\leq \sum_{\gamma \in \Gamma} \|\Sigma\|_{\mathrm{Lip}}^{|\gamma|} \prod_{i=1}^{|\gamma|} \|\theta_{\gamma_i}\|_2 \tag{47}$$

$$\leq \sum_{\gamma \in \Gamma} \|\Sigma\|_{\mathrm{Lip}}^{|\gamma|} \|\theta\|_{2,\mathscr{L}}^{|\gamma|} \tag{48}$$

$$\leq |\Gamma| \max\left(1, \|\Sigma\|_{\mathrm{Lip}} \|\theta\|_{2,\mathscr{L}}\right)^{\mathrm{len}(\mathscr{L})}. \tag{49}$$

The results follows for $\Lambda(\mathscr{L}) := |\Gamma|$. □

Lemma 6. *Let $\mathcal{N}^\theta$ be a feedforward neural net of layer poset $(\mathscr{L}, \leq)$ and activation function set $\Sigma \subset \mathrm{Lip}$. Assume $\mathcal{N}^\theta$ is defined on some open bounded domain $\mathcal{U}$ and assume that layers are either dense, split, or concatenation, then for any layer ℓ:*

$$\|\nabla_{\theta_\ell} \mathcal{N}^\theta\|_\infty \leq \sup_{x \in \mathcal{U}} \|x\| \Lambda(\mathscr{L}) \max\left(\|\Sigma\|_{\mathrm{Lip}} \|\theta\|_{2,\mathscr{L}}, 1\right)^{\mathrm{len}(\mathscr{L})} \tag{50}$$

with $\|\Sigma\|_{\mathrm{Lip}} := \max_{\sigma \in \Sigma} \|\sigma\|_{\mathrm{Lip}}$.

Proof. Proceed as for Lemma 5. □

Theorem 6. *Assume $\mathcal{N}^\theta$ is given by a Lipchitz feedforward neural network having also Lipchitz derivative of the layer set $\mathscr{L}$ and let $\leq$ be the order relation on $\mathscr{L}$ induced by the computational graph. Let $J_\ell^\theta := \nabla_{\theta_\ell} \mathcal{N}^\theta(X)$ be its Jacobian on $X \sim \mathcal{D}$ for layer $\ell \in \mathscr{L}$, and let Y be the concatenation of the outputs of the layers $\ell \in \mathscr{L}_{\mathrm{neg}}$. Assume that $\mathscr{L}_{\mathrm{neg}}$ is a Cauchy domain of $\mathscr{L}$ (i.e., $\mathscr{L}_{\mathrm{neg}}$ intersects exactly once every maximal totally ordered subsets of $\mathscr{L}$), then:*

$$\mathcal{W}\left(J_\ell^\theta; J_\ell^{\theta'}\right) \leq A_\ell \ \mathrm{TV}(Y_-; -Y_-), \qquad \forall \ell > \mathscr{L}_{\mathrm{neg}} \tag{51}$$

$$\mathcal{W}\left(\epsilon J_\ell^\theta; J_\ell^{\theta'}\right) \leq A_\ell \ \mathrm{TV}((X, Y_-); (X, -Y_-)), \quad \forall \ell \leq \mathscr{L}_{\mathrm{neg}} \tag{52}$$

where $\epsilon = (-1)^{\ell \notin \mathscr{L}_{\mathrm{neg}}}$, $\mathcal{W}$ and TV are the Wasserstein and total variation distances, and $(A_\ell)_{\ell \in \mathscr{L}}$ are positive constants depending on θ and the support of X.

Proof. Define $\theta_{\neq \mathrm{neg}} = (\theta_\ell)_{\ell \notin \mathscr{L}_{\mathrm{neg}}}$ and for layer subset $\mathscr{L}_1 \leq \mathscr{L}_1$ denote by $\mathcal{N}^\theta_{\mathscr{L}_1 \to \mathscr{L}_2}$ the sub-model of $\mathcal{N}^\theta$ taking as input the concatenation of the outputs of layers in $\mathscr{L}_1$ and outputs the concatenation of the output of layers in $\mathscr{L}_2$. We take the convention that if $\mathscr{L}_1 = \emptyset$ means the input is the input of the model $\mathcal{N}^\theta$ and $\mathscr{L}_2 = \emptyset$, the output is the output of model $\mathcal{N}^\theta$. With this convention, $\mathcal{N}^\theta_{\emptyset \to \emptyset} = \mathcal{N}^\theta$. The sub-model $\mathcal{N}^\theta_{\mathscr{L}_1 \to \mathscr{L}_2}$ is well defined if $\mathscr{L}_1$ is a Cauchy subset but may not be otherwise. Note that differentiation may only be taken with respect to the weights of a dense layer since split, concatenation and activation functions don't have weights under our assumptions.

First, let $\ell > \mathscr{L}_{\mathrm{neg}}$. We denote by X' a random variable having the same law as X but with a coupling to X to be

chosen later. Since $\mathscr{L}_{\text{neg}}$ is a Cauchy domain:

$$\|J_\ell^\theta - J_\ell^{\theta'}\| := \left\| \nabla_{\theta_\ell} \left(\mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset} \mathcal{N}^\theta_{\emptyset\to\mathscr{L}_{\text{neg}}} \right)\Big|_{x=X} - \nabla_{\theta_\ell} \left(\mathcal{N}^{\theta'}_{\mathscr{L}_{\text{neg}}\to\emptyset} \mathcal{N}^{\theta'}_{\emptyset\to\mathscr{L}_{\text{neg}}} \right)\Big|_{x=X'} \right\| \tag{53}$$

$$= \left\| \left(\nabla_{\theta_\ell} \mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset} \right)\Big|_{x=\mathcal{N}^\theta_{\emptyset\to\mathscr{L}_{\text{neg}}}(X)} - \left(\nabla_{\theta_\ell} (\mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset} \right)\Big|_{x=\mathcal{N}^{\theta'}_{\emptyset\to\mathscr{L}_{\text{neg}}}(X')} \right\| \tag{54}$$

By Lemmata 3 and 4 we get:

$$\mathcal{W}\left(J_\ell^\theta \,;\, J_\ell^{\theta'}\right) \le 2\|\nabla_{\theta_\ell}\mathcal{N}^{\theta'}_{\mathscr{L}_{\text{neg}}\to\emptyset}\|_\infty \tag{55}$$

$$\times \,\text{TV}(\mathcal{N}^\theta_{\emptyset\to\mathscr{L}_{\text{neg}}}(X) \,;\, \mathcal{N}^{\theta'}_{\emptyset\to\mathscr{L}_{\text{neg}}}(X')) \tag{56}$$

$$= 2\|\nabla_{\theta_\ell}\mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset}\|_\infty \text{TV}(Y_-; -Y_-) \tag{57}$$

Then, by Lemma 6, we may set:

$$A_\ell = 2\left(\sup_{x\in\mathcal{U}}\|x\|\right)\Lambda(\mathscr{L}^+_{\text{neg}})\max\left(\|\Sigma\|_{\text{Lip}}\|\theta\|_{2,\mathscr{L}},1\right)^{\text{len}(\mathscr{L}^+_{\text{neg}})} \tag{58}$$

with $\mathscr{L}^+_{\text{neg}} := \{\ell' \in \mathscr{L} \mid \ell' > \mathscr{L}_{\text{neg}}\}$.

