Machine Unlearning under Overparameterization

Jacob L. Block [1] **Aryan Mokhtari** [1] **Sanjay Shakkottai** [1]

Abstract

We study the unlearning problem in the overparameterized regime, where many models interpolate the data. In this setting, defining the unlearning solution as any loss minimizer over the retained data—as in prior work in the underparameterized case—is inadequate, since the original model may already interpolate the retained data and satisfy this condition. Further, loss gradients vanish, rendering prior methods based on loss gradient perturbations ineffective, motivating new unlearning definitions and algorithms. We define the unlearning solution as the minimum-complexity interpolator of the retained data and propose a framework to recover this solution that minimizes a regularized objective under a relaxation of the interpolation constraint, enforcing the perturbation of the original model to be orthogonal to the model gradients on the retained data. For different model classes, we provide exact and approximate unlearning guarantees, and we show that an implementation of our framework outperforms existing baselines across unlearning experiments.

1. Introduction

As models are trained on vast datasets, the ability to remove the influence of specific samples from a trained model is essential. *Machine unlearning* algorithms (Cao & Yang, 2015) address this issue by modifying a model trained on a dataset $\mathcal{D}$ to forget a subset of samples, termed the forget set $\mathcal{D}_f$, yielding a model that behaves as if trained only on the remaining data, denoted the retain set $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$, without having to retrain the model from scratch on $\mathcal{D}_r$.

In the *underparameterized* regime, where the training loss has a unique minimizer, the unlearning solution is naturally defined as the unique loss minimizer over $\mathcal{D}_r$. When the loss is further strongly convex, approximate algorithms have been developed using influence functions, which estimate the effect of removing a sample via a gradient ascent step preconditioned by the inverse loss Hessian (Bae et al., 2022; Sekhari et al., 2021; Guo et al., 2020). In contrast, the *overparameterized* regime admits many interpolating solutions, making loss optimality alone an insufficient unlearning definition: the original model $\boldsymbol{\theta}^*$ may already minimize the loss on $\mathcal{D}_r$ while encoding information about $\mathcal{D}_f$. Moreover, interpolation causes gradients to vanish, rendering loss-gradient-based methods such as influence function ineffective (Theorem 2.2). This motivates new definitions and algorithms tailored to the overparameterized setting.

We define the unlearning solution in the overparameterized setting as the model which minimizes a complexity measure R (e.g. the parameter norm), subject to minimizing the loss over $\mathcal{D}_r$. This ensures the unlearned model reveals no information about the forgotten data and maintains strong generalization. We further propose a new algorithmic framework to compute this solution. We focus on settings where the loss is minimized by any interpolating model, so the loss minimization constraint reduces to requiring interpolation over $\mathcal{D}_r$. We then relax the interpolation constraint via a first-order Taylor expansion around $\boldsymbol{\theta}^*$, enforcing the perturbation $\boldsymbol{\Delta}$ of the initial model to be orthogonal to the model gradients at $\boldsymbol{\theta}^*$ on $\mathcal{D}_r$. This simplifies the problem and requires only gradient access on $\mathcal{D}_r$. To mitigate error from this relaxation, we add a regularizer $\hat{R}(\boldsymbol{\Delta})$ to control the size and direction of the drift. The final objective minimizes $R(\boldsymbol{\theta}^*+\boldsymbol{\Delta}) + \hat{R}(\boldsymbol{\Delta})$ under the relaxed orthogonal gradient constraint, yielding updated parameters $\boldsymbol{\theta}^* + \boldsymbol{\Delta}$.

Contributions. In theory, we show that for linear models and networks, the relaxed problem recovers the exact unlearning solution when R measures the ℓ_2-norm of either the effective linear predictor or the full parameter vector. For two-layer perceptrons with non-linear activation where R measures network width, our framework yields interpolating solutions that match best known upper bounds on network size. We then translate our framework into a practical algorithm MinNorm-OG that accesses a subset of $\mathcal{D}_r$, aligning with the data access assumptions in prior empirical work. Our method alternates between minimizing a regularized objective under the orthogonal gradient constraints and descending the retain set loss. We demonstrate strong empirical performance across unlearning experiments.

[1]Department of Electrical and Computer Engineering, The University of Texas at Austin. Correspondence to: Jacob L. Block <jblock@utexas.edu>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

2. Unlearning in Overparameterized Settings

We first introduce notation (additional standard definitions in Appendix A) for our unlearning setting, highlighting why loss optimality alone no longer suffices to define the ground truth unlearning solution and demonstrating why loss-gradient-based methods, originally designed for the underparameterized case, prove ineffective.

We define the training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, with inputs $\boldsymbol{x}_i \in \mathbb{R}^m$ and outputs $\boldsymbol{y}_i \in \mathbb{R}^l$ from data domain $\mathcal{Z} = \mathbb{R}^m \times \mathbb{R}^l$. Initial training is performed on $\mathcal{D}$ over the models $f(\boldsymbol{\theta}, \cdot)$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$, where f maps $(\boldsymbol{\theta}, \boldsymbol{x})$ to the prediction $f(\boldsymbol{\theta}, \boldsymbol{x}) \in \mathbb{R}^l$. We define the training procedure $\mathcal{A} : 2^{\mathcal{Z}} \to \mathbb{R}^d$, which takes in a dataset and returns the parameters $\boldsymbol{\theta}^*$ corresponding to the trained model. We assume $\mathcal{A}$ is faithful to a known loss function $\mathcal{J}$, meaning $\mathcal{A}(\mathcal{D}) = \boldsymbol{\theta}^*$ is guaranteed to minimize $\mathcal{J}$ over $\mathcal{D}$:

$$\mathcal{A}(\mathcal{D}) = \boldsymbol{\theta}^* \in \operatorname*{argmin}_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}; \mathcal{D}),$$

where $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{n} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D}} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y})$ is the average of the sample-wise loss $\mathcal{L}$ on $\mathcal{D}$. For our discussion, we consider losses $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y})$ which are minimized when $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{y}$, meaning that interpolation implies loss minimization (e.g. ℓ_p-norm regression).

Given a request for the model to forget a subset of the training data $\mathcal{D}_f \subseteq \mathcal{D}$, we apply an unlearning algorithm $M(\mathcal{A}, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f)$ which is given the learning algorithm $\mathcal{A}$, side information $\mathcal{I}_r$ (e.g., a subset of $\mathcal{D}_r$, or loss Hessian on $\mathcal{D}_r$), initial solution $\mathcal{A}(\mathcal{D})$, and forget set $\mathcal{D}_f$, and attempts to recover the unlearning solution, denoted by $\boldsymbol{\theta}_r^*$.

2.1. Defining Unlearning Beyond Loss Optimality

In the overparameterized setting, defining the unlearning solution as any minimizer of $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}_r)$–as done in prior work focused on the underparameterized case–is inadequate: the original model $\boldsymbol{\theta}^*$, which interpolates all of $\mathcal{D}$, remains a valid minimizer on $\mathcal{D}_r$ yet encodes information about $\mathcal{D}_f$ (see Appendix C for a concrete illustration). To address this, we define the unlearning solution as the loss minimizer which minimizes an additional objective function $R(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}_r^* \in \operatorname*{argmin}_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) \quad \text{s.t. } \boldsymbol{\theta} \in \operatorname*{argmin}_{\boldsymbol{\theta}'} \mathcal{J}(\boldsymbol{\theta}'; \mathcal{D}_r). \quad (1)$$

This bilevel optimization problem searches for the model which minimizes the complexity measure R among all models which minimize the retain set loss. When R admits a unique solution, this formulation overcomes the prior issues of non-uniqueness and the risk of revealing information from the forget set. For our theoretical results, we focus on R as a regularization function that penalizes model complexity. This way, the solution $\boldsymbol{\theta}_r^*$ to (1) corresponds to the simplest interpolating model over $\mathcal{D}_r$, a desirable property in overparameterized settings (Hastie et al., 2022).

2.2. Loss Gradient Methods Deployed Under Overparameterization

For the unlearning solution defined in (1), existing methods based on loss gradient perturbations fail to achieve meaningful unlearning updates. Prior theoretical works proposed gradient-ascent style updates (Bae et al., 2022; Sekhari et al., 2021; Guo et al., 2020), while existing empirical methods perform combinations of loss ascent over $\mathcal{D}_f$, loss descent over $\mathcal{D}_r$, and parameter noising (Neel et al., 2021; Graves et al., 2021; Chourasia & Shah, 2023; Kurmanji et al., 2023). We characterize these methods as *loss-gradient unlearning*.

Definition 2.1. Let $\boldsymbol{\theta}^* = \mathcal{A}(\mathcal{D})$. We say M performs loss-gradient unlearning if for any positive semi-definite $\boldsymbol{P}_r, \boldsymbol{P}_f \in \mathbb{R}^{d \times d}$ and zero-mean random variable $\boldsymbol{\xi} \in \mathbb{R}^d$,

$$M(\mathcal{A}, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f) = \boldsymbol{\theta}^* - \boldsymbol{P}_r \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_r) + \boldsymbol{P}_f \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_f) + \boldsymbol{\xi} \quad (2)$$

In theory loss-gradient unlearning is principled in the underparameterized case, but it fails under overparameterization.

Theorem 2.2. *Let $f(\boldsymbol{\theta}^*, \cdot)$ interpolate $\mathcal{D}$, so $f(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{y}$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$, and let M_{LG} be any loss-gradient unlearning method. If the sample loss $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ is minimized when $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{y}$, then for all $\mathcal{D}_f \subseteq \mathcal{D}$,*

$$M_{LG}(\mathcal{A}, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f) = \boldsymbol{\theta}^* + \boldsymbol{\xi}.$$

Since $\boldsymbol{\theta}^*$ already minimizes $\mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_r)$, the loss gradients vanish and M_{LG} only adds noise and fails to achieve the desired unlearn solution.

3. Our Proposed Framework

We present a new framework to efficiently unlearn under overparameterization. We consider losses where data interpolation implies minimization, so we replace loss minimization in (1) with an interpolation constraint, yielding:

$$\boldsymbol{\theta}_r^* \in \operatorname*{argmin}_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) \text{ s.t. } f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{y} \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_r \quad (3)$$

We aim to efficiently recover $\boldsymbol{\theta}_r^*$ using the feasibility of the initial model $\boldsymbol{\theta}^*$ for the above problem. While the constraint in (3) is generally hard to enforce, we use the first-order approximation $f(\boldsymbol{\theta}, \boldsymbol{x}) \approx f(\boldsymbol{\theta}^*, \boldsymbol{x}) + \nabla f(\boldsymbol{\theta}^*, \boldsymbol{x})^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*)$. Since $\boldsymbol{\theta}^*$ interpolates $\mathcal{D}_r$, this reduces the constraint in (3) to $\nabla f(\boldsymbol{\theta}^*, \boldsymbol{x})^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*) = 0$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_r$, yielding the following problem with linear constraints.

$$\min_{\boldsymbol{\Delta}} R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) \text{ s.t. } \nabla f(\boldsymbol{\theta}^*, \boldsymbol{x})^\top \boldsymbol{\Delta} = 0 \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_r \quad (4)$$

where we define the drift variable $\boldsymbol{\Delta} = \boldsymbol{\theta} - \boldsymbol{\theta}^*$. This relaxation is reasonable, but accurate for general functions only when the drift $\boldsymbol{\Delta}$ remains small in some norm. To keep

the solution close to that of the original problem (3), we add a regularization term $\hat{R}(\boldsymbol{\Delta})$ to control the drift. The resulting objective function is $\tilde{R}(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) := R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) + \hat{R}(\boldsymbol{\Delta})$. We then propose to solve the following problem, which notably only requires model gradients over $\mathcal{D}_r$.

$$\tilde{\boldsymbol{\Delta}} \in \underset{\boldsymbol{\Delta}}{\operatorname{argmin}} \quad \tilde{R}(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) \qquad (5)$$
$$\text{s.t.} \quad \nabla f(\boldsymbol{\theta}^*, \boldsymbol{x})^\top \boldsymbol{\Delta} = 0 \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_r$$

The suggested unlearned model is then $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$. Although (5) uses relaxed constraints, we will show that for various model classes there exists an $\hat{R}$ such that the solution to (5) either solves the original unlearning problem (3), or yields a model that both interpolates $\mathcal{D}_r$, remaining feasible for (3), and satisfies an upper bound on the complexity measure R.

4. Theoretical Guarantees

This section provides theoretical guarantees for using our proposed relaxation (5) to solve the exact unlearning problem (3). Going forward, we denote Euclidean projection onto the set $\boldsymbol{S}$ as $\mathcal{P}_{\boldsymbol{S}}(\cdot)$ and define the penalty function $\delta_{\{a\}}$ which is $+\infty$ if condition a is satisfied and 0 otherwise.

4.1. Linear Model

Given a linear model $\boldsymbol{\theta}^*$ with $\boldsymbol{\theta}^{*\top}\boldsymbol{x} = y$ for all $(\boldsymbol{x}, y) \in \mathcal{D}$, we achieve exact unlearning for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$ in (3).

Theorem 4.1. *Let $\tilde{\boldsymbol{\Delta}}$ solve (5) for $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$ and $\tilde{R}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$. Then the recovered solution $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ solves the exact unlearning problem (3) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$*

In the linear case, the relaxed constraints of (5) introduce no approximation error, so $\hat{R}(\cdot) = 0$ suffices.