Second, let $\ell < \mathscr{L}_{\text{neg}}$ and proceed in the same way:

$$\|J_\ell^\theta + J_\ell^{\theta'}\| := \left\|\nabla_{\theta_\ell}\left(\mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset}\mathcal{N}^\theta_{\emptyset\to\mathscr{L}_{\text{neg}}}\right)(X) + \nabla_{\theta_\ell}\left(\mathcal{N}^{\theta'}_{\mathscr{L}_{\text{neg}}\to\emptyset}\mathcal{N}^{\theta'}_{\emptyset\to\mathscr{L}_{\text{neg}}}\right)(X')\right\| \tag{59}$$

$$= \left\| \nabla_x \mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset}\Big|_{x=Y_-} \nabla_{\theta_\ell}\mathcal{N}^\theta_{\emptyset\to\mathscr{L}_{\text{neg}}}(X) - \nabla_x \mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset}\Big|_{x=-Y'_-} \nabla_{\theta_\ell}\mathcal{N}^\theta_{\emptyset\to\mathscr{L}_{\text{neg}}}(X') \right\| \tag{60}$$

Beware that this time, the terms can not be rewritten as a function of only Y_-. We thus apply Lemma 3 to the whole term as a function of the couple variable (X, Y_-):

$$(x_0, y_0) \mapsto \left(\nabla_x \mathcal{N}^\theta_{\mathscr{L}_{\text{neg}}\to\emptyset}\Big|_{x=y_0} \right)\left(\nabla_{\theta_\ell}\, \mathcal{N}^\theta_{\emptyset\to\mathscr{L}_{\text{neg}}}\Big|_{x=x_0} \right). \tag{61}$$

Then, Lemmata 6 and 5, allow to conclude the same way with:

$$A_\ell = \left(\sup_{x\in\mathcal{U}}\|x\|\right)\Lambda(\mathscr{L})\max\left(\|\Sigma\|_{\text{Lip}}\|\theta\|_{2,\mathscr{L}},1\right)^{\text{len}(\mathscr{L})} \tag{62}$$

Finally, the case $\ell \in \mathscr{L}_{\text{neg}}$ is treated the same way. □

One may also prove similar bounds replacing the total variation by Wasserstein distance. However, the constant A_ℓ then depends on the Lipchitz norm of the derivative of the activation functions. If the activation functions do not have Lipchitz derivative, total variation is necessary as we only have the following Lemma to control the Wasserstein distance of output distributions.

Lemma 7. *Let $f \in \text{Lip}(\mathcal{U}, F) + L^\infty(\mathcal{U}, F)$ for some $\mathcal{U} \subset E$ open. Then,*

$$f^* : (\mathcal{P}(\mathcal{U}), \mathcal{W} + \text{TV}) \to (\mathcal{P}(F), \mathcal{W}) \tag{63}$$

is $\|f\|_{\text{Lip},\infty}$-Lipchitz.

Proof. For any coupling π of μ, ν and any decomposition $f = g + h$ with $g \in W_1^{1,\infty}$ and $h \in W_{0,\text{LBV}}$, with $(x, y) \sim \pi$ we have:

$$\mathbb{E}\|f(x) - f(y)\| \le \mathbb{E}\left[\|g(x) - g(y)\| + \|h(x) - h(y)\|\right] \tag{64}$$

$$\le \mathbb{E}\left[\|g\|_{\text{Lip}}\|x - y\| + \mathbf{1}_{x\neq y}\|h\|_\infty\right] \tag{65}$$

Since the inequality is true for any coupling π, we may choose a coupling of the form $\pi = D\#\min(\mu,\nu) + \pi'$ with $D : x \mapsto (x, x)$ and π' a coupling between $(\mu-\nu)^+$ and $(\mu-\nu)^-$. Here $\mu \wedge \nu := \min\left(\frac{d\mu}{d(\mu+\nu)} \,;\, \frac{d\nu}{d(\mu+\nu)}\right) \times (\mu+\nu)$ where the derivative denote the Radon-Nikodym derivative (see [7] p125). We then have:

$$\mathbb{E}\left[\mathbf{1}_{x\neq y}\|h\|_\infty\right] \quad = \quad 2\|h\|_\infty \text{TV}(\mu,\nu) \tag{66}$$

and since $\inf_{\pi'}\mathbb{E}\left[\|x-y\|\right] = \mathcal{W}((\mu-\nu)^+, (\mu-\nu)^-)$. Taking the infimum over $g + h = f$ and adding TV on both sides we get:

$$\mathcal{W}(f\#\mu, f\#\nu) \le \inf_{f=g+h}\left(\|g\|_{\text{Lip}}\mathcal{W}((\mu-\nu)^+ \,;\, (\mu-\nu)^-) + (1 + 2\|h\|_{L^\infty})\text{TV}(\mu,\nu)\right). \tag{67}$$

Note that by cyclical monotonicity of the optimal transport plan (see [60] pp79-80), for all positive measures α, β, γ we have $\mathcal{W}(\alpha+\beta \,;\, \alpha+\gamma) \ge \mathcal{W}(\alpha,\beta)$. Therefore,

$$\mathcal{W}(\mu,\nu) = W(\mu\wedge\nu + (\mu-\nu)^+ \,;\, \mu\wedge\nu + (\mu-\nu)^-) \ge \mathcal{W}((\mu-\nu)^+ \,;\, (\mu-\nu)^-) \tag{68}$$

so,

$$\mathcal{W}(f\#\mu, f\#\nu) \le \inf_{f=g+h}\max(\|g\|_{\text{Lip}}, (1 + 2\|h\|_{L^\infty})) \times (\mathcal{W}(\mu \,;\, \nu) + \text{TV}(\mu,\nu)). \tag{69}$$

Finally,

$$\mathcal{W}(f\#\mu, f\#\nu) \le (1 + \|f\|_{\text{Lip},\infty})(\mathcal{W}(\mu \,;\, \nu) + \text{TV}(\mu,\nu)). \tag{70}$$

□

12.4. Layer-Wise Optimality

We introduce a new metric to quantify how far a layer of a model is from being effectively pretrained, as discussed in Section 4.2: how much the layer has to be modified to become a layer with an optimal set of weights?

Definition 7 (Layer-wise optimality norm)**.** *Let $\mathcal{N}^\theta$ denote a model parameterized by $\theta \in \Theta$, and let ℓ be a layer of $\mathcal{N}^\theta$ with parameters θ_ℓ. The layer-wise optimality norm of layer ℓ is defined as:*

$$\|\theta\|_{*,\ell} := \inf_{\alpha\in\mathbb{R}_+,\theta^*\in\Theta^*} \|\theta_\ell - \alpha\theta^*_\ell\|_2, \tag{71}$$

where θ^* represents the set of optimal parameters.