4.2. L-Layer Linear Network

We extend our analysis to the model class of L-layer linear networks. Partition the parameters $\boldsymbol{\theta} = [\boldsymbol{c}; \operatorname{vec}(\boldsymbol{A}_1); \ldots; \operatorname{vec}(\boldsymbol{A}_{L-1})]$, with $\boldsymbol{A}_\ell \in \mathbb{R}^{h_\ell \times h_{\ell-1}}$ and $\boldsymbol{c} \in \mathbb{R}^{h_{L-1}}$ for $\ell = 1, \ldots, L-1$, and define $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{c}^\top \boldsymbol{A}_{L-1} \cdots \boldsymbol{A}_1 \boldsymbol{x}$. The input dimension is $m = h_0$, and we work within the overparameterized regime, meaning $n < m$. For brevity, define the effective predictor $\mathbf{w}(\boldsymbol{\theta}) = \boldsymbol{A}_1^\top \cdots \boldsymbol{A}_{L-1}^\top \boldsymbol{c}$, so that $f(\boldsymbol{\theta}, \boldsymbol{x}) = \mathbf{w}(\boldsymbol{\theta})^\top \boldsymbol{x}$. We study two natural choices for R in (3): (i) the norm of the effective predictor, and (ii) the norm of all model parameters.

4.2.1. Minimizing Predictor Norm

We first consider $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$. Given initial model $\boldsymbol{\theta}^* = \left[\boldsymbol{c}^*; \operatorname{vec}(\boldsymbol{A}_1^*); \ldots; \operatorname{vec}(\boldsymbol{A}_{L-1}^*)\right]$ such that $\mathbf{w}(\boldsymbol{\theta}^*)^\top \boldsymbol{x} = y$ for all $(\boldsymbol{x}, y) \in \mathcal{D}$, we aim to solve (3) for this choice of R. In this case f is non-linear with respect to $\boldsymbol{\theta}$, so the first-order approximation for the constraints is not tight. However, we show that adding a suitable regularizer $\hat{R}$ to control model drift ensures that solving the relaxed problem gives a solution to the exact problem.

Theorem 4.2. *The solution to the relaxed unlearning problem (5) with the following choice of $\tilde{R}$ solves the exact unlearning problem (3) for $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$.*

$$\tilde{R}(\boldsymbol{\theta}; \boldsymbol{\theta}^*) = \|\mathbf{w}(\boldsymbol{\theta})\|_2 + \delta_{\{\boldsymbol{c} \neq \boldsymbol{c}^*\}} + \sum_{\ell=2}^{L-1} \delta_{\{\boldsymbol{A}_\ell \neq \boldsymbol{A}_\ell^*\}} \qquad (6)$$

Thus, if $\tilde{\boldsymbol{\Delta}}$ solves the relaxed problem (5) with the above choice of $\tilde{R}$, $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ solves the exact problem (3).

4.2.2. Minimizing Parameter Norm

We next analyze when $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$. Here, we can construct an exact unlearning solution from the unlearning solution in the previous case when $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$.

Theorem 4.3. *Let $\hat{\boldsymbol{\theta}}_r^*$ solve (3) for $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$, so $\mathbf{w}(\hat{\boldsymbol{\theta}}_r^*)$ is the min ℓ_2-norm linear predictor over $\mathcal{D}_r$. Define $\rho = \|\mathbf{w}(\hat{\boldsymbol{\theta}}_r^*)\|_2$ and let $\boldsymbol{v}_\ell \in \mathbb{R}^{h_\ell}$ for $\ell \in [L-1]$ each satisfy $\|\boldsymbol{v}_\ell\|_2 = 1$. Set*

$$\tilde{\boldsymbol{A}}_1 = \rho^{\frac{1-L}{L}} \boldsymbol{v}_1 \mathbf{w}(\hat{\boldsymbol{\theta}}_r^*)^\top, \quad \tilde{\boldsymbol{c}} = \rho^{\frac{1}{L}} \boldsymbol{v}_{L-1},$$
$$\tilde{\boldsymbol{A}}_\ell = \rho^{\frac{1}{L}} \boldsymbol{v}_\ell \boldsymbol{v}_{\ell-1}^\top \quad \text{for } \ell = 2, \ldots, L-1.$$

Then $\tilde{\boldsymbol{\theta}} = \left[\tilde{\boldsymbol{c}}; \operatorname{vec}(\tilde{\boldsymbol{A}}_1); \ldots; \operatorname{vec}(\tilde{\boldsymbol{A}}_{L-1})\right]$ solves the exact unlearning problem (3) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$.

Thus, we can apply the previous results to find a solution for (3) when R measures the predictor norm and then update the parameters as prescribed by Theorem 4.3.

4.3. 2-Layer Perceptron

We lastly consider a 2-layer perceptron with a non-linear activation. We define $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{c}^\top \phi(\boldsymbol{A}\boldsymbol{x})$, where we partition $\boldsymbol{\theta} = \left[\boldsymbol{c}; \operatorname{vec}(\boldsymbol{A})\right]$ for $\boldsymbol{c} \in \mathbb{R}^h$, $\boldsymbol{A} \in \mathbb{R}^{h \times m}$. $\phi : \mathbb{R} \to \mathbb{R}$ is an activation function, and we abuse notation and write $\phi(\boldsymbol{A}\boldsymbol{x})$ to denote the element-wise application of ϕ to $\boldsymbol{A}\boldsymbol{x}$. We analyze the case where R measures the number of active neurons of the network. Formally, we denote $\boldsymbol{a}_i^\top$ as the ith row of $\boldsymbol{A}$, and we set $R(\boldsymbol{\theta}) = \sum_{i=1}^h \mathbb{1}\{|\boldsymbol{c}_i| \, \|\boldsymbol{a}_i\|_2 > 0\}$. Given that $\boldsymbol{c}^{*\top} \phi(\boldsymbol{A}^* \boldsymbol{x}) = y$ for all $(\boldsymbol{x}, y) \in \mathcal{D}$, we chase the minimum neuron interpolating solution to $\mathcal{D}_r$:

$$\boldsymbol{\theta}_r^* \in \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \, R(\boldsymbol{\theta}) \;\; \text{s.t. } \boldsymbol{c}^\top \phi(\boldsymbol{A}\boldsymbol{x}) = y \; \forall (\boldsymbol{x}, y) \in \mathcal{D}_r \quad (7)$$

While we aim to solve (7) for any retain set $\mathcal{D}_r$, the exact minimal-width solution remains unknown. For $\mathcal{D}_r$ which has $n_r = |\mathcal{D}_r|$ samples, prior work shows that $n_r + 1$ neurons suffice for general activations (Rosset et al., 2007), while for ReLU activations, some n_r-sample datasets need at least $n_r - 2$ neurons (Yun et al., 2019). The following theorem shows that applying our framework and choosing $\hat{R}$ to restrict perturbations in $\boldsymbol{A}^*$ ensures that solving (5) yields a network feasible for (7) with at most n_r active neurons.

Theorem 4.4. *For $R(\boldsymbol{\theta})$ which measures the network width, define the surrogate objective*

$$\tilde{R}(\boldsymbol{\theta}\,;\boldsymbol{\theta}^*) = R(\boldsymbol{\theta}) + \delta_{\{\boldsymbol{A}\neq\boldsymbol{A}^*\}}. \tag{8}$$

Then the solution to the relaxed unlearning problem (5) with this choice of $\tilde{R}$ results in a network which interpolates $\mathcal{D}_r$, achieving feasibility for the exact unlearning problem (7), and admits at most $s = \dim\left(\operatorname{span}\{\phi(\boldsymbol{A}^*\boldsymbol{x})\}_{(\boldsymbol{x},y)\in\mathcal{D}_r}\right) \leq n_r$ active neurons, where $n_r = |\mathcal{D}_r|$.

For general activation functions, we recover a network that interpolates $\mathcal{D}_r$ with at most s active neurons, where s is the dimension of the span of the representations $\{\phi(\boldsymbol{A}^*\boldsymbol{x})\}_{(\boldsymbol{x},y)\in\mathcal{D}_r}$. Since s can never exceed $n_r = |\mathcal{D}_r|$, we guarantee a worst-case interpolation width of at most n_r, thereby improving the general bound of $n_r + 1$ implied by (Rosset et al., 2007) for minimum width interpolation.

5. From Theory to Practice

We translate our framework into a practical algorithm MinNorm-OG. We alternate between solving the relaxed problem (5) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$ and $\hat{R}(\boldsymbol{\Delta}) = \lambda_t \|\boldsymbol{\Delta}\|_2^2$, and descending the loss on $\mathcal{D}_r$ to remain feasible for the exact unlearning problem (3). We only enforce the orthogonality constraint in (5) over a subsample of size n_{pert} of each batch. We solve this version of (5) by projecting $\boldsymbol{\theta}$ onto the span of the model gradients over the subsample via a QR decomposition. This has complexity $O(dn_{\text{pert}}^2)$, equivalent to the $O(dn_B)$ cost of gradient descent when $n_{\text{pert}} < \sqrt{n_B}$, where $n_B = |\mathcal{B}|$ is the batch size. See Appendix E for pseudocode.

5.1. Experiments

We test our algorithm against the following methods. GD (Neel et al., 2021) runs gradient descent on $\mathcal{J}(\boldsymbol{\theta}\,;\mathcal{D}_r)$, while NGD (Chourasia & Shah, 2023) adds gradient noise to the GD steps. GA (Graves et al., 2021) runs gradient ascent on $\mathcal{J}(\boldsymbol{\theta}\,;\mathcal{D}_f)$. NegGrad+ (NGP) (Kurmanji et al., 2023) minimizes a weighted combination of the GD and GA objectives. SCRUB (Kurmanji et al., 2023) optimizes three objectives: minimizing $\mathcal{J}(\boldsymbol{\theta}\,;\mathcal{D}_r)$, minimizing KL divergence of model outputs on $\mathcal{D}_r$ relative to the original model, and maximizing KL divergence on $\mathcal{D}_f$. Negative Preference Optimization (NPO) (Zhang et al., 2024) runs a form of gradient ascent over $\mathcal{J}(\boldsymbol{\theta}\,;\mathcal{D}_f)$ inspired by preference optimization. We lastly include ridge regression, which approximates our unlearning objective (3) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$ by minimizing $\mathcal{J}(\boldsymbol{\theta};\mathcal{D}_r) + \lambda_t \|\boldsymbol{\theta}\|_2^2$ for $\lambda_t \geq 0$.

We use MNIST and CIFAR-10 (LeCun et al., 2010; Krizhevsky, 2009), generating red, green, and gray versions of each image. Models are trained to predict both content (digit or object) and color. The retain set $\mathcal{D}_r$ includes all content classes in gray only, while the forget set $\mathcal{D}_f$ includes all colors. The ground truth unlearned model performs well on gray inputs and always assigns probability 1 to gray over all input colors. Retain quality is measured by accuracy on gray test images; forget quality by mean squared error of the predicted gray probability to the ideal value of 1 on colored inputs. Figure 1 shows the Pareto frontier across methods, with each point representing the median of 5 trials for a hyperparameter setting. The shading represents half the interquartile range. The optimal point $(1, 0)$ indicates perfect retain accuracy and zero forget error, and GT denotes the ground truth unlearned model. MinNorm-OG performs best in both tasks, though all methods struggle to preserve accuracy on CIFAR-10, a harder task than to MNIST. Descent-based methods (GD, NGD, Ridge) remain near the initial model (upper right), which is already near-optimal on $\mathcal{D}_r$ and provides weak gradients for unlearning.

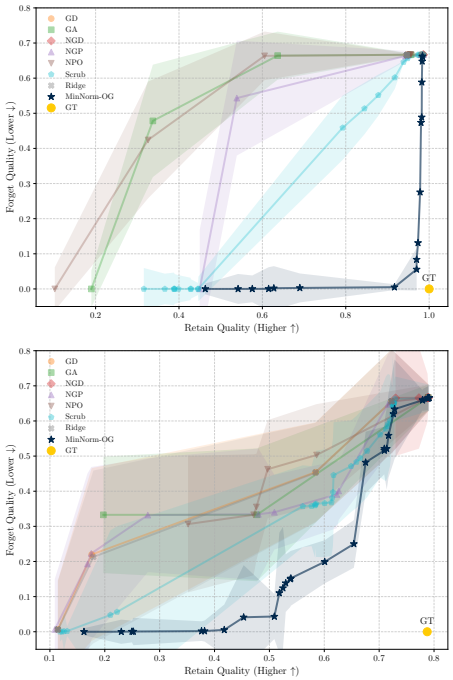


Figure 1. Pareto frontiers for each method across hyperparameter settings on MNIST (top) and CIFAR-10 (bottom). The x-axis shows accuracy on gray test images (higher is better), and the y-axis shows mean squared error between predicted probability of gray on all inputs and the target of 1 (lower is better). MinNorm-OG (ours) strictly dominates the other methods.

6. Conclusion

We proposed an unlearning framework for overparameterized models that selects the simplest model consistent with the retained data. We provided guarantees for recovering the minimum-complexity solution via a tractable relaxation, and showed that a practical implementation outperforms baselines. Future work includes extending the analysis to more complex models and experimenting at larger scales.

Acknowledgments

This work was supported in part by NSF Grants 2019844, 2107037, and 2112471, ONR Grant N00014-19-1-2566, the Machine Learning Lab (MLL) at UT Austin, the NSF AI Institute for Foundations of Machine Learning (IFML), and the Wireless Networking and Communications Group (WNCG) Industrial Affiliates Program. We are grateful for computing support on the Vista GPU Cluster through the Center for Generative AI (CGAI) and the Texas Advanced Computing Center (TACC) at the University of Texas at Austin.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bae, J., Ng, N., Lo, A., Ghassemi, M., and Grosse, R. B. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.

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

Chourasia, R. and Shah, N. Forget unlearning: Towards true data-deletion in machine learning. In *International conference on machine learning*, pp. 6028–6073. PMLR, 2023.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. Orthogonal gradient descent for continual learning. In *International conference on artificial intelligence and statistics*, pp. 3762–3773. PMLR, 2020.

Graves, L., Nagisetty, V., and Ganesh, V. Amnesiac machine learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11516–11524, 2021.

Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 3832–3842, 2020.

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of statistics*, 50(2):949, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. Technical Report.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. Towards unbounded machine unlearning. *Advances in neural information processing systems*, 36:1957–1987, 2023.

LeCun, Y., Cortes, C., and Burges, C. J. Mnist handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

Neel, S., Roth, A., and Sharifi-Malvajerdi, S. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pp. 931–962. PMLR, 2021.

Rosset, S., Swirszcz, G., Srebro, N., and Zhu, J. l1 regularization in infinite dimensional feature spaces. In *Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings 20*, pp. 544–558. Springer, 2007.

Sekhari, A., Acharya, J., Kamath, G., and Suresh, A. T. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

Yun, C., Sra, S., and Jadbabaie, A. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. *Advances in neural information processing systems*, 32, 2019.

Zhang, R., Lin, L., Bai, Y., and Mei, S. Negative preference optimization: From catastrophic collapse to effective unlearning. In *First Conference on Language Modeling*, 2024.

A. General Notation

Vectors and matrices are in bold, with vectors lowercase and matrices uppercase. For sets A, B, $A \sqcup B$ denotes disjoint union. 2^A is the power set. For a proposition a, $\mathbb{1}\{a\}$ is 1 if true and 0 otherwise; $\delta_{\{a\}}$ is $+\infty$ if true and 0 otherwise. For $\boldsymbol{x} \in \mathbb{R}^d$ and $A \subseteq \mathbb{R}^d$, $\mathcal{P}_A(\boldsymbol{x})$ is the Euclidean projection onto A. For $\boldsymbol{Z} \in \mathbb{R}^{m \times n}$, $\mathrm{vec}(\boldsymbol{Z}) \in \mathbb{R}^{mn}$ is the columnwise vectorization. $\mathrm{im}(\boldsymbol{Z})$, $\ker(\boldsymbol{Z})$, and $\mathrm{row}(\boldsymbol{Z})$ denote the image, kernel, and rowspace. $\|\boldsymbol{Z}\|_F$ is the Frobenius norm, $\|\boldsymbol{Z}\|_*$ is the nuclear norm, and $\|\boldsymbol{Z}\|_2$ is the spectral norm. For $Y \in \mathbb{R}^{m \times n}$, $\langle \boldsymbol{Z}, \boldsymbol{Y} \rangle$ is the Frobenius inner product and $\boldsymbol{Z} \odot \boldsymbol{Y}$ is the element-wise product. $\mathrm{tr}\{\cdot\}$ is the trace. For $\boldsymbol{x} \in \mathbb{R}^{d_x}$ and $\boldsymbol{y} \in \mathbb{R}^{d_y}$, $[\boldsymbol{x}; \boldsymbol{y}] \in \mathbb{R}^{d_x + d_y}$ stacks $\boldsymbol{x}$ and $\boldsymbol{y}$. $\|\boldsymbol{x}\|_p$ is the ℓ_p norm. $[n] = \{1, \ldots, n\}$. For $x \in \mathbb{R}$, $(x)_+ = \max\{x, 0\}$ is the ReLU. Let $\mathbf{0}$ and $\mathbf{1}$ denote the vectors with each entry equal to 0 and 1 respectively. Further, for $\boldsymbol{x} \in \mathbb{R}^d$ and $c \in \mathbb{R}$, let $\mathbf{1}_{\boldsymbol{x} \neq c}$ denote the vector which is 1 in each entry of $\boldsymbol{x}$ which is not equal to c and 0 otherwise.

B. Minimum Norm Solutions to Linear Regression

Here we prove various properties of minimum norm solutions to linear regression problems which we later use for our unlearning results. Following the notation in Section 2, we consider the full n-sample dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with sample inputs $\boldsymbol{x}_i \in \mathbb{R}^m$ and outputs $y_i \in \mathbb{R}$. We consider training a linear model $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^m$. We work within the overparameterized setting, so we assume $m > n$. Define the span of the input vectors $\boldsymbol{S} = \mathrm{span}\{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}\}$, and assume $\dim(\boldsymbol{S}) = n$ so the regression problem is realizable. Consider solving the following problem for finding the linear regression solution with minimum ℓ_2 norm:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\mathrm{argmin}} \|\boldsymbol{\theta}\|_2 \text{ s.t. } f(\boldsymbol{\theta}, \boldsymbol{x}) = y \quad \forall (\boldsymbol{x}, y) \in \mathcal{D}$$

Let $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ be the wide matrix whose ith row is equal to $\boldsymbol{x}_i^\top$, and let $\boldsymbol{y} \in \mathbb{R}^n$ be the vector whose ith element is y_i. Then, we can write an equivalent problem in matrix form.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\mathrm{argmin}} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 \text{ s.t. } \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} \tag{9}$$

We can then characterize the solution to the above problem relative to the constraint set.

Lemma B.1. *$\boldsymbol{\theta}^*$ is the unique vector in $\mathrm{row}(\boldsymbol{X})$ which is feasible for (9)*

Proof. The objective (9) is a convex objective with linear constraints which is bounded from below by 0 and has a non-empty feasible set. Thus, the KKT conditions are necessary and sufficient for optimality. We now derive the solution $\boldsymbol{\lambda}^* \in \mathbb{R}^n$ to the dual problem.

$$\begin{aligned}
\min_{\boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} \quad &= \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\lambda} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 + \boldsymbol{\lambda}^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\
&= \max_{\boldsymbol{\lambda}} \min_{\boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 + \boldsymbol{\lambda}^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}) \\
&= \max_{\boldsymbol{\lambda}} \frac{1}{2} \left\|\boldsymbol{X}^\top \boldsymbol{\lambda}\right\|_2^2 + \boldsymbol{\lambda}^\top \left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\lambda}\right) \quad \text{s.t.} \quad \boldsymbol{\theta} = \boldsymbol{X}^\top \boldsymbol{\lambda} \\
&= \max_{\boldsymbol{\lambda}} -\frac{1}{2} \left\|\boldsymbol{X}^\top \boldsymbol{\lambda}\right\|_2^2 + \boldsymbol{\lambda}^\top \boldsymbol{y} \quad \text{s.t.} \quad \boldsymbol{\theta} = \boldsymbol{X}^\top \boldsymbol{\lambda} \\
&\implies \boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{\lambda}^* = \boldsymbol{y} \text{ and } \boldsymbol{\theta}^* = \boldsymbol{X}^\top \boldsymbol{\lambda}^*
\end{aligned} \tag{10}$$

Thus the primal solution $\boldsymbol{\theta}^*$ must be of the form $\boldsymbol{X}^\top \boldsymbol{\lambda}^* \in \mathrm{row}(\boldsymbol{X})$. To show uniqueness, consider $\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^* \in \mathrm{row}(\boldsymbol{X})$ that are both feasible for (9). Then,

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta}_1^* = \boldsymbol{X}\boldsymbol{\theta}_2^* \implies \boldsymbol{X}(\boldsymbol{\theta}_1^* - \boldsymbol{\theta}_2^*) = \mathbf{0} \implies \boldsymbol{\theta}_1^* - \boldsymbol{\theta}_2^* \in \ker(\boldsymbol{X}).$$

But, since $\mathrm{row}(\boldsymbol{X})$ is a subspace, $\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^* \in \mathrm{row}(\boldsymbol{X})$ implies $\boldsymbol{\theta}_1^* - \boldsymbol{\theta}_2^* \in \mathrm{row}(\boldsymbol{X})$. Further, $\mathrm{row}(\boldsymbol{X}) = \ker(\boldsymbol{X})^\perp$. Thus,

$$\boldsymbol{\theta}_1^* - \boldsymbol{\theta}_2^* \in \ker(\boldsymbol{X}) \cap \ker(\boldsymbol{X})^\perp = \{\mathbf{0}\} \implies \boldsymbol{\theta}_1^* = \boldsymbol{\theta}_2^*$$

□

Using the same analysis, we can characterize the entire feasible set in terms of $\boldsymbol{\theta}^*$.

Lemma B.2. *The feasible set to* (9) $\{\boldsymbol{\theta} \mid \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta}\} = \boldsymbol{\theta}^* + \ker(\boldsymbol{X})$.

Proof. Let $\boldsymbol{\theta}'$ satisfy $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta}'$. Then, $\boldsymbol{X}(\boldsymbol{\theta}' - \boldsymbol{\theta}^*) = \boldsymbol{0}$ so $\boldsymbol{\theta}' - \boldsymbol{\theta} \in \ker(\boldsymbol{X})$.

To show the converse, take any $\boldsymbol{z} \in \ker(\boldsymbol{X})$. Then $\boldsymbol{X}(\boldsymbol{\theta}^* + \boldsymbol{z}) = \boldsymbol{X}\boldsymbol{\theta}^* + \boldsymbol{X}\boldsymbol{z} = \boldsymbol{X}\boldsymbol{\theta}^* = \boldsymbol{y}$. □

Using this characterization of $\boldsymbol{\theta}^*$ and the feasible set, we can cleanly understand how to achieve minimum norm solutions over just a subset of the constraints given a feasible point. This is central to our unlearning setup in later sections.

Lemma B.3. *Consider any subset $\mathcal{D}_r \subseteq \mathcal{D}$, and define $\boldsymbol{\theta}_r^*$ as the linear regression solution over just $\mathcal{D}_r$ with minimum norm:*

$$\boldsymbol{\theta}_r^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2 \text{ s.t. } f(\boldsymbol{\theta}, \boldsymbol{x}) = y \quad \forall (\boldsymbol{x}, y) \in \mathcal{D}_r \tag{11}$$

Let $\boldsymbol{S}_r = \operatorname{span}\{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}_r\}$. Then $\boldsymbol{\theta}_r^* = \mathcal{P}_{\boldsymbol{S}_r}(\boldsymbol{\theta}^*)$.

Proof. $\boldsymbol{\theta}^*$ already satisfies the feasibility constraint over the whole dataset $\mathcal{D}$, so it must be be feasible for (11). Applying Lemmas B.1 and B.2 to the minimum norm problem over just $\mathcal{D}_r$ (11), we must have that $\boldsymbol{\theta}_r^* \in \boldsymbol{S}_r$ and $\boldsymbol{\theta}^* = \boldsymbol{\theta}_r^* + \boldsymbol{z}$ for some $\boldsymbol{z} \in \boldsymbol{S}_r^\perp$. Then,

$$\mathcal{P}_{\boldsymbol{S}_r}(\boldsymbol{\theta}^*) = \mathcal{P}_{\boldsymbol{S}_r}(\boldsymbol{\theta}_r^* + \boldsymbol{z}) = \mathcal{P}_{\boldsymbol{S}_r}(\boldsymbol{\theta}_r^*) = \boldsymbol{\theta}_r^*.$$

□

C. Loss Minimization Does not Protect Against Data Leakage

The following example concretely demonstrates how certain minimizers of the retain set loss do not align with the intended goals of unlearning.

Recall the unlearning problem for linear regression discussed in Section 4.1. In this case, we use the linear model $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^m$. Further suppose the original dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ has n samples with $\boldsymbol{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R}$. Denote the subspace $\boldsymbol{S} = \operatorname{span}\{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}\}$, and assume $\dim(\boldsymbol{S}) = n$ so the problem is realizable. We work in the overparameterized setting where $m > n$ and the objective function is defined as the mean squared error denoted by

$$\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{n} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}} \left(y_i - \boldsymbol{\theta}^\top \boldsymbol{x}_i\right)^2$$

Consider when the learning algorithm $\mathcal{A}$ runs gradient descent on the loss, initialized at $\boldsymbol{0}$. Due to the overparameterization, $\mathcal{J}$ has an infinite number of minimizers which each achieve 0 loss. However, $\mathcal{A}$ is biased towards a specific minimizer which is the unique minimizer to the loss on the span of the input samples, denoted as the subspace $\boldsymbol{S}$.

Proposition C.1. *Let $\mathcal{A}^k(\mathcal{D})$ be a learning algorithm which runs k steps of gradient descent on $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D})$ initialized at $\boldsymbol{0}$, and define $\boldsymbol{S} = \operatorname{span}\{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}\}$. If $\lim_{k \to \infty} \mathcal{A}^k(\mathcal{D})$ converges to some $\boldsymbol{\theta}^*$, then*

$$\{\boldsymbol{\theta}^*\} = \boldsymbol{S} \cap \operatorname{argmin} \mathcal{J}(\boldsymbol{\theta}; \mathcal{D})$$

Proof. We write the loss function $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D})$ in vector form $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}) = \frac{1}{n}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta}\|_2^2$, where the ith entry of $\boldsymbol{y} \in \mathbb{R}^n$ is y_i and the ith row of $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ is $\boldsymbol{x}_i^\top$. Note that the gradient of the loss for any value of $\boldsymbol{\theta}$ is contained the subspace $\boldsymbol{S}$, as $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}; \mathcal{D}) = \frac{2}{n}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})$ and $\operatorname{im}(\boldsymbol{X}^\top) = \boldsymbol{S}$. Further, the initial iterate of $\mathcal{A}^k$ is $\boldsymbol{0} \in \boldsymbol{S}$. Since subspaces are closed under addition, every iterate of gradient descent on $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D})$ starting from $\boldsymbol{0}$ must be contained in $\boldsymbol{S}$. Thus if $\mathcal{A}^k(\mathcal{D})$ converges, it must converge to a zero of the gradient of the loss, and this point must also be in $\boldsymbol{S}$. Since the loss is convex, this point must be a loss minimizer. □

In this case, the original training solution $\boldsymbol{\theta}^*$ which results from simply performing gradient descent interpolates all of $\mathcal{D}$ and lies on $\boldsymbol{S}$, the span of the input samples in $\mathcal{D}$. Then, given an unlearning request to forget any subset $\mathcal{D}_f$ from $\mathcal{D}$, $\boldsymbol{\theta}^*$ itself is a minimizer to the loss on the resulting retain set $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$. However, since $\boldsymbol{\theta}^* \in \boldsymbol{S}$ reveals information about all the input samples in $\mathcal{D}$, it necessarily leaks information about the samples in $\mathcal{D}_f$. Thus, even though $\boldsymbol{\theta}^*$ is a valid minimizer of $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}_r)$, it is not an acceptable unlearning solution.