For a Lipchitz transformations $\sigma : \Theta \to \Theta$, acting solely on layer ℓ, the layer-wise optimality norm of σ is given by:

$$\|\sigma\|_{*,\ell} := \inf_{A>0} \sup_{\theta\in\Theta} \left(\|\mathcal{N}^{\sigma(\theta)}\|_{*,\ell} - A\|\mathcal{N}^\theta\|_{*,\ell} \right). \tag{72}$$

Definition 8. *Let $\mathcal{N}^\theta$ be a model of layer set $\mathscr{L}$. A Lipchitz Lipchitz transformations $\sigma : \Theta \to \Theta$ is LWO-Lipchitz if:*

$$\|\sigma\|_* := \sum_{\ell\in\mathscr{L}} \|\sigma\|_{*,\ell} = 0. \tag{73}$$

Remarks. ❶ These definitions are meaningful only if $\Theta^* \neq \emptyset$. Strictly speaking, $\|\cdot\|_{*,\ell}$ on Θ is not a norm but the distance to a subset. ❷ The factor α ensures that the layer-wise optimality norm remains bounded, and it is reasonable since such a scaling factor is usually easy to recover via gradient descent. ❸ The norm $\|\sigma\|_{*,\ell}$ quantifies how well the transformation σ preserves the optimality of layer ℓ. LWO-Lipchitz property means σ is Lipchitz for all the the layer-wise optimality norms. Thus, it preserves layer-wise optimality quantitatively.

A low layer-wise optimality norm does not guarantee rapid convergence but implies that the layer can be frozen while the rest of the model is trained from scratch. Even if not frozen, it is expected to reduce the effective dimensionality of the space explored by gradient descent.

12.4.1. Affine Compensation of Layer Negation.

A key point is that the usual sigmoid-like or odd activation functions ψ satisfy an algebraic relation:

$$\forall x \in \mathbb{R},\ \psi(-x) + \psi(x) = C. \tag{74}$$

for some constant $C \in \mathbb{R}$.

Lemma 8. *Let ℓ_1, ℓ_2 be two linear layers, and let ψ be an activation function. If $-\ell_1$ denotes the layer obtained by negating the parameters of ℓ_1, then:*

If ψ satisfies an algebraic relation of the form:

$$\exists a, b, c \in \mathbb{R},\quad (ab \neq 0 \ \textit{and}\ \forall x \in \mathbb{R},\ a\psi(x) + b\psi(-x) = c), \tag{75}$$

there exists a linear layer ℓ_2' such that:

$$\ell_2 \circ \psi \circ \ell_1 = \ell_2' \circ \psi \circ (-\ell_1). \tag{76}$$

Furthermore, if ℓ_2 is convolutional, then ℓ_2' can also be chosen as convolutional.

Proof. Let $a, b, c \in \mathbb{R}$ satisfy the property above. Define $\ell_3(x) = c - \frac{b}{a}x$. Then $\ell_2' = \ell_2 \circ \ell_3$ satisfies the desired properties. The layer ℓ_2' is linear and convolutional if ℓ_2 is convolutional. □

Theorem 4 is a consequence of the following Theorem.

Theorem 7. *The negation perturbation is LWO-Lipchitz if $\mathscr{L}_{\text{neg}}$ is an antichain of the poset $\mathscr{L}$ containing no maximal element, and each $\ell \in \mathscr{L}_{\text{neg}}$ is activated by sigmoid-like, odd, or even functions* (e.g., $\mathbf{1}_{>0}$, $\tanh$, $\sin$, x^2)*.*

Proof. Without loss of generality we may assume that $\mathscr{L}_{\text{neg}}$ is a singleton $\mathscr{L}_{\text{neg}} = \{\ell_1\}$.

Let $\mathcal{N}^\theta$ be a feedforward neural network with a set of linear or convolution layers $\mathscr{L}$, and let ℓ_1 be a layer that is not the output layer. Denote the subsequent layer by ℓ_2, and assume the activation function ψ of ℓ satisfies: $\forall x \in \mathbb{R}, \psi(x)+\psi(-x) = Cte$ or $\forall x \in \mathbb{R}, \psi(x)-\psi(-x) = Cte$.

For any $\varepsilon > 0$, let $(\theta^*_\ell)_{\ell\in\mathscr{L}} \in \Theta^*$ such that: $\|\theta_{\ell_1} - \theta^*_{\ell_1}\| \leq \|\mathcal{N}^\theta\|_{*,\ell_1} + \varepsilon$. Define $\widetilde{\theta}^* \in \Theta$ as follows:

- $\widetilde{\theta}^*_\ell = \theta^*_\ell$ for $\ell \notin \{\ell_1, \ell_2\}$;
- $\widetilde{\theta}^*_{\ell_1} = -\theta^*_{\ell_1}$;
- $\widetilde{\theta}^*_{\ell_2}$ the parameters of the layer given by Lemma 8.

Since:

$$\ell_2(\widetilde{\theta}^*_{\ell_2}) \circ \psi \circ \ell(\widetilde{\theta}^*_{\ell_1}) = \ell_2(\theta^*_{\ell_2}) \circ \psi \circ \ell_1(\theta^*_{\ell_1}), \tag{77}$$

we deduce that $\mathcal{N}^{\theta^*} = \mathcal{N}^{\widetilde{\theta}^*}$ as functions, and hence $\widetilde{\theta}^* \in \Theta^*$. Thus:

$$\begin{aligned}
\|\mathcal{N}^{\sigma(\theta)}\|_{*,\ell_1} &\leq \|\sigma(\theta)_{\ell_1} - \widetilde{\theta}^*_{\ell_1}\| && (78)\\
&= \| - \theta_{\ell_1} - (-\theta^*_{\ell_1})\| && (79)\\
&= \|\theta_{\ell_1} - \theta^*_{\ell_1}\| && (80)\\
&\leq \|\mathcal{N}^\theta\|_{*,\ell_1} + \varepsilon. && (81)
\end{aligned}$$

As this inequality holds for all $\varepsilon > 0$, we conclude: $\|\mathcal{N}^{\sigma(\theta)}\|_{*,\ell_1} \leq \|\mathcal{N}^\theta\|_{*,\ell_1}$. Consequently, $\|\sigma\|_{*,\ell_1} = 0$. □

13. Empirical Support of Theoretical Analysis

This section presents empirical evidence supporting the hypotheses used in the theoretical analysis. Specifically, we examine the following:

1. **CKA Analysis:** Demonstrates effective pretraining and the breaking of co-adaptation.
2. **Post-Negation Fine-Tuning:** Validates the claim of dimensionality reduction during fine-tuning.
3. **Unlearning Lower Bound:** Evaluates the unlearning lower bound to show that it is constraining for natural forgetting.

13.1. Quantitative Unlearning Time Constraint

We aim to assess the accuracy of the proposed unlearning time lower bound. Recall the derived inequality:

$$t \geq \frac{\mathbb{E}(\delta(\theta^t) - \delta(\theta^0))^2}{\|\delta\|_{\text{Lip}}^2 \left[|\mathcal{L}_{D_r}(\theta^0) - \mathbb{E}\mathcal{L}_{D_r}(\theta^t)| + A\right]} := t_{\text{unlearn}}, \tag{82}$$

where t_{unlearn} represents the estimated unlearning time lower bound.

The goal is not to compute this precisely but to determine its order of magnitude. Each term in the formula can be approximated as follows:

- **Estimating δ:** δ is computable for any model through a forward pass on both the remaining and forgotten data.
- **Estimating $\delta(\theta^t) - \delta(\theta^0)$:** For a model $\mathcal{N}^{\theta^0}$, we use:

$$|\delta(\theta^t) - \delta(\theta^0)| \geq (1-\varepsilon)\left|\delta(\theta^*_{\text{Retrain}}) - \delta(\theta^0)\right|, \tag{83}$$

  where $\theta^*_{\text{Retrain}}$ represents the parameters of a model trained from scratch on D_r and $0 < \varepsilon \ll 1$.
- **Estimating $\|\delta\|_{\text{Lip}}$:** Since $\|\delta\|_{\text{Lip}} = \|\nabla_\theta \delta\|_\infty$, we estimate it by computing the gradient of $\mathcal{L}_{D_r}$ and $\mathcal{L}_{D_u}$ at various θ and taking the supremum of the norm across for these θ:

$$\|\delta\|_{\text{Lip}} \simeq \max_{b \in \text{Batch}} \|\nabla_\theta \delta(\theta^b)\| \tag{84}$$

  For a given training path, we choose to take the supremum of θ^t at different times. This gives a slightly sharper bound than taking the maximum across models, as NoT tends to have larger gradients than the other training.

$$\|\delta\|_{\text{Lip}} \simeq \max_{t \in \{t_1, \cdots, t_n\}} \|\nabla_\theta \delta(\theta^t)\| \tag{85}$$

- **Estimating $|\mathcal{L}_{D_r}(\theta^0) - \mathbb{E}\mathcal{L}_{D_r}(\theta^t)|$:** To estimate the available loss decrease, we use a lower bound given by:

$$|\mathcal{L}_{D_r}(\theta^0) - \mathbb{E}\mathcal{L}_{D_r}(\theta^t)| \leq |\mathcal{L}_{D_r}(\theta^0) - \mathcal{L}_{D_r}(\theta^*)|, \tag{86}$$

  where θ^* is obtained from a model trained from scratch on D_r for an extended period (*e.g.*, twice as long as the original training).

Table 3. Comparison of the estimated unlearning bound t_{unlearn} (right side of Equation (2)).

Dataset & Model	Method	$t_{\text{unlearn}} \downarrow$	Est. Comp. Cost (FLOPs) $\downarrow$
CIFAR-10 CNN	FT	2.46	$1.51e^{14}$
	NoT	0.04	$2.33e^{12}$
CIFAR-10 ResNet-18	FT	4.77	$3.00e^{16}$
	NoT	0.0017	$1.04e^{13}$
Caltech-101 ViT	FT	0.299	$1.74e^{19}$
	NoT	0.0006	$3.56e^{16}$

- **Estimating the Stochasticity Term:** For the term:

$$\frac{1}{2}\int_0^t \left|\text{Tr}\left(\Sigma(\theta^s, s)^2 \cdot \nabla^2 \mathcal{L}_{D_r}(\theta^s)\right)\right| ds, \tag{87}$$

  we assume $\Sigma(\theta^s, s)$ is diagonal and use Formula 8 from [56] to approximate $\Sigma(\theta^s, s)^2$. The Hessian $\nabla^2 \mathcal{L}_{D_r}$ is also approximated as diagonal, reducing the trace to a product of the diagonal elements.

Results. The compiled results are presented in Table 3, alongside computational cost estimates, for comparison with experimental results in Table 1.

Observations:

- **Stochasticity Term:** The term $\Sigma(\theta^s, s)^2$ is orders of magnitude smaller than the available loss decrease, allowing us to neglect its contribution.
- **Comparison of FT and NoT:** We obtain a significant difference between FT and NoT with a predicted cost ratio close to the experimental cost ratio.
- **Accuracy of the Bound:** The theoretical lower bound for t_{unlearn} is significantly lower than empirical results, suggesting potential for refinement by incorporating details of the gradient descent trajectory.

13.2. Client-wise FU in a Non-IID Setting

Table 4 presents results for client-wise FU in a non-IID setting, where data across the 10 clients follow a Dirichlet distribution with $\beta = 0.1$. Results show that even in non-IID setting NoT outperforms existing baselines.

13.3. Class-wise FU in an IID Setting

Table 5 presents results for class-wise FU in an IID setting, where samples of class 0 are to be forgotten from all clients. Each client sends a request to the server, then the server starts the unlearning process by applying layer-wise weight negation to the global model. Finally, several rounds of fine-tuning are done on the clients' retain data. Results in Table 5 demonstrates the effectiveness of NoT compared to baselines in class-wise FU.

Table 4. **Client-wise federated unlearning in a non-IID setting.** Performance comparison of NoT with baselines in a 10-client setup, where the first client requests unlearning. Client data follows a Dirichlet distribution with $\beta = 0.1$. The best average gap is marked in red.

Dataset & Model	Method	Accuracy (%)			Privacy (%)	Avg. Gap ↓	Cost (Bytes & FLOPs)	
		Retain (Δ ↓)	Forget (Δ ↓)	Test (Δ ↓)	MIA (Δ ↓)		Comm. ↓	Comp. ↓
Caltech-101 ViT	Retrain	$98.42_{\pm 0.45}$ (0.00)	$43.80_{\pm 2.15}$ (0.00)	$45.56_{\pm 0.61}$ (0.00)	$52.20_{\pm 1.79}$ (0.00)	0.00	$1.77e^{12}$	$9.88e^{21}$
	FT	$99.80_{\pm 0.06}$ (1.38)	$80.04_{\pm 2.18}$ (36.24)	$47.75_{\pm 0.08}$ (2.19)	$67.87_{\pm 1.73}$ (15.67)	13.87	$1.48e^{12}$	$8.24e^{21}$
	PGD	$99.72_{\pm 0.08}$ (1.30)	$75.94_{\pm 2.54}$ (32.14)	$47.25_{\pm 0.50}$ (1.69)	$65.73_{\pm 0.82}$ (13.53)	12.16	$1.48e^{12}$	$8.25e^{21}$
	MoDE	$97.87_{\pm 1.19}$ (1.19)	$47.69_{\pm 2.53}$ (6.39)	$46.81_{\pm 0.67}$ (1.71)	$51.70_{\pm 2.35}$ (3.00)	3.07	$1.64e^{12}$	$9.18e^{21}$
	FCU	$99.34_{\pm 0.02}$ (0.92)	$52.01_{\pm 0.57}$ (8.21)	$45.53_{\pm 0.57}$ (0.03)	$54.80_{\pm 1.26}$ (2.60)	2.94	$1.48e^{12}$	$9.44e^{21}$
	NoT (Ours)	$97.71_{\pm 0.41}$ (0.71)	$45.20_{\pm 2.40}$ (1.40)	$45.93_{\pm 0.54}$ (0.37)	$51.23_{\pm 1.49}$ (0.97)	0.86	$8.90e^{11}$	$4.95e^{21}$

Table 5. **Class-wise federated unlearning.** Performance comparison of NoT with baselines in a 10-client setup, where samples of class 0 are to be forgotten from all clients. The data distribution among clients is **IID**. The best average gap is marked in red.