D. Proofs

D.1. Proof of Theorem 2.2

We assume $f(\boldsymbol{\theta}^*, \cdot)$ interpolates all of $\mathcal{D}$, so $f(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{y}$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$, and that the sample-wise loss $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ is minimized when $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{y}$. Thus, $\boldsymbol{\theta}^*$ must minimize each of the sample-wise losses $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$. Therefore, $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^*, \boldsymbol{x}, \boldsymbol{y}) = \mathbf{0}$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$.

Since $\mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_r) = \frac{1}{|\mathcal{D}_r|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D}_r} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ and $\mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_f) = \frac{1}{|\mathcal{D}_f|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{D}_f} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$, we must have that $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_r) = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_f) = \mathbf{0}$.

Then, if M_{LG} is any loss-gradient unlearning method, the update rule must be of the form

$$M(\mathcal{A}, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f) = \boldsymbol{\theta}^* - \boldsymbol{P}_r \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_r) + \boldsymbol{P}_f \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_f) + \boldsymbol{\xi},$$

where $\boldsymbol{P}_r$ and $\boldsymbol{P}_f$ are positive semi-definite matrices and $\boldsymbol{\xi}$ is a zero-mean random variable. Applying the fact that $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_r) = \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^*; \mathcal{D}_f) = \mathbf{0}$ to the update of M_{LG} gives the desired result:

$$M_{\text{LG}}(\mathcal{A}, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f) = \boldsymbol{\theta}^* + \boldsymbol{\xi}$$

D.2. Proof of Theorem 4.1

Recall we have a feasible vector $\boldsymbol{\theta}^*$ such that $\boldsymbol{\theta}^{*\top} \boldsymbol{x} = y$ for all $(\boldsymbol{x}, y) \in \mathcal{D}$, and we want to recover $\boldsymbol{\theta}_r^*$, the minimum ℓ_2 norm solution over just a subset $\mathcal{D}_r \subseteq \mathcal{D}$:

$$\boldsymbol{\theta}_r^* = \operatorname*{argmin}_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2 \quad \text{s.t. } \boldsymbol{\theta}^\top \boldsymbol{x} = y \quad \forall (\boldsymbol{x}, y) \in \mathcal{D}_r \tag{12}$$

Consider solving the relaxed unlearning problem (5) for $\tilde{R}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$:

$$\tilde{\boldsymbol{\Delta}} = \operatorname*{argmin}_{\boldsymbol{\Delta}} \|\boldsymbol{\theta}^* + \boldsymbol{\Delta}\|_2 \text{ s.t. } \boldsymbol{\Delta} \perp \boldsymbol{x} \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_r$$

Define $\boldsymbol{S}_r = \operatorname{span}\{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}_r\}$ and write the equivalent problem:

$$\tilde{\boldsymbol{\Delta}} = \operatorname*{argmin}_{\boldsymbol{\Delta} \in \boldsymbol{S}_r^\perp} \frac{1}{2} \|\boldsymbol{\theta}^* + \boldsymbol{\Delta}\|_2^2$$

By first order optimality, $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}} \in \boldsymbol{S}_r$, so we must have that

$$\tilde{\boldsymbol{\Delta}} = -\mathcal{P}_{\boldsymbol{S}_r^\perp}(\boldsymbol{\theta}^*)$$

Thus the updated unlearned vector is

$$\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}} = \boldsymbol{\theta}^* - \mathcal{P}_{\boldsymbol{S}_r^\perp}(\boldsymbol{\theta}^*) = \mathcal{P}_{\boldsymbol{S}_r}(\boldsymbol{\theta}^*).$$

Then, $\mathcal{P}_{\boldsymbol{S}_r}(\boldsymbol{\theta}^*) = \boldsymbol{\theta}_r^*$ by Lemma B.3.

D.3. Proof of Theorem 4.2

Before proving the theorem, we first show an intermediate result.

Lemma D.1. *Consider the linear network model class with parameters $\boldsymbol{\theta}$ partitioned as $\boldsymbol{\theta} = [\boldsymbol{c}; \operatorname{vec}(\boldsymbol{A}_1); \ldots; \operatorname{vec}(\boldsymbol{A}_{L-1})]$ and $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{c}^\top \boldsymbol{A}_{L-1} \cdots \boldsymbol{A}_1 \boldsymbol{x}$. Denote the retain set input subspace by $\boldsymbol{S}_r = \operatorname{span}\{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}_r\}$ and partition the perturbation as $\tilde{\boldsymbol{\Delta}} = \left[\tilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}; \operatorname{vec}(\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}); \ldots; \operatorname{vec}(\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_{L-1}})\right]$ in the same manner as $\boldsymbol{\theta}$. Set*

$$\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1} = -\left\|\boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*\right\|_2^{-2} \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \mathcal{P}_{\boldsymbol{S}_r^\perp}(\mathbf{w}(\boldsymbol{\theta}^*))^\top \tag{13}$$

and all other components of $\tilde{\boldsymbol{\Delta}}$ to zero. Then $\tilde{\boldsymbol{\Delta}}$ is orthogonal to the gradient of mapping $f(\boldsymbol{\theta}, \boldsymbol{x})$ evaluated at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ for each input $\boldsymbol{x}$ in the retain set and hence feasible for the relaxed problem (5). Moreover, $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ solves the exact unlearning problem (3) for $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$.

D.3.1. Proof of Lemma D.1

Recall that in this case we are interested in minimizing $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$, where $\mathbf{w}(\boldsymbol{\theta}) = \boldsymbol{A}_1^\top \cdots \boldsymbol{A}_{L-1}^\top \boldsymbol{c}$ returns the effective linear predictor parameterized by $\boldsymbol{\theta}$.

We first show that $\tilde{\boldsymbol{\Delta}}$ is feasible for the relaxed problem (5). Firstly, $\tilde{\boldsymbol{\Delta}}$ is zero in all entries except those corresponding to the perturbation of $\boldsymbol{A}_1$, so we only need to ensure that $\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}$ is orthogonal to $\nabla_{\boldsymbol{A}_1} f(\boldsymbol{\theta}^*, \boldsymbol{x})$ for each $(\boldsymbol{x}, y) \in \mathcal{D}_r$. Recall we denote the retain set input space as $\boldsymbol{S}_r = \operatorname{span}\{\, \boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}_r \,\}$, and $\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}$ is defined as

$$\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1} = -\left\|\boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*\right\|_2^{-2} \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \mathcal{P}_{\boldsymbol{S}_r^\perp}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right)^\top .$$

Further, the gradients are computed as

$$\nabla_{\boldsymbol{A}_1} f(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \boldsymbol{x}^\top$$

Then for any $(\boldsymbol{x}, y) \in \mathcal{D}_r$,

$$\begin{aligned}
\langle \tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}, \nabla_{\boldsymbol{A}_1} f(\boldsymbol{\theta}^*, \boldsymbol{x}) \rangle &= \operatorname{tr}\left\{ \left(\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}\right)^\top \nabla_{\boldsymbol{A}_1} f(\boldsymbol{\theta}^*, \boldsymbol{x}) \right\} \\
&= \operatorname{tr}\left\{ \nabla_{\boldsymbol{A}_1} f(\boldsymbol{\theta}^*, \boldsymbol{x}) \left(\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}\right)^\top \right\} \\
&= -\left\|\boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*\right\|_2^{-2} \operatorname{tr}\left\{ \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \boldsymbol{x}^\top \mathcal{P}_{\boldsymbol{S}_r^\perp}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right) \boldsymbol{c}^{*\top} \boldsymbol{A}_{L-1}^* \cdots \boldsymbol{A}_2^* \right\} \\
&= 0,
\end{aligned}$$

where the last step follows from the fact that the inner term $\boldsymbol{x}^\top \mathcal{P}_{\boldsymbol{S}_r^\perp}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right) = 0$ since $\boldsymbol{x} \in \mathcal{D}_r$ implies $\boldsymbol{x} \in \boldsymbol{S}_r$ by definition.

We now show that $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ achieves the optimal unlearning solution $\boldsymbol{\theta}^*$. By construction of $\tilde{\boldsymbol{\Delta}}$, the only entries of $\boldsymbol{\theta}^*$ that are perturbed are those which correspond to $\boldsymbol{A}_1$. Thus, we compute the effective linear predictor after the perturbation:

$$\begin{aligned}
\mathbf{w}(\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}) &= \mathbf{w}(\boldsymbol{\theta}^*) + \tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}^\top \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \\
&= \mathbf{w}(\boldsymbol{\theta}^*) - \left\|\boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*\right\|_2^{-2} \mathcal{P}_{\boldsymbol{S}_r^\perp}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right) \boldsymbol{c}^{*\top} \boldsymbol{A}_{L-1}^* \cdots \boldsymbol{A}_2^* \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \\
&= \mathbf{w}(\boldsymbol{\theta}^*) - \left\|\boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*\right\|_2^{-2} \mathcal{P}_{\boldsymbol{S}_r^\perp}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right) \left(\boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*\right)^\top \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \\
&= \mathbf{w}(\boldsymbol{\theta}^*) - \mathcal{P}_{\boldsymbol{S}_r^\perp}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right) \\
&= \mathcal{P}_{\boldsymbol{S}_r}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right)
\end{aligned}$$

Since the linear predictor $\mathbf{w}(\boldsymbol{\theta}^*)$ already interpolated $\mathcal{D}$, $\mathcal{P}_{\boldsymbol{S}_r}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right)$ must be the minimum norm linear predictor over $\mathcal{D}_r$ by Lemma B.3. Thus, the effective predictor of the perturbed parameters $\mathbf{w}(\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}})$ solves the exact unlearning problem (3) when $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$, so $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ achieves the optimal unlearning solution.

D.3.2. Proof of the Theorem

Let $\tilde{\boldsymbol{\Delta}}$ be the perturbation which satisfies the conditions in Lemma D.1. Then, $\tilde{\boldsymbol{\Delta}}$ is feasible for the relaxed problem (5), and further $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ solves the exact unlearning problem (3).

Now, let $\boldsymbol{\Delta}^*$ minimize the relaxed problem (5) for this $\tilde{R}$ defined in (6). Then because $\tilde{R}$ ensures that all elements of $\boldsymbol{\Delta}^*$ which do not correspond to $\boldsymbol{A}_1$ are zero, we must have that for any $(\boldsymbol{x}, y) \in \mathcal{D}_r$:

$$\begin{aligned}
\mathbf{w}(\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*)^\top \boldsymbol{x} &= \boldsymbol{c}^{*\top} \boldsymbol{A}_{L-1}^* \cdots \boldsymbol{A}_2^* \left(\boldsymbol{A}_1^* + \boldsymbol{\Delta}_{\boldsymbol{A}_1}^*\right) \boldsymbol{x} \\
&= y + \boldsymbol{c}^{*\top} \boldsymbol{A}_{L-1}^* \cdots \boldsymbol{A}_2^* \boldsymbol{\Delta}_{\boldsymbol{A}_1}^* \boldsymbol{x} \\
&= y + \langle \boldsymbol{\Delta}^*, \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*, \boldsymbol{x}) \rangle \\
&= y,
\end{aligned}$$

where the last equality follows from the feasibility of $\boldsymbol{\Delta}^*$ to (5). Thus, $\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*$ interpolates $\mathcal{D}_r$, so $\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*$ is feasible for the exact unlearning problem (3). We now show this point is also optimal for (3).

Since $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ solves the exact unlearning problem (3) and $\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*$ is another feasible point, we must have that

$$R(\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}) \leq R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*).$$

Further, both $\tilde{\boldsymbol{\Delta}}$ and $\boldsymbol{\Delta}^*$ are feasible for (5) and $\boldsymbol{\Delta}^*$ is defined as the solution to (5), so we must have that

$$\tilde{R}(\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*) \leq \tilde{R}(\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}).$$

But, since both $\tilde{\boldsymbol{\Delta}}$ and $\boldsymbol{\Delta}^*$ are non-zero only in the entries corresponding to $\boldsymbol{A}_1$, applying R and $\tilde{R}$ yields the same value:

$$R(\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}) = \tilde{R}(\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}) \quad \text{and} \quad R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*) = \tilde{R}(\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*)$$

Thus, $R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*) = R(\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}})$, so $\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*$ achieves the optimal objective value of (3). Since we established feasibility and optimality, $\boldsymbol{\theta}^* + \boldsymbol{\Delta}^*$ must solve (3).