Dataset & Model	Method	Accuracy (%)			Privacy (%)	Avg. Gap ↓	Cost (Bytes & FLOPs)	
		Retain (Δ ↓)	Forget (Δ ↓)	Test (Δ ↓)	MIA (Δ ↓)		Comm. ↓	Comp. ↓
Caltech-101 ViT	Retrain	$99.32_{\pm 0.11}$ (0.00)	$0.00_{\pm 0.00}$ (0.00)	$46.85_{\pm 0.35}$ (0.00)	$92.07_{\pm 1.47}$ (0.00)	0.00	$1.48e^{12}$	$8.24e^{21}$
	FT	$99.93_{\pm 0.03}$ (0.61)	$77.39_{\pm 2.37}$ (77.39)	$46.69_{\pm 0.24}$ (0.16)	$71.53_{\pm 0.74}$ (20.54)	24.68	$1.19e^{12}$	$6.60e^{21}$
	MoDE	$98.74_{\pm 0.89}$ (0.73)	$0.00_{\pm 0.00}$ (0.00)	$46.97_{\pm 0.42}$ (0.18)	$84.70_{\pm 6.16}$ (8.20)	2.28	$1.35e^{12}$	$7.54e^{21}$
	NoT (Ours)	$99.36_{\pm 0.10}$ (0.04)	$0.70_{\pm 0.49}$ (0.70)	$46.54_{\pm 0.17}$ (0.31)	$84.83_{\pm 1.37}$ (7.24)	2.07	$8.90e^{11}$	$4.95e^{21}$

13.4. Instance-wise FU in an IID Setting

Table 6 presents results for instance-wise FU in an IID setting. We set 10% of each client's samples to be forgotten. Similar to class-wise FU, each client requests from the the server to initiate the unlearning process. Next, the server applies layer-wise weight negation to the global model. Finally, several rounds of fine-tuning are done on the clients' retain data. Results in Table 6 demonstrates the effectiveness of NoT compared to baselines in instance-wise FU.

13.5. Backdoor Attack

We further evaluate unlearning effectiveness by removing the influence of backdoor triggers [20]. Following prior works [21, 63, 71], a fraction of the target client images is poisoned with a 3×3 pixel pattern trigger (using the Adversarial Robustness Toolbox [44]) and relabeled as 'airplane' (excluding pre-existing airplane labels). ViT and Caltech-101 are used in this experiment. A global model is then trained from scratch using all clients, including the poisoned one, until convergence. Due to the target client, the global FL model becomes vulnerable to the backdoor trigger. A successful unlearning method should eliminate the influence of the poisoned client. Table 7 shows that NoT achieves the lowest average gap with minimal communication and computation costs, outperforming baselines in removing backdoor influence.

13.6. Further CKA Analysis

Figure 3, presented CKA comparisons for a CNN model, demonstrating alignment with the theoretical analysis. Here, we extend CKA analysis to ResNet-18 and ViT models, confirming that **layer negation disrupts co-adaptation while preserving similar features**. Figure 7 shows the CKA similarity between the original (FT), random (Retrain), and layer-wise-negation (NoT) models, before (τ:0) and after (τ:-1) fine-tuning. The FT model at τ:0 serves as the reference.

- **ResNet-18:** The first convolutional layer (index ℓ:0) is negated. Activations from the ReLU following each layer and the head are analyzed.
- **ViT:** The convolutional projection layer (index ℓ:1) is negated. Focus is on Conv_Proj, Encoder_0, and Head activations.

Comparing NoT to FT at round τ:0 reveals greater divergence at deeper layers, reflecting disrupted co-adaptation. In ViT, we can see the significant CKA dissimilarity for the subsequent layer (Encoder_0) and output (Heads). However, the transformer residual connections seem to bring features back to high similarity with a decreasing trend in depth. Since this behavior is observed across perturbations and even randomized network shows similar pattern, we normalize the CKA via $\frac{\text{CKA}_Q - \text{CKA}_{Retrain@\tau:0}}{1-\text{CKA}_{Retrain@\tau:0}}$ to clarify these results, where Q is any model. After fine-tuning, Retrain and NoT (τ:-1) exhibit similar CKA but differ from FT. This demonstrates that after fine-tuning, NoT becomes closely similar to Retrain. Moreover, an indication of NoT preserving effective pretraining benefits is by comparing two random models, at τ:0, where the first layer has the original weights while the other has the negation of the original weights as in FT@τ=0. The high CKA similarity between those two models confirms the effective pretraining of negating the first layer.

Table 6. **Instance-wise federated unlearning.** Performance comparison of NoT with baselines in a 10-client setup, where 10% of each client's data is randomly selected to be forgotten. The data distribution among clients is **IID**. The best average gap is marked in red.

Dataset & Model	Method	Accuracy (%)			Privacy (%)	Avg. Gap ↓	Cost (Bytes & FLOPs)	
		Retain (Δ ↓)	Forget (Δ ↓)	Test (Δ ↓)	MIA (Δ ↓)		Comm. ↓	Comp. ↓
Caltech-101 ViT	Retrain	$99.71_{\pm 0.01}$ (0.00)	$47.58_{\pm 1.09}$ (0.00)	$47.98_{\pm 0.23}$ (0.00)	$49.70_{\pm 1.44}$ (0.00)	0.00	$1.77e^{12}$	$9.88e^{21}$
	FT	$99.96_{\pm 0.02}$ (0.25)	$96.61_{\pm 0.91}$ (49.03)	$48.91_{\pm 0.19}$ (0.93)	$75.73_{\pm 1.39}$ (26.03)	19.06	$1.48e^{12}$	$8.24e^{21}$
	MoDE	$97.48_{\pm 1.58}$ (1.02)	$51.94_{\pm 5.86}$ (2.49)	$48.16_{\pm 0.40}$ (0.35)	$51.33_{\pm 5.27}$ (0.17)	1.01	$1.05e^{12}$	$5.90e^{21}$
	NoT (Ours)	$99.85_{\pm 0.03}$ (0.14)	$50.25_{\pm 1.24}$ (2.67)	$47.97_{\pm 0.23}$ (0.01)	$50.73_{\pm 1.75}$ (1.03)	0.96	$9.19e^{11}$	$5.50e^{21}$

Table 7. **Backdoor attack in federated unlearning.** Performance comparison of NoT with baselines in a 10-client setup, where one client with 80% backdoored samples requests unlearning. The data distribution among clients is **IID**. The best average gap is marked in red.

Dataset & Model	Method	Accuracy (%)			Privacy (%)	Avg. Gap ↓	Cost (Bytes & FLOPs)	
		Retain (Δ ↓)	Forget (Δ ↓)	Test (Δ ↓)	MIA (Δ ↓)		Comm. ↓	Comp. ↓
Caltech-101 ViT	Retrain	$99.73_{\pm 0.08}$ (0.00)	$15.74_{\pm 0.34}$ (0.00)	$48.31_{\pm 0.24}$ (0.00)	$73.17_{\pm 1.55}$ (0.00)	0.00	$1.76e^{12}$	$1.37e^{21}$
	FT	$99.93_{\pm 0.02}$ (0.20)	$23.24_{\pm 0.96}$ (7.50)	$47.85_{\pm 0.07}$ (0.46)	$60.60_{\pm 0.57}$ (12.57)	5.18	$1.74e^{12}$	$1.36e^{21}$
	PGD	$99.41_{\pm 0.37}$ (0.32)	$62.82_{\pm 29.77}$ (47.08)	$45.95_{\pm 1.28}$ (2.36)	$70.07_{\pm 6.27}$ (3.10)	13.21	$1.56e^{12}$	$1.22e^{21}$
	MoDE	$98.70_{\pm 0.92}$ (1.03)	$30.81_{\pm 13.87}$ (15.07)	$45.54_{\pm 2.04}$ (2.77)	$63.55_{\pm 10.35}$ (9.62)	7.12	$1.76e^{12}$	$1.38e^{21}$
	FCU	$99.51_{\pm 0.08}$ (0.22)	$15.50_{\pm 0.53}$ (0.24)	$48.46_{\pm 0.21}$ (0.15)	$73.10_{\pm 1.13}$ (0.07)	0.17	$1.06e^{12}$	$9.98e^{20}$
	NoT (Ours)	$99.60_{\pm 0.02}$ (0.13)	$15.62_{\pm 0.17}$ (0.12)	$47.95_{\pm 0.07}$ (0.36)	$73.17_{\pm 1.38}$ (0.00)	0.15	$1.43e^{12}$	$1.11e^{21}$

13.7. Direct Test of Layer-wise Optimality

By definition, Layer-wise Optimality (LWO) requires that any layer ℓ in a model $\mathcal{N}^\theta$ can be frozen while the remaining layers are reinitialized, and the model can still be fine-tuned to achieve optimal performance. We validate this property by directly comparing the accuracy of models subjected to this process with models trained from scratch.