D.4. Proof of Theorem 4.3

Denote the minimum ℓ_2 norm solution $\mathbf{w}(\hat{\boldsymbol{\theta}}_r^*)$ to $\boldsymbol{y} = \boldsymbol{X}\mathbf{w}$ as just $\mathbf{w}_r^*$ for brevity. Using $\mathbf{w}_r^*$, we construct a solution to the exact unlearning problem (3) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$, which we restate below:

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\boldsymbol{\theta}\|_2 \text{ s.t. } \mathbf{w}(\boldsymbol{\theta})^\top \boldsymbol{x} = y \quad \forall (\boldsymbol{x}, y) \in \mathcal{D}_r$$

Expanding $\boldsymbol{\theta} = \left[\boldsymbol{c}\,;\, \operatorname{vec}(\boldsymbol{A}_1)\,;\, \ldots\,;\, \operatorname{vec}(\boldsymbol{A}_{L-1})\right]$ into the sub-parameters, squaring the objective, and organizing $(\boldsymbol{x}, y) \in \mathcal{D}_r$ into input data matrix $\boldsymbol{X}_r \in \mathbb{R}^{|\mathcal{D}_r| \times d}$ and output vector $\boldsymbol{y}_r \in \mathbb{R}^{|\mathcal{D}_r|}$ gives an equivalent problem:

$$\underset{\boldsymbol{c}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_{L-1}}{\operatorname{argmin}} \|\boldsymbol{c}\|_2^2 + \sum_{\ell=1}^{L-1} \|\boldsymbol{A}_\ell\|_F^2 \text{ s.t. } \boldsymbol{y}_r = \boldsymbol{X}_r \boldsymbol{A}_1^\top \ldots \boldsymbol{A}_{L-1}^\top \boldsymbol{c} \tag{14}$$

Let $\boldsymbol{c}^*, \boldsymbol{A}_1^*, \ldots, \boldsymbol{A}_{L-1}^*$ be a solution to (14). Then, $\boldsymbol{A}_1^{*\top} \ldots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*$ interpolates $\mathcal{D}_r$, so $\boldsymbol{A}_1^{*\top} \ldots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* = \mathbf{w}_r^* + \boldsymbol{z}$ where $\mathbf{w}_r^* \in \operatorname{row}(\boldsymbol{X}_r)$ and $\boldsymbol{z} \in \ker(\boldsymbol{X}_r)$ by Lemma B.2.

Let $\boldsymbol{P}_{\mathbf{w}_r^*} = \frac{1}{\|\mathbf{w}_r^*\|_2^2} \mathbf{w}_r^* \mathbf{w}_r^{*\top}$ be the projection matrix onto $\operatorname{span}(\mathbf{w}_r^*)$. Then replacing $\boldsymbol{A}_1^*$ with $\boldsymbol{A}_1^* \boldsymbol{P}_{\mathbf{w}_r^*}$ maintains feasibility since $\boldsymbol{P}_{\mathbf{w}_r^*}^\top \boldsymbol{A}_1^{*\top} \ldots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* = \boldsymbol{P}_{\mathbf{w}_r^*} (\mathbf{w}_r^* + \boldsymbol{z}) = \mathbf{w}_r^*$ which is feasible by definition. Further, $\boldsymbol{A}_1^* \boldsymbol{P}_{\mathbf{w}_r^*}$ achieves smaller objective function value since

$$\left\|\boldsymbol{A}_1^* \boldsymbol{P}_{\mathbf{w}_r^*}\right\|_F^2 = \operatorname{tr}\{\boldsymbol{A}_1^* \boldsymbol{P}_{\mathbf{w}_r^*} \boldsymbol{P}_{\mathbf{w}_r^*} \boldsymbol{A}_1^{*\top}\} = \operatorname{tr}\{\boldsymbol{P}_{\mathbf{w}_r^*} \boldsymbol{A}_1^{*\top} \boldsymbol{A}_1^*\} \leq \left\|\boldsymbol{P}_{\mathbf{w}_r^*}\right\|_2 \left\|\boldsymbol{A}_1^{*\top} \boldsymbol{A}_1^*\right\|_* = \|\boldsymbol{A}_1^*\|_F^2 .$$

The second equality follows from the cyclic property of trace and the fact that $\boldsymbol{P}_{\mathbf{w}_r^*}$ is both symmetric and idempotent, and the inequality is a generalized Hölder's inequality for matrices.

Thus, replacing $\boldsymbol{A}_1^*$ with the rank-1 matrix $\boldsymbol{A}_1^* \boldsymbol{P}_{\mathbf{w}_r^*}$ must preserve optimality of any solution that contains $\boldsymbol{A}_1^*$. Write $\boldsymbol{A}_1^* \boldsymbol{P}_{\mathbf{w}_r^*} = \lambda_1 \boldsymbol{v}_1 \mathbf{w}_r^{*\top}$ for some $\lambda_1 \in \mathbb{R}$, $\boldsymbol{v}_1 \in \mathbb{R}^{h_\ell}$ with $\|\boldsymbol{v}_1\|_2 = 1$.

We can apply an analogous argument with the matrix $\boldsymbol{P}_{\boldsymbol{v}_1}$, which projects its input onto $\operatorname{span}(\boldsymbol{v}_1)$, to show that any solution that contains $\boldsymbol{A}_2^*$ must remain optimal with $\boldsymbol{A}_2^*$ replaced by the rank-1 matrix $\boldsymbol{A}_2^* \boldsymbol{P}_{\boldsymbol{v}_1}$. Continuing this argument for each $\boldsymbol{A}_\ell^*$, $\ell = 3, \ldots, L-1$ as well as for $\boldsymbol{c}^*$ shows that we can search for solution over a much smaller space. Specifically, for some $\lambda_\ell \in \mathbb{R}$ and $\boldsymbol{v} \in \mathbb{R}^{h_\ell}$, we can decompose $\boldsymbol{c}^*$ and each $\boldsymbol{A}_\ell^*$ as

$$\boldsymbol{A}_1^* = \lambda_1 \boldsymbol{v}_1 \mathbf{w}_r^{*\top} \qquad \boldsymbol{A}_\ell^* = \lambda_\ell \boldsymbol{v}_\ell \boldsymbol{v}_{\ell-1}^\top \text{ for } \ell = 2, \ldots, L-1 \qquad \boldsymbol{c}^* = \lambda_L \boldsymbol{v}_{L-1}$$

Then, (14) reduces to

$$\min_{\lambda_i, \boldsymbol{v}_\ell} \|\lambda_L \boldsymbol{v}_{L-1}\|_2^2 + \left\|\lambda_1 \boldsymbol{v}_1 \mathbf{w}_r^{*\top}\right\|_F^2 + \sum_{\ell=2}^{L-1} \left\|\lambda_\ell \boldsymbol{v}_\ell \boldsymbol{v}_{\ell-1}^\top\right\|_F^2$$
$$\text{s.t.} \quad (\lambda_1 \mathbf{w}_r^* \boldsymbol{v}_1^\top)(\lambda_2 \boldsymbol{v}_1 \boldsymbol{v}_2^\top) \cdots (\lambda_{L-1} \boldsymbol{v}_{L-2} \boldsymbol{v}_{L-1}^\top)(\lambda_L \boldsymbol{v}_{L-1}) = \mathbf{w}_r^* \text{ and } \|\boldsymbol{v}_\ell\|_2 = 1$$
$$= \min_{\lambda_i} \|\mathbf{w}_r^*\|_2^2 \lambda_1^2 + \sum_{\ell=2}^{L} \lambda_\ell^2 \quad \text{s.t.} \quad \lambda_1 \lambda_2 \cdots \lambda_L = 1 \tag{15}$$

We perform a change of variables setting $\gamma_i = \lambda_i^2$ and enforcing $\gamma_i > 0$.

$$\min_{\gamma_i > 0} \|\mathbf{w}_r^*\|_2^2 \gamma_1 + \sum_{\ell=2}^{L} \gamma_\ell \quad \text{s.t.} \quad \gamma_1 \gamma_2 \cdots \gamma_L = 1 \tag{16}$$

Define $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_L)$, objective function $g(\boldsymbol{\gamma}) = \|\mathbf{w}_r^*\|_2^2 \gamma_1 + \sum_{\ell=2}^{L} \gamma_\ell$, and constraint $h(\boldsymbol{\gamma}) = \gamma_1 \gamma_2 \cdots \gamma_L - 1 = 0$. By the AM-GM inequality, we have that for any feasible $\boldsymbol{\gamma}$

$$g(\boldsymbol{\gamma}) \geq L \left(\|\mathbf{w}_r^*\|_2^2 \gamma_1 \cdots \gamma_L\right)^{\frac{1}{L}} = L \|\mathbf{w}_r^*\|_2^{2/L},$$

where the last equality follows from the constraint $h(\boldsymbol{\gamma}) = 0$. Define feasible point $\boldsymbol{\gamma}^*$ such that

$$\boldsymbol{\gamma}^* = \left(\|\mathbf{w}_r^*\|_2^{\frac{2(1-L)}{L}}, \|\mathbf{w}_r^*\|_2^{\frac{2}{L}}, \ldots, \|\mathbf{w}_r^*\|_2^{\frac{2}{L}}\right).$$

Then $g(\boldsymbol{\gamma}^*) = \|\mathbf{w}_r^*\|_2^{2/L}$ achieves the lower bound, so it must solve (16). Thus, the optimal values $\lambda_1^*, \ldots, \lambda_L^*$ to (15) result from taking square roots of $\boldsymbol{\gamma}_\ell^*$. Then, the following values for the network parameters must be optimal for (14):

$$\boldsymbol{A}_1^* = \|\mathbf{w}_r^*\|_2^{\frac{(1-L)}{L}} \boldsymbol{v}_1 \mathbf{w}_r^{*\top} \qquad \boldsymbol{A}_\ell^* = \|\mathbf{w}_r^*\|_2^{\frac{1}{L}} \boldsymbol{v}_\ell \boldsymbol{v}_{\ell-1}^\top \text{ for } \ell = 2, \ldots, L-1 \qquad \boldsymbol{c}^* = \|\mathbf{w}_r^*\|_2^{\frac{1}{L}} \boldsymbol{v}_{L-1}.$$

D.5. Proof of Theorem 4.4

We prove the theorem using the following lemma. See the end of the section for a proof.

Lemma D.2. *For $\boldsymbol{c} \in \mathbb{R}^h$ and subspace $\mathcal{G} \subseteq \mathbb{R}^h$ such that $\dim(\mathcal{G}) = s$, there exists $\boldsymbol{\Delta}_c \in \mathcal{G}_r^\perp$ such that $\|\boldsymbol{c} + \boldsymbol{\Delta}_c\|_0 \leq s$, where the ℓ_0-"norm" $\|\cdot\|_0$ counts the number of non-zero elements.*

Because $\hat{R}$ does not allow any perturbation of $\boldsymbol{A}^*$, any solution to (8) must only perturb $\boldsymbol{\theta}^*$ in the entries corresponding to $\boldsymbol{c}^*$. Let $s = \dim(\operatorname{span}\{\phi(\boldsymbol{A}^* \boldsymbol{x})\}_{(\boldsymbol{x},y) \in \mathcal{D}_r})$. Note that by definition we have that $s \leq |\mathcal{D}_r|$. Apply the lemma to $\boldsymbol{c}^*$ and $\operatorname{span}\{\phi(\boldsymbol{A}^* \boldsymbol{x})\}_{(\boldsymbol{x},y) \in \mathcal{D}_r}$ so that there exists $\tilde{\boldsymbol{\Delta}}_c \in \operatorname{span}\left(\{\phi(\boldsymbol{A}^* \boldsymbol{x})\}_{(\boldsymbol{x},y) \in \mathcal{D}_r}\right)^\perp$ such that $\|\boldsymbol{c}^* + \tilde{\boldsymbol{\Delta}}_c\|_0 \leq s$. Define $\tilde{\boldsymbol{\Delta}} = \left[\tilde{\boldsymbol{\Delta}}_c \,;\, \mathbf{0}\right]$.

Then the network defined by $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ has at most s active neurons since any zero element of $\boldsymbol{c}^* + \tilde{\boldsymbol{\Delta}}_c$ cannot contribute an active neuron. Further, $\{\phi(\boldsymbol{A}^* \boldsymbol{x})\}_{(\boldsymbol{x},y) \in \mathcal{D}_r} = \{\nabla_{\boldsymbol{c}} f(\boldsymbol{\theta}^*, \boldsymbol{x})\}_{\boldsymbol{x},y \in \mathcal{D}_r}$, so the perturbation $\tilde{\boldsymbol{\Delta}}$ is feasible for the relaxed problem (5). But, f is linear in $\boldsymbol{c}$, so this perturbation must preserve function value on $\mathcal{D}_r$, since the constraints of the relaxed problem are tight when just perturbing $\boldsymbol{c}^*$. Thus, the resulting network defined by $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ both interpolates $\mathcal{D}_r$ and has at most $s = \dim(\operatorname{span}\{\phi(\boldsymbol{A}^* \boldsymbol{x})\}_{(\boldsymbol{x},y) \in \mathcal{D}_r})$ active neurons.

Proof of Lemma D.2:

Let the columns of some $\boldsymbol{P} \in \mathbb{R}^{h \times (h-s)}$ form a basis for $\mathcal{G}^\perp$ so that $\operatorname{im}(\boldsymbol{P}) = \mathcal{G}^\perp$. Consider the reduced column echelon form of $\boldsymbol{P}$ denoted $\operatorname{rcef}(\boldsymbol{P}) = \tilde{\boldsymbol{P}}$. By definition, $\operatorname{im}(\tilde{\boldsymbol{P}}) = \operatorname{im}(\boldsymbol{P}) = \mathcal{G}^\perp$, so $\operatorname{rank}(\tilde{\boldsymbol{P}}) = h - s$ and thus each of the $h - s$

Algorithm 1 MinNorm-OG

1: **Input:** $\boldsymbol{\theta}^*$, loss $\mathcal{L}(\boldsymbol{\theta})$, retained subset $\mathcal{D}_r' \subseteq \mathcal{D}_r$, step size η_t, regularization constant $\lambda_t \geq 0$, subsample size n_{pert}
2: Initialize $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^*$
3: **for** $t = 1$ to n_{epochs} **do**
4: **for** each batch $\mathcal{B}$ from $\mathcal{D}_r'$ **do**
5: **if** $\lambda_t < \infty$ **then**
6: Set $\boldsymbol{g}_i = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{x}_i)$ for $x_i \in \mathcal{B}$, $i = 1, \ldots, n_{\text{pert}}$
7: Solve: $\tilde{\boldsymbol{\Delta}} = \arg\min_{\boldsymbol{\Delta}} \|\boldsymbol{\theta} + \boldsymbol{\Delta}\|_2^2 + \lambda_t \|\boldsymbol{\Delta}\|_2^2$, s.t. $\boldsymbol{\Delta} \perp \boldsymbol{g}_i$ for all i
8: Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \tilde{\boldsymbol{\Delta}}$
9: **end if**
10: Loss descent: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_t \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$
11: **end for**
12: **end for**
13: **return** $\boldsymbol{\theta}$

columns of $\tilde{\boldsymbol{P}}$ has a leading one. Let $\tilde{\boldsymbol{p}}_i$ be the ith column of $\tilde{\boldsymbol{P}}$ and let j_i denote the index of the leading one in $\tilde{\boldsymbol{p}}_i$ for all $i \in [h - s]$.