We test the LWO of $\mathcal{N}^{\theta'}$, defined as:

$$\theta' := (-\theta_\ell)_{\ell \in \mathscr{L}_{\text{neg}}} \oplus (\theta_\ell)_{\ell \in \mathscr{L} \setminus \mathscr{L}_{\text{neg}}}. \tag{88}$$

with $\mathcal{N} \in \{\text{CNN}, \text{ResNet-18}, \text{ViT}\}$; $\mathscr{L}_{\text{neg}}$ only contains the first layer for CNN and ResNet-18, and only the convolutional projection for ViT. In all cases, layer-wise optimality of $\ell \in \mathscr{L} \setminus \mathscr{L}_{\text{neg}}$ is obvious since they are layers of an optimal model $\mathcal{N}^\theta$. Therefore, we only need to test layer-wise optimality of $\ell \in \mathscr{L}_{\text{neg}}$.

Procedure: We freeze θ_ℓ for $\ell \in \mathscr{L}_{\text{neg}}$, randomize (reinitialize) $\theta_{\ell'}$ for $\ell' \notin \mathscr{L}_{\text{neg}}$, and fine-tune. **Results:** Table 8 shows that NR-Freeze achieves performance on par with models trained from scratch (Retrain), confirming the theoretical prediction that negation preserves layer-wise optimality. The same stopping conditions as for Table 1 to compute the costs are used.

13.8. Spectral Content of Gradient Covariance

Given a model $\mathcal{N}^\theta : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ with $\theta \in \mathbb{R}^d$, define B the minibatch random variable with value in $\mathbb{R}^{d_{\text{in}} \times b}$ with b the bath size, $(\theta^t)_{t \in [0, t_{\max}]}$ is the parameter vector of the model across gradient descent and $\Sigma := \text{Cov}(\nabla_\theta \mathcal{L}_B(\mathcal{N}^{\theta^T}))$ the covariance matrix of the random variable whose value is the gradient of the model on a random batch B at a random time $T \in [0, t_{\max}]$. For our simulations, $t_{\max} = 0$. Consider $\text{Sp}(\Sigma) = \{\lambda_1, \cdots, \lambda_d\}$ the spectrum of the covariance matrix with $\lambda_1 > \lambda_2 > \cdots > \lambda_d$. We are interested in visualizing the spectral content curves:

$$\Psi : \alpha \mapsto \frac{\sum_{i=1}^{[\alpha d]} \lambda_i}{\sum_{i=1}^{d} \lambda_i}, \quad \alpha \in [0, 1]. \tag{89}$$

This spectral content measures the fraction of eigenvectors generating the linear space generated by the gradients (up to some noise threshold). More precisely, for $\beta = 95\%$, define $\alpha_\beta := \min\{\alpha \mid \Psi(\alpha) \geq \beta\}$. The quantity $[\alpha_\beta d]$ measures the minimal number of dimension for a linear subspace to contain 95% of the Euclidean norm squared of the noise of the gradient across training.

The most direct approach to evaluation the function Ψ is based on a direct use of the spectrum of the empirical covariance matrix [2]. First, we evaluate the gradient $(X_i)_{i=1..b} := (\nabla_\theta \mathcal{L}_{B_i}(\mathcal{N}^{\theta^0}))_{i=1..b}$ for $b \in \mathbb{N}$ randomly sampled mini-batchs $(B_i)_{i=1}^b$; and where θ^τ is the parameter vector of the model at communication round τ. We sample a random subsets $K_1, \cdots, K_p \subset \{1, \cdots, d\}$ of model parameters and project each X_i onto $\mathbb{R}^{K_j}$ to get $X_i^{K_j}$. Then, we compute the spectrum $S_j := \text{Sp}(\text{Cov}(X_i^{K_j}, X_i^{K_j})) = \{\lambda_1, \cdots, \lambda_k\}$ of the empirical covariance matrix. Finally, we compute our estimators $\widehat{\Psi}_j$ with Equation (89) and $\widehat{\Psi} := \frac{1}{p} \sum_{j=1}^p \widehat{\Psi}_j$.

Beware that, this direct estimator of the spectrum of Σ is inconsistent [17, 30, 67] as we are in the setting of very high dimensional random vectors (long-story short, covariance matrices converge toward Gaussian ensembles, the spectrum thus converges toward Tracy-Widom distribution

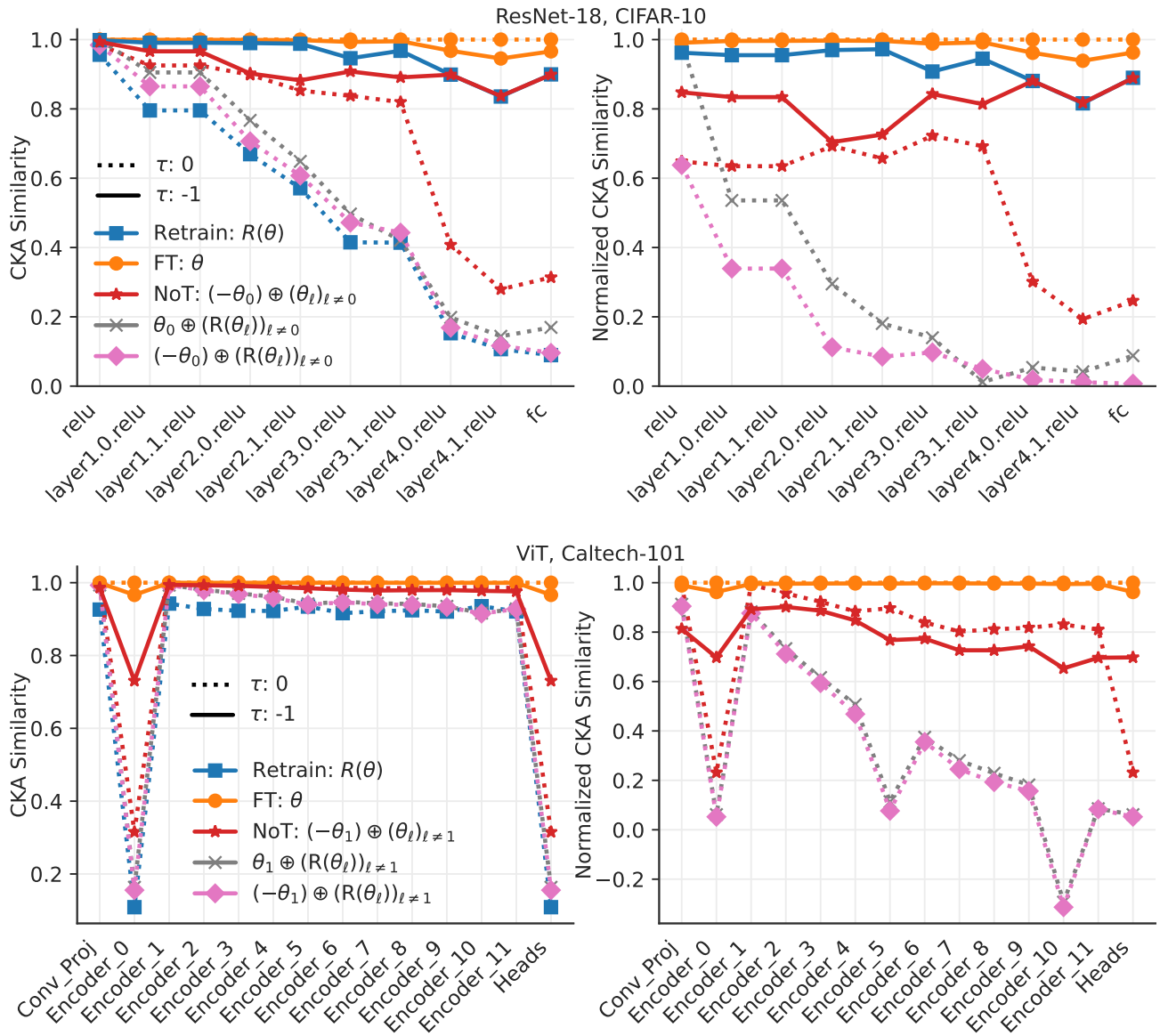

Figure 7. CKA (left) and normalized CKA (right) for **ResNet-18** (top) and **ViT** (bottom) layer activations, compared to the original model ($\theta = \theta^*$) before fine-tuning (FT@τ:0). τ:0 and -1 denote the first and last communication rounds. Models with negated first-layer weights (ℓ:0 for ResNet-18 and ℓ:1 for ViT) and randomized (reinitialized) remaining layers are denoted as $(-\theta_0) \oplus (\mathrm{R}(\theta_\ell))_{\ell \neq 0}$, where $R(\cdot)$ refers to reintitializing. We normalize the CKA via $\frac{\mathrm{CKA}_Q - \mathrm{CKA}_{Retrain@\tau:0}}{1 - \mathrm{CKA}_{Retrain@\tau:0}}$ to clarify these results, where Q is any model.

[42]). As a result, although still barely meaningful for shallow CNN, it yields inconclusive results for larger models (ResNet and ViT). We thus implement the L^∞ version of the algorithm described in [31] to extract meaningful information from the empirical spectrum of Σ obtained by the method above.

14. Further Implementation Details

This section details the hyperparameters used for the experiments in this paper.

14.1. Federated Unlearning

The global model was trained until convergence using the local data of all 10 clients using SGD with 0.9 momentum, 0.001 learning rate, and 5e-4 for weight decay. In every communication round, all clients participate in the federation. For all the experiments, except for CNN architectures, we upscaled the images to 256 and utilized random cropping to 224 and random horizontal flip with 0.5 probability. For CNN experiments, we used the original image sizes of 32 with random horizontal flip with 0.5 probability. Local

Table 8. **Layer-Wise Optimality (LWO)** test of negated models via reinitialization of non-negated layers followed by fine-tuning. Retrain denotes a model trained from scratch, and NR-freeze denotes a model obtained via **Negating & Freezing $\mathscr{L}_{\text{neg}}$, Reinitializing $\mathscr{L} \setminus \mathscr{L}_{\text{neg}}$, and fine-tuning**. Results confirm that negation preserves layer-wise optimality, confirming our theoretical prediction that negation is Layer-Wise Optimality Preserving.

Dataset & Model	Method	Accuracy (%)			Privacy (%)	Avg. Gap ↓	Cost (Bytes & FLOPs)	
		Retain (Δ ↓)	Forget (Δ ↓)	Test (Δ ↓)	MIA (Δ ↓)		Comm. ↓	Comp. ↓
CIFAR-10 CNN	Retrain	$91.66_{\pm 0.12}$ (0.00)	$83.05_{\pm 0.23}$ (0.00)	$82.32_{\pm 0.30}$ (0.00)	$50.23_{\pm 0.39}$ (0.00)	0.00	$1.35e^{10}$	$5.81e^{16}$
	NR - Freeze	$91.71_{\pm 0.11}$ (0.05)	$82.93_{\pm 0.17}$ (0.12)	$82.32_{\pm 0.28}$ (0.00)	$50.23_{\pm 0.12}$ (0.00)	0.04	$1.04e^{10}$	$4.51e^{16}$
CIFAR-100 CNN	Retrain	$72.32_{\pm 0.11}$ (0.00)	$53.31_{\pm 0.87}$ (0.00)	$54.28_{\pm 0.25}$ (0.00)	$49.70_{\pm 0.64}$ (0.00)	0.00	$1.38e^{10}$	$5.96e^{16}$
	NR-Freeze	$72.46_{\pm 0.51}$ (0.14)	$53.34_{\pm 0.79}$ (0.03)	$54.53_{\pm 0.48}$ (0.25)	$49.77_{\pm 0.53}$ (0.07)	0.12	$1.23e^{10}$	$5.30e^{16}$
CIFAR-10 ResNet-18	Retrain	$100.00_{\pm 0.00}$ (0.00)	$87.66_{\pm 0.64}$ (0.00)	$87.73_{\pm 0.35}$ (0.00)	$49.37_{\pm 0.29}$ (0.00)	0.00	$1.23e^{12}$	$5.66e^{18}$
	NR - Freeze	$99.98_{\pm 0.00}$ (0.02)	$87.47_{\pm 0.05}$ (0.19)	$86.72_{\pm 0.02}$ (1.01)	$49.70_{\pm 0.70}$ (0.33)	0.39	$5.98e^{11}$	$2.75e^{18}$
CIFAR-100 ResNet-18	Retrain	$99.96_{\pm 0.00}$ (0.00)	$59.96_{\pm 0.61}$ (0.00)	$60.66_{\pm 0.63}$ (0.00)	$50.30_{\pm 0.30}$ (0.00)	0.00	$7.34e^{11}$	$3.38e^{18}$
	NR - Freeze	$99.98_{\pm 0.13}$ (0.01)	$60.80_{\pm 0.24}$ (0.82)	$60.30_{\pm 0.31}$ (1.67)	$50.00_{\pm 0.22}$ (0.90)	0.85	$7.43e^{11}$	$3.42e^{18}$
Caltech-101 ViT	Retrain	$99.73_{\pm 0.04}$ (0.00)	$48.29_{\pm 0.44}$ (0.00)	$48.02_{\pm 0.72}$ (0.00)	$49.67_{\pm 3.47}$ (0.00)	0.00	$1.76e^{12}$	$1.37e^{21}$
	NR - Freeze	$99.71_{\pm 0.08}$ (0.02)	$48.41_{\pm 1.03}$ (0.12)	$48.10_{\pm 0.37}$ (0.08)	$49.73_{\pm 1.59}$ (0.06)	0.07	$1.55e^{12}$	$1.21e^{21}$