Let $(\tilde{\boldsymbol{p}}_i)_k$ denote the kth element of $\tilde{\boldsymbol{p}}_i$. By definition of the reduced column echelon form, we have that $(\tilde{\boldsymbol{p}}_i)_k = 0$ for all $k < j_i$. Define

$$\boldsymbol{\Delta}_c = \sum_{i=1}^{h-s} \gamma_i \tilde{\boldsymbol{p}}_i$$

for coefficients $\gamma_i \in \mathbb{R}$ defined as

$$\gamma_i = -\left(\boldsymbol{c}^* + \sum_{k=1}^{i-1} \tilde{\boldsymbol{p}}_k \right)_{j_i}$$

Since each $\tilde{\boldsymbol{p}}_i$ is only non-zero in the indices j_i to h, we must have that $(\boldsymbol{c}^* + \boldsymbol{\Delta}_c)_{j_i} = 0$ for all $i \in [h - s]$, so $\|\boldsymbol{c}^* + \boldsymbol{\Delta}_c\|_0 \leq s$.

E. MinNorm-OG Algorithm

We first present the pseudocode for our algorithm MinNorm-OG as Algorithm 1.

Step 7 of Algorithm 1 solves a version of the relaxed unlearning problem (5) for the specific choice $\tilde{R}(\boldsymbol{\theta} + \boldsymbol{\Delta}) = \|\boldsymbol{\theta} + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2$. We next derive its closed form solution and show how it can be obtained using a projection.

Define the span of the model gradients over the subsampled batch as the subspace $\mathcal{G}_r = \operatorname{span}\{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{x}_i) \mid \boldsymbol{x}_i \in \mathcal{B}, i = 1, \ldots, n_{\text{pert}}\}$ and consider any $\lambda \geq 0$. We then solve the following problem:

$$\tilde{\boldsymbol{\Delta}} = \operatorname*{argmin}_{\boldsymbol{\Delta}} \|\boldsymbol{\theta} + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2 \quad \text{s.t. } \boldsymbol{\Delta} \in \mathcal{G}_r^{\perp}. \tag{17}$$

This is a strongly convex problem over a linear constraint, so its solution $\tilde{\boldsymbol{\Delta}}$ is the unique point which satisfies the following condition for first order optimality:

$$(1 + \lambda)\tilde{\boldsymbol{\Delta}} + \boldsymbol{\theta} \in \mathcal{G}_r.$$

Note that this is satisfied by the projection

$$\tilde{\boldsymbol{\Delta}} = -\frac{1}{1 + \lambda} \mathcal{P}_{\mathcal{G}_r^{\perp}}(\boldsymbol{\theta}),$$

which must then be the unique solution to (17).

F. Experiments

We first standardize the notation for each algorithm. Throughout our experiments, we sweep over hyperparameters and report the best results for each algorithm, and we sweep related hyperparameters for each algorithm through the same set of values. For example, every algorithm has a learning rate which is selected from searching over the same set of values. We first define the hyperparameter names we use along with the algorithms they apply to.

Table 1. Hyperparameter definitions and their associated methods.

Symbol	Methods	Description
T	All	Number of epochs
η	All	Learning rate
λ_{GA}	NGP, Scrub	Loss ascent coefficient
λ_{reg}	NPO, Scrub, MinNorm-OG, Ridge	Regularization coefficient
σ	NGD	Gradient noise standard deviation
T_{GD}	Scrub, MinNorm-OG	Number of final descent epochs on retain set
γ_{reg}	MinNorm-OG, Ridge	Regularization coefficient decay rate
T_{Proj}	MinNorm-OG	Projection period
n_{pert}	MinNorm-OG	Subsample size to compute gradient space

F.1. Implementations

We now define the exact implementation of each method. Consider a batch of retain samples $\mathcal{B}_r$ and forget samples $\mathcal{B}_f$, along with loss function $\mathcal{J}$. For each method, we use the AdamW optimizer with learning rate η on different effective loss functions. We express the loss functions below.

F.1.1. GD

$$\mathcal{J}_{\text{GD}}(\boldsymbol{\theta}\,;\mathcal{B}_r) = \mathcal{J}\left(\boldsymbol{\theta}\,;\mathcal{B}_r\right)$$

F.1.2. GA

$$\mathcal{J}_{\text{GA}}(\boldsymbol{\theta}\,;\mathcal{B}_f) = -\mathcal{J}\left(\boldsymbol{\theta}\,;\mathcal{B}_f\right)$$

F.1.3. NGD

$$\mathcal{J}_{\text{NGD}}(\boldsymbol{\theta}\,;\mathcal{B}_r) = \mathcal{J}\left(\boldsymbol{\theta}\,;\mathcal{B}_r\right) + \boldsymbol{\theta}^\top\boldsymbol{\xi},$$

where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I})$ is a zero-mean Gaussian random vector.

F.1.4. NGP

$$\mathcal{J}_{\text{NGP}}(\boldsymbol{\theta}\,;\mathcal{B}_r) = \mathcal{J}\left(\boldsymbol{\theta}\,;\mathcal{B}_r\right) - \lambda_{\text{GA}}\mathcal{J}\left(\boldsymbol{\theta}\,;\mathcal{B}_f\right)$$

F.1.5. NPO

Recall that $\boldsymbol{\theta}^*$ denotes the initial trained model parameters. Then, the NPO loss is

$$\mathcal{J}_{\text{NPO}}\left(\boldsymbol{\theta}_0\,;\mathcal{B}_f, \lambda_{\text{GA}}\right) = \frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x}_f, y_f)\in\mathcal{B}_f}\frac{2}{\lambda_{\text{GA}}}\log\left(1 + \frac{\pi_{\boldsymbol{\theta}_0}(y_f \mid \boldsymbol{x}_f)}{\pi_{\boldsymbol{\theta}^*}(y_f \mid \boldsymbol{x}_f)}\right)^{\lambda_{\text{GA}}},$$

where $\pi_{\boldsymbol{\theta}}(y_f \mid \boldsymbol{x}_f)$ denotes the model's predicted probability of class y_f for input $\boldsymbol{x}_f$ for parameter vector $\boldsymbol{\theta}$. Note that this is equivalent to setting the parameter β in the original NPO paper (Zhang et al., 2024) to λ_{reg}.

F.1.6. SCRUB

The Scrub loss decomposes into different terms depending on the epoch. Let $\pi_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ denote the model's predicted distribution over classes $\boldsymbol{y}$ for input $\boldsymbol{x}$ for parameter vector $\boldsymbol{\theta}$, and define $\mathrm{KL}(\cdot \,\|\, \cdot)$ as the Kullback-Leiber divergence. Recall $\boldsymbol{\theta}^*$ denotes the initial trained model parameters, and denote the current epoch $t \in \{0, \ldots, T-1\}$. Then the Scrub loss $\mathcal{J}_{\text{Scrub}}(\boldsymbol{\theta}\,; \mathcal{B}_r, \mathcal{B}_f, \lambda_{\text{reg}}, \lambda_{\text{GA}}, t)$ is defined as:

$$\mathcal{J}_{\text{Scrub}}(\boldsymbol{\theta}\,; \mathcal{B}_r, \mathcal{B}_f, \lambda_{\text{reg}}, \lambda_{\text{GA}}, t) = \begin{cases} \mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{B}_r\right) + \frac{\lambda_{\text{reg}}}{|\mathcal{B}_r|} \sum\limits_{(\boldsymbol{x}_r, y_r) \in \mathcal{B}_r} \mathrm{KL}(\pi_{\boldsymbol{\theta}^*}(\boldsymbol{y} \mid \boldsymbol{x}_r) \,\|\, \pi_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}_r)) & \text{if } t \text{ even or } t \geq T - T_{\text{GD}} \\ -\frac{\lambda_{\text{GA}}}{|\mathcal{B}_f|} \sum\limits_{(\boldsymbol{x}_f, y_f) \in \mathcal{B}_f} \mathrm{KL}(\pi_{\boldsymbol{\theta}^*}(\boldsymbol{y} \mid \boldsymbol{x}_f) \,\|\, \pi_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}_f)) & \text{otherwise} \end{cases}$$

F.1.7. MINNORM-OG

For each batch $\mathcal{B}_r$, we always perform a loss descent step:

$$\mathcal{J}_{\text{MinNorm-OG}}(\boldsymbol{\theta}\,; \mathcal{B}_r) = \mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{B}_r\right)$$

Following the AdamW update for this loss, we then (depending on the epoch) perform the model update corresponding to solving the relaxed unlearning problem (5) for $\tilde{R}(\boldsymbol{\theta} + \boldsymbol{\Delta}) = \|\boldsymbol{\theta} + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2$, where λ is a saved parameter of the algorithm. We use the parameters T_{Proj} and T_{GD} to determine which epochs to perform the unlearning update. For the T_{GD} last epochs, we only perform the descent step and skip the unlearning update, similar to Scrub. In the first $T - T_{\text{GD}}$ epochs, we perform the unlearning update every T_{Proj} epochs.

We initialize $\lambda = \frac{1}{\lambda_{\text{reg}}} - 1$, and each time we perform the unlearning update, we grow the value of λ through the update $\lambda \leftarrow \frac{\lambda+1}{\gamma_{\text{reg}}} - 1$ using the decay factor $\gamma_{\text{reg}} \in [0, 1]$. For our algorithm we only use values of λ_{reg} such that $\lambda_{\text{reg}} \leq 1$. The update for λ leads to solutions to the relaxed unlearning problem which result in more conservative perturbations.

To interpret these values, first recall that we solve the relaxed unlearning problem over a subsample of each batch $\mathcal{B}_r' \subseteq \mathcal{B}_r$ where $|\mathcal{B}_r'| = n_{\text{pert}}$. For convenience, define the gradient subspace $\mathcal{G}_r' = \operatorname{span}\{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{x})\}_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{B}_r'}$. As we showed in Appendix E, for any value of λ, the optimal perturbation is then $\tilde{\boldsymbol{\Delta}} = -\frac{1}{1+\lambda} \mathcal{P}_{\mathcal{G}_r'^{\perp}}(\boldsymbol{\theta})$. Thus, the initial value $\lambda = \frac{1}{\lambda_{\text{reg}}} - 1$ leads to the perturbation $\tilde{\boldsymbol{\Delta}} = -\lambda_{\text{reg}} \mathcal{P}_{\mathcal{G}_r'^{\perp}}(\boldsymbol{\theta})$. Further, the coefficient update $\lambda = \frac{\lambda'+1}{\gamma_{\text{reg}}} - 1$ leads to a more conservative unlearning perturbation $\tilde{\boldsymbol{\Delta}} = -\gamma_{\text{reg}} \frac{1}{1+\lambda'} \mathcal{P}_{\mathcal{G}_r'^{\perp}}(\boldsymbol{\theta})$, as it is down-weighted by γ_{reg}. Thus, λ_{reg} is the initial strength of the perturbation, normalized to the range $[0, 1]$, and γ_{reg} represents a multiplicative decay of this strength through each update to λ.

We formally write the unlearning update at epoch t as follows, where $\boldsymbol{\theta}_0$ is the current parameter vector, $\boldsymbol{\theta}_{\text{new}}$ is the updated vector, and mod denotes the modulo operation.

if $t \bmod T_{\text{Proj}} \neq 0$ or $t \geq T - T_{\text{GD}}$

$\quad \boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_0$

else

$$\tilde{\boldsymbol{\Delta}} = \operatorname*{argmin}_{\boldsymbol{\Delta} \in \mathcal{G}_r'^{\perp}} \|\boldsymbol{\theta}_0 + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2$$

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_0 + \tilde{\boldsymbol{\Delta}}$$

$$\lambda \leftarrow \frac{\lambda + 1}{\gamma_{\text{reg}}} - 1$$

Gradients for Classification. We make a special note of how we compute the gradient subspace $\mathcal{G}_r'$ for classification tasks. At the parameter value $\boldsymbol{\theta}_0$, the model prediction is $f(\boldsymbol{\theta}_0, \boldsymbol{x}) = \operatorname{argmax} \boldsymbol{z}_{\boldsymbol{\theta}_0}(\boldsymbol{y} \mid \boldsymbol{x})$ where $\boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ denotes the

model's unnormalized logits over the classes $\boldsymbol{y}$ for input $\boldsymbol{x}$ for parameter vector $\boldsymbol{\theta}$. This is not a continuous function of $\boldsymbol{\theta}$, so we cannot compute its gradient directly. However, following prior works (Farajtabar et al., 2020), we use the gradient $\nabla_{\boldsymbol{\theta}} \left(\boldsymbol{z}_{\boldsymbol{\theta}_0}(\boldsymbol{y} \mid \boldsymbol{x})\right)_j$, where $j = f(\boldsymbol{\theta}_0, \boldsymbol{x})$ is the model's predicted class for input $\boldsymbol{x}$. In other words, we take the gradient of the the unnormalized logits at the index of the maximum value, where we do not treat the index as a function of $\boldsymbol{\theta}$.

F.1.8. RIDGE

We again store a regularization weighting λ which we initialize to $\lambda = \lambda_{\text{reg}}$. We define the ridge loss as

$$\mathcal{J}_{\text{Ridge}}(\boldsymbol{\theta}\,; \mathcal{B}_r) + \lambda \left\|\boldsymbol{\theta}\right\|_2^2 .$$

After updating the parameter vector using this loss on each batch, we update λ as

$$\lambda \leftarrow \gamma_{\text{reg}} \lambda.$$

Recall γ_{reg} is always set within the range $[0, 1]$, so the update to λ approximates the limit as λ goes to 0 as we iterate through the epochs. This attempts to recover the minimum norm, or ridgeless, training loss minimizer.

F.2. Multi-Class Label Erasure

We first go over the experimental setup presented in the main body in further detail. We refer to this experiment as Multi-Class Label Erasure, as it tests whether each unlearning algorithm can effectively forget about the non-gray colors. We use the MNIST and CIFAR-10 (LeCun et al., 2010; Krizhevsky, 2009) datasets, creating red, green, and gray copies of each image in the training sets. We construct the retain set as the entire gray copy, and the forget set as a random subset of the red and green copies. Specifically, we construct the forget set as a random sample of 5 percent of the red samples combined with a random sample of the same size of the green samples. We then train a model to predict both the image content class (digit for MNIST, object for CIFAR) as well as the color on the combined data to serve as the initial model for unlearning. For MNIST, we use a CNN with two convolutional layers, one fully connected layer, and then a separate fully connected prediction head for the color and content class. We train for 100 epochs using an initial learning rate of 10^{-3} and a batch size of 3000 along with the AdamW optimizer. For CIFAR-10, we use a modified ResNet-18 (He et al., 2016) architecture also with separate prediction heads for the two class types. In this case, we train for 120 epochs using stochastic gradient descent (SGD) with momentum and weight decay. We set the learning rate to 0.02, momentum to 0.9, and weight decay to 5×10^{-4}, and we use a batch size of 256. We also apply a learning rate scheduler which applies a multiplicative decay of 0.1 every 50 epochs. For each dataset, the ground truth models are trained on the gray images alone using the same training parameters.

We then apply each of the unlearning algorithms over different constraints on the number of unlearning epochs and the amount of available retain data. We define $p_{\text{retain}} \in [0, 1]$ as the proportion of $\mathcal{D}_r$ available during unlearning. For each of the 5 trials, we train a new initial model and sample p_{retain} proportion of $\mathcal{D}_r$ to serve as the available retain data. During each unlearning epoch, the algorithms iterate over batches from the forget set. For every forget set batch, a corresponding batch of the same size is sampled from the available retained data. The epoch ends once all forget set batches have been processed, regardless of whether there are unused retain set samples remaining. Any unused retain batches are not discarded—they will be sampled in subsequent epochs. Once all available retain set batches have been used at least once, the sampling process begins again from the start of the available retain set samples.

The ground truth unlearned model is only trained on gray samples, so it achieves strong accuracy on gray-colored inputs and always predicts the input image to be gray, no matter the input image color. We thus evaluate retain quality by accuracy on gray-colored test samples, and forget quality by the mean squared error between the predicted gray probability and the ideal value of 1 across all colored inputs. For each method, we sweep hyperparameters and plot the Pareto frontier for each method, where the optimal point is at $(1, 0)$ which indicates perfect retain accuracy and zero gray prediction error. Each point in the frontier for a given method represents the median results over 5 trials of a single hyperparameter combination, with the shaded uncertainty shown as half the interquartile range in each direction. We label the performance of the ground truth unlearned model as GT.

We plot the Pareto frontiers and report the hyperparameters used to obtain the optimal curves in the figures and tables below. We do not necessarily sweep over every combination of the reported settings for every algorithm, as we selected

some hyperparameter choices to fill out different areas of the frontier when needed. For example, we often had to set larger learning rates for Scrub to trace a full curve from the upper right to the bottom left. Without doing so, the Scrub results did not reach the bottom half of the plot as the unlearned models remained too close to the initial trained model. Similarly, for some of the CIFAR-10 experiments our algorithm MinNorm-OG needed smaller learning rates and small values of λ_{reg} than usual to sweep through full range through the top right corner, as this area represents models which remain close to the original trained model.

We observe that MinNorm-OG performs the best across all settings. We see that the CIFAR-10 experiments are much more challenging than those on MNIST, as the retain set accuracy degrades much sharper on CIFAR-10 for all unlearning methods. Further, for a small number of allowed unlearning epochs, the performance of MinNorm-OG relative to the other methods can be substantial.

All training and parameter searches were performed on a cluster of NVIDIA GH200 GPUs. For example, sweeping through 150 parameter combinations for Scrub using 8 GPUs at once takes around 15 minutes for 5 unlearning epochs on MNIST.

Table 2. Hyperparameter values tested for the results in Figure 2 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .05$ and $T = 5$.

Hyperparameter	**Sweep Values**
η	$\{10^{-4}, 3\times10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0, 2.0\}$
λ_{reg}	$\{0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1, 0.5, 1.0\}$
T_{GD}	$\{0, 1, 2, 3, 4\}$
γ_{reg}	$\{0.3, 0.6, 0.9\}$
T_{Proj}	$\{1, 2\}$
n_{pert}	$\{20, 40\}$

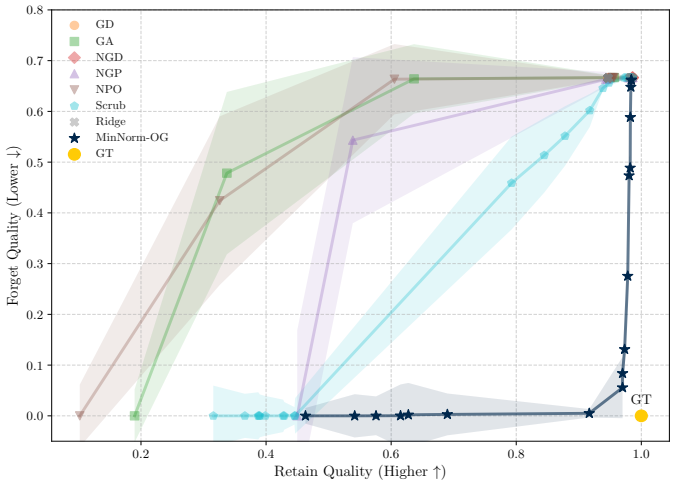


Figure 2. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .05$ and $T = 5$. This is an enlarged version of the top subfigure in Figure 1.

Table 3. Hyperparameter values tested for the results in Figure 3 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .01$ and $T = 5$.

Hyperparameter	**Sweep Values**
η	$\{10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.5, 1.0\}$
T_{GD}	$\{0, 1, 2, 3, 4\}$
γ_{reg}	$\{0.3, 0.6, 0.9\}$
T_{Proj}	$\{1, 2\}$
n_{pert}	$\{20, 40\}$

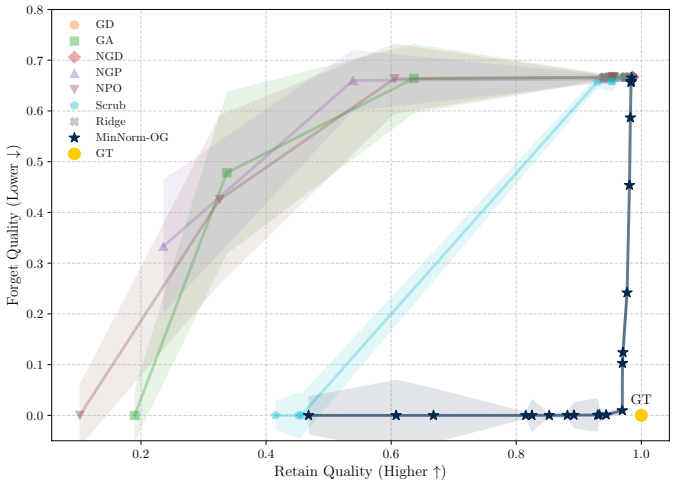


Figure 3. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .01$ and $T = 5$.

Table 4. Hyperparameter values tested for the results in Figure 4 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .05$ and $T = 2$.

Hyperparameter	**Sweep Values**
η	$\{10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1, 0.5\}$
T_{GD}	$\{0, 1\}$
γ_{reg}	$\{0.3, 0.6, 0.9\}$
T_{Proj}	$\{1, 2\}$
n_{pert}	$\{20, 40\}$

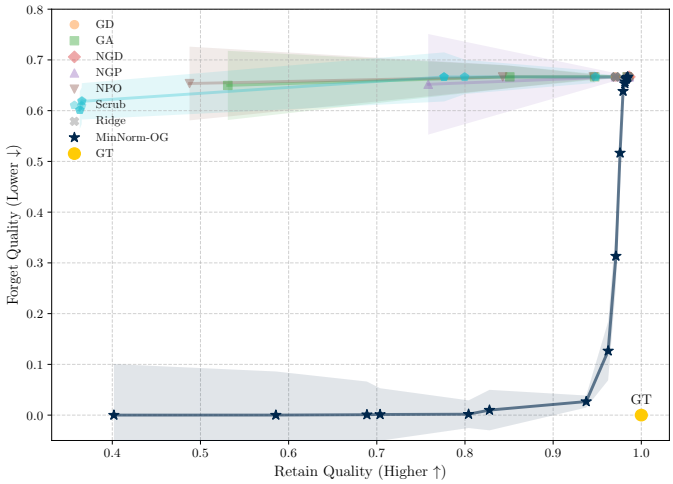


Figure 4. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .05$ and $T = 2$.

Table 5. Hyperparameter values tested for the results in Figure 5 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .01$ and $T = 8$.

Hyperparameter	**Sweep Values**
η	$\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 5 \times 10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1\}$
T_{GD}	$\{0, 1, 2, 3, 4, 5, 6, 7\}$
γ_{reg}	$\{0.3, 0.6, 0.9\}$
T_{Proj}	$\{1, 2\}$
n_{pert}	$\{20, 40\}$

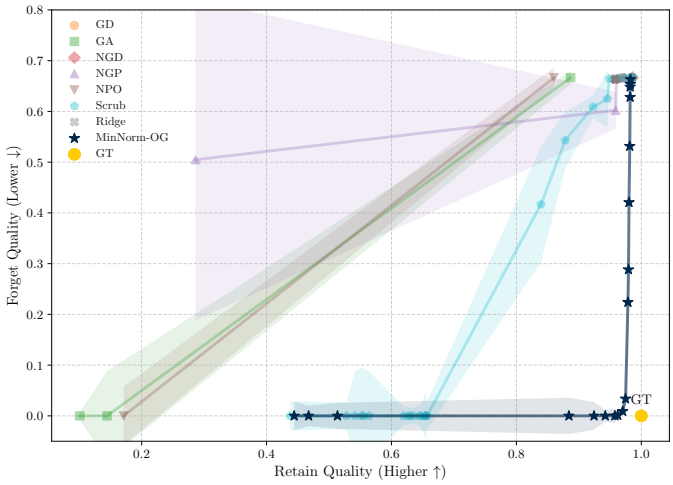


Figure 5. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .01$ and $T = 8$.

Table 6. Hyperparameter values tested for the results in Figure 6 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 5$.

Hyperparameter	**Sweep Values**
η	$\{10^{-7}, 10^{-6}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.0, 0.01, 0.05, 0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1\}$
T_{GD}	$\{0, 1, 2, 4\}$
γ_{reg}	$\{0.3, 0.6, 0.9, 1.0\}$
T_{Proj}	$\{1, 2, 3, 4\}$
n_{pert}	$\{20\}$

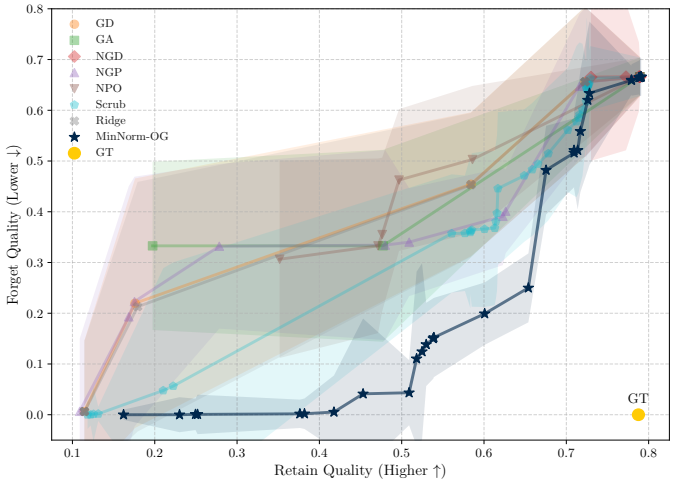


Figure 6. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 5$. This is an enlarged version of the bottom subfigure in Figure 1.

Table 7. Hyperparameter values tested for the results in Figure 7 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 10$.

Hyperparameter	**Sweep Values**
η	$\{10^{-7}, 10^{-6}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.0, 0.01, 0.05, 0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1\}$
T_{GD}	$\{1, 2, 3, 4\}$
γ_{reg}	$\{0.3, 0.6, 0.9, 1.0\}$
T_{Proj}	$\{1, 2, 3, 4\}$
n_{pert}	$\{20\}$

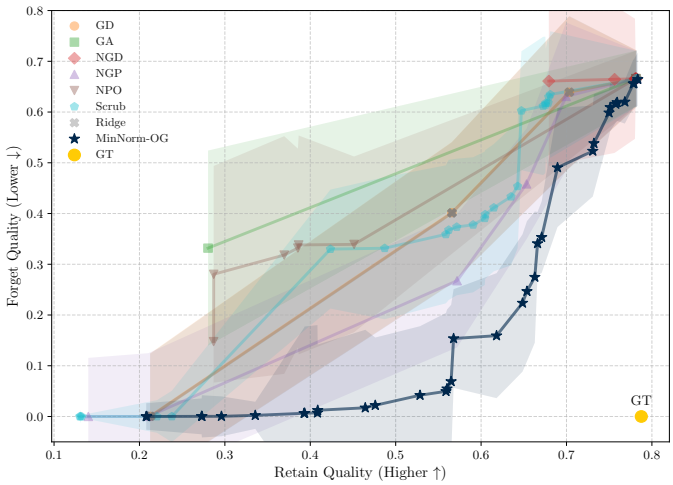


Figure 7. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 10$.