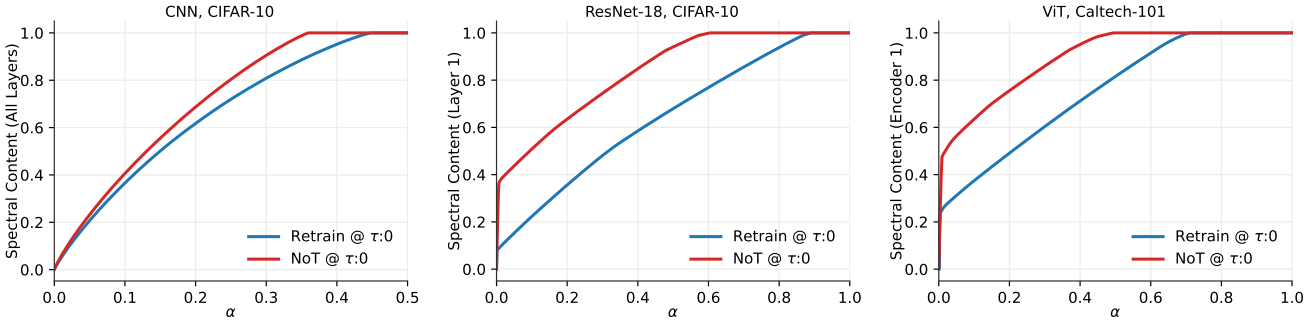

Figure 8. Spectral content of CNN (left), ResNet-18 (mid) and ViT (right) models. CNN: batch size 64, $b = 2048$, $|K| = 2000$ and $p = 100$ with K with indices in all layers. ResNet: batch size 64, $b = 128$, $|K| = 2000$ and $p = 100$ with K with indices in only the first residual block. ViT: batch size 16, $b = 128$, $|K| = 2000$ and $p = 100$ with K with indices in only the first encoder. In all cases, we observe that the spectral content reaches 100% for NoT with between 20% and 30% less dimensions: $\Psi^{-1}_{\text{NoT}}(1^-)/\Psi^{-1}_{\text{Retrain}}(1^-) \in [0.7, 0.8]$. For ViT and ResNet, we focus on the first layer after the convolution projection on which dimensionality reduction is the strongest and because the size of these models makes it increasingly difficult to compute an accurate spectrum estimator.

client training is set to 1. The global model was trained for communication rounds $\mathcal{T}$ = 2000 (CNN + CIFAR-10), 2000 (CNN + CIFAR-100), 800 (ResNet-18 + CIFAR-10), 1000 (ResNet-18 + CIFAR-100), 300 (ViT + Caltech-101), and 500 (ViT + CIFAR-100).

Below are the parameters we used for each unlearning algorithm, where we select client 0 to be the target client and assume unlearning starts at round $\mathcal{T}$ and lasts for another $\mathcal{T}$ number of rounds. We utilized the same parameters for training of the global model unless stated otherwise below. We fine-tune all the hyperparameters to get the lowest average gap for the same number of communication rounds as the training of the global model.

- **Retrain.** We use the same global values for training and save the model after $\mathcal{T}$ rounds.
- **FT.** We use the same global values for training.
- **FedEraser.** For *CNN*: We use the same global values for training.
- **FUKD.** For *CNN*: learning rate for unlearning is set to 1e-4, momentum to 0.9, distillation epochs to 3, and temperature to 3.
- **PGD.** For *CNN*: learning rate for unlearning is set to 0.1, unlearning iterations to 10, momentum to 0.9, updates per iteration to 5, distance threshold to 3, and clip gradient to 2. For *ResNet-18*: learning rate for unlearning is set to 0.001, unlearning iterations to 2, momentum to 0.9, updates per iteration to 7, distance threshold to 2.2, and clip gradient to 1. *ViT*: learning rate for unlearning is set to 0.001, unlearning iterations to 10, momentum to 0.9, updates per iteration to 5, distance threshold to 1, and clip gradient to 4.
- **MoDE.** For *CNN*: memory guidance rounds is set to 3 with 2 MoDE rounds, MoDe coefficient to 0.4, learning rate for the models is 0.005, and the learning rate for the degradation model is 0.1. For *ResNet-18*: memory guidance rounds is set to 9 with 8 MoDE rounds, MoDe coefficient to 0.95, learning rate for the models is 0.05, and the learning rate for the degradation model is 0.1. *ViT*: mem-

ory guidance rounds is set to 15 with 12 MoDE rounds, MoDe coefficient to 0.9, and learning rate for all models is 0.001.

- **FCU.** An Adam optimizer is used where the momentum terms are set to 0.9 and 0.99, and a ReduceLROnPlateau scheduler with learning rate 0.1 as the starting point while reducing it 1e-7 with a factor of 0.1 and patience of 2. For *CNN*: learning rate for unlearning is set to 0.01, unlearning iterations to 20, fusion interval to 10, and low-frequency to 0.9. For *ResNet-18*: learning rate for unlearning is set to 0.01, unlearning iterations to 10, fusion interval to 10, and low-frequency to 0.7. *ViT*: learning rate for unlearning is set to 0.01, unlearning iterations to 2000, fusion interval to 9, and low-frequency to 0.7.
- **NoT.** For all architectures we negate the first layer, except efficient net we negate the weights of all convolution and dense layers with indices ≤ 150. For *ViT*, we negate the convolution projection (conv_proj) layer.

14.2. Centralized Unlearning

In the following, we provide the hyperparameters we used for our baselines in Table 2. For the base model, we used ResNet-18 with CIFAR-10 dataset where we used the official train set to train the base model using SGD with 0.9 momentum, 0.01 learning rate, and 5e-4 for weight decay. We also used cosine annealing for 200 epochs with a minimum of 0.0001 and saved the best model with the highest accuracy on the test set. For all the experiments, we upscaled the CIFAR-10 images to 256 and utilized random cropping to 224 and random horizontal flip with 0.5 probability.

Below are the parameters we used for each unlearning algorithm, where we randomly select 10% of the data as forget data. We utilized the same parameters for training of the base model unless stated otherwise below. We fine-tune all the hyperparameters to get the lowest average gap for 50 epochs.

- **Retrain.** We use the same base values for training and save the model with highest test accuracy.
- **FT.** We use the same base values for training.
- **RandL.** We use the same base values for training. Further, at each round, we iterativly train the model on random labels for one epoch and then one epoch on retain data.
- **GA.** We use the same base values for training, and we do one epoch of GA and 49 epochs of fine-tuning.
- **BadT.** We use 1 as our temperature for BadT.
- ℓ_1**-sparse.** We do 2 epochs with ℓ_1 loss where we use 0.002 as our coefficient. Then, we do 48 epochs of fine-tuning.
- **SSD.** We search over the space of dampening constants and selection weights. We got the best results for 0.5 for both dempening constant and and selection weights.
- **SalUn.** We masked 20% of the model weights for SalUn.
- **NoT.** We negate only the weights of the first layer of ResNet-18 since it had the best performance. We tune the learning rate and select 0.1 as the starting point while reducing it 1e-5 in 10 epochs using cosine annealing.