Table 8. Hyperparameter values tested for the results in Figure 8 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 5$.

Hyperparameter	**Sweep Values**
η	$\{10^{-7}, 10^{-6}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.0, 0.01, 0.05, 0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1\}$
T_{GD}	$\{1, 2, 3, 4\}$
γ_{reg}	$\{0.3, 0.6, 0.9, 1.0\}$
T_{Proj}	$\{1, 2, 3, 4\}$
n_{pert}	$\{20\}$

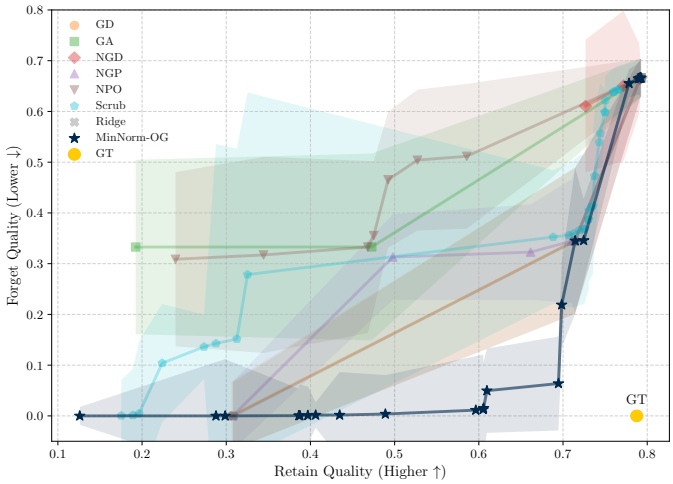


Figure 8. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 5$.

Table 9. Hyperparameter values tested for the results in Figure 9 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 10$.

Hyperparameter	**Sweep Values**
η	$\{10^{-7}, 10^{-6}, 5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}, 10^{-3}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.0, 0.01, 0.05, 0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1, 0.5\}$
T_{GD}	$\{1, 2, 3, 4\}$
γ_{reg}	$\{0.3, 0.6, 0.9, 1.0\}$
T_{Proj}	$\{1, 2, 3, 4\}$
n_{pert}	$\{20\}$

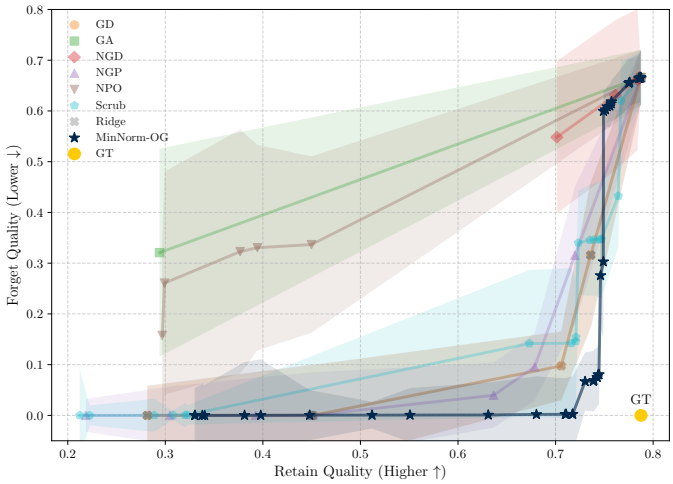


Figure 9. Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 10$.

F.3. Data Poisoning

We include another experiment which tests how well each unlearning algorithm can forget about poisoned samples which degrade model performance on the underlying population trend represented by the retain set samples, which we construct as a simple sine wave.

We train a 3-layer multilayer perceptron with a hidden dimension of 300 using the sigmoid linear unit (SiLU) activation function. For each seed, we randomly sample 50 retain set points $(x_r, y_r) \in \mathcal{D}_r$ with $y_r = \sin(x_r)$ and 5 forget set points $(x_f, y_f) \in \mathcal{D}_f$ with $y_f = 1.5$, over the input domain $\mathcal{X} = [-5\pi, 5\pi] \subseteq \mathbb{R}$. We initially train the poisoned model on all the samples using the AdamW optimizer with a learning rate of 10^{-3} over 100,000 epochs.

Given these poisoned models, we apply each of the unlearning algorithms over a sweep of hyperparameters and evaluate the output $\boldsymbol{\theta}$ of each unlearning method by measuring the deviation from the retain set trend, given by $\sup_{\boldsymbol{x} \in \mathcal{X}} |f(\boldsymbol{\theta}, \boldsymbol{x}) - \sin(\boldsymbol{x})|$. We fix the number of epochs for each algorithm and allow full data access, so each method has access to all of $\mathcal{D}_r$ during unlearning. We repeat the entire process over 20 trials. For the number of unlearning epochs $T \in \{10, 100, 1000\}$, we report the best performance of each algorithm in Table 10 along with the associated hyperparameters in Table 11. We select the parameters for each method by finding the best performing parameters from the possible values in Table 12 using the first 5 trials. We then evaluate over the full 20 trials to obtain our results. We also include visualizations of the recovered models from each unlearning method in Figures 10, 11, and 12. WE observe that the methods which primarily perform loss descent (GD, NGD, Ridge) do not effectively move away from the initial poisoned model, as the initial solution is nearly optimal over all data subsets. Further, methods which combine loss ascent (GA, NGP) deviate from the population trend as the loss ascent objective provides a coarse unlearning signal that interferes with fitting the retained samples. All experiments were run on either a single NVIDIA A40 GPU or a single NVIDIA GH200 GPU.

Table 10. Data Poisoning experiment results, measured as the sup-norm distance between the retain set trend $y = \sin(x)$ and the outputs of the unlearning algorithms (smaller is better). We report medians over 20 trials, along with the range of the central 10 values

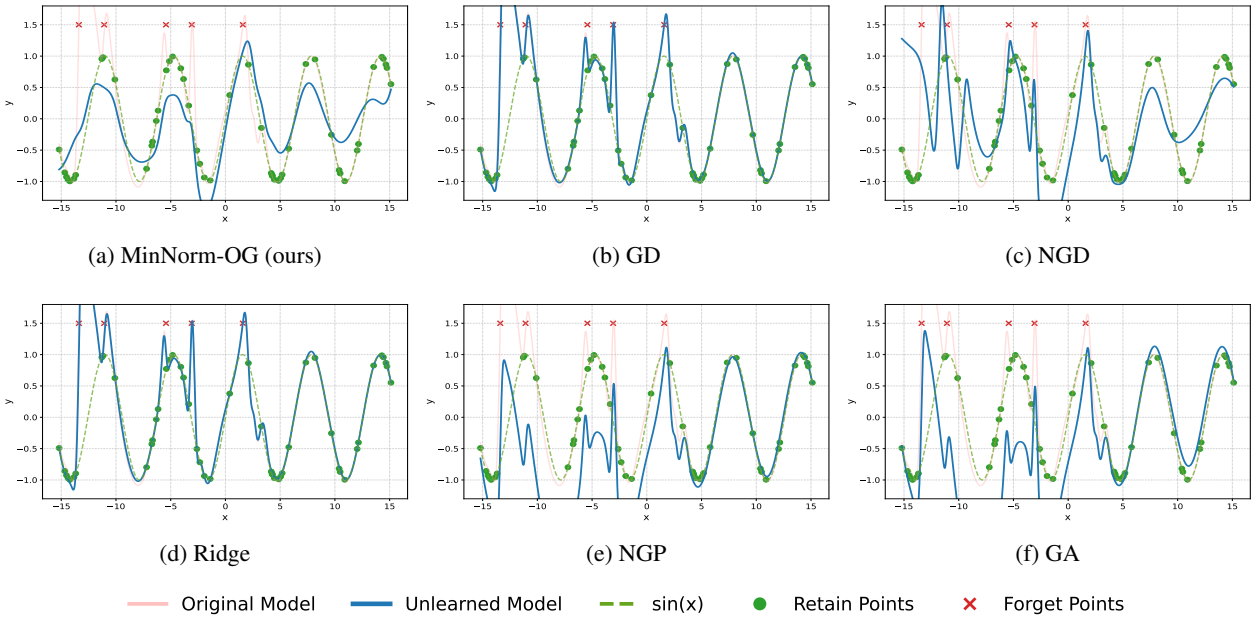
Epochs	GA	GD	NGD	NGP	MinNorm-OG	Ridge
10	3.56 (2.34, 6.52)	3.38 (2.62, 7.48)	3.63 (2.71, 7.56)	3.70 (2.28, 7.37)	**1.89** (1.10, 6.02)	3.38 (2.62, 7.48)
100	27.7 (20.6, 36.2)	1.85 (1.51, 2.76)	2.54 (1.56, 6.09)	1.81 (1.41, 2.93)	**1.07** (0.62, 1.32)	1.67 (1.37, 3.31)
1000	1700 (1200, 2600)	1.58 (1.04, 2.43)	1.35 (.93, 3.47)	2.29 (1.54, 5.07)	**0.84** (0.64, 1.24)	1.29 (0.87, 2.12)

Table 11. Hyperparameter settings for each entry in Table 10. Blank entries indicate that the hyperparameter is not applicable to the corresponding method.

Epochs	Method	η	λ_{GA}	λ_{reg}	σ	T_{GD}	γ_{reg}	T_{Proj}	n_{pert}
10	GA	1e-4							
	GD	1e-4							
	NGD	1e-2			.5				
	NGP	1e-4	1.0						
	MinNorm-OG	1e-3		.3		0	.3	1	50
	Ridge	1e-4		1.0			.3		
100	GA	1e-4							
	GD	1e-2							
	NGD	1e-2			1.0				
	NGP	1e-2	1e-3						
	MinNorm-OG	1e-3		0.1		50	0.9	1	50
	Ridge	1e-2		3.0			0.6		
1000	GA	1e-4							
	GD	1e-2							
	NGD	1e-2			0.1				
	NGP	1e-2	1e-3						
	MinNorm-OG	1e-2		0.3		0	0.3	200	50
	Ridge	1e-2		3.0			1.0		

Table 12. Hyperparameter values tested in the experiments corresponding to Table 10. We denote the total number of epochs T.

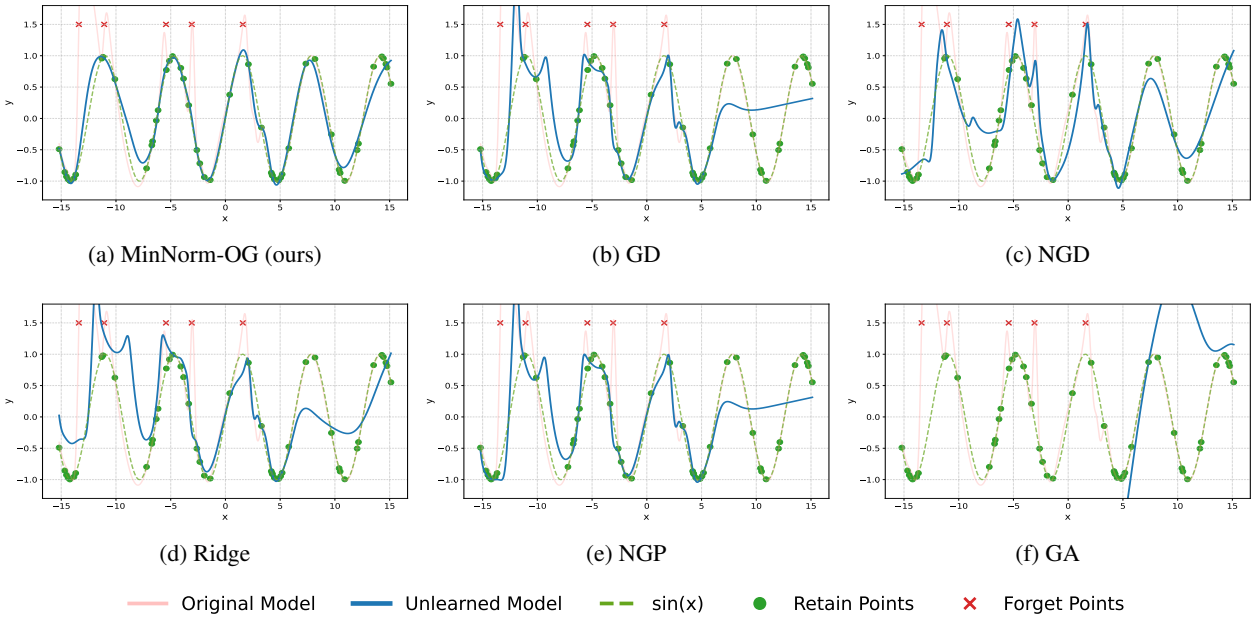
Hyperparameter	Sweep Values
η	$\{10^{-4}, 10^{-3}, 10^{-2}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}, 1.0\}$
λ_{reg}	$\{0.1, 0.3, 0.5, 1.0, 3.0\}$
σ	$\{0.1, 0.5, 1.0\}$
T_{GD}	$\{0, T/10, T/2\}$
γ_{reg}	$\{0.3, 0.6, 0.9, 1.0\}$
T_{Proj}	$\{1, T/10, T/5\}$
n_{pert}	$\{50\}$



(a) MinNorm-OG (ours) (b) GD (c) NGD

(d) Ridge (e) NGP (f) GA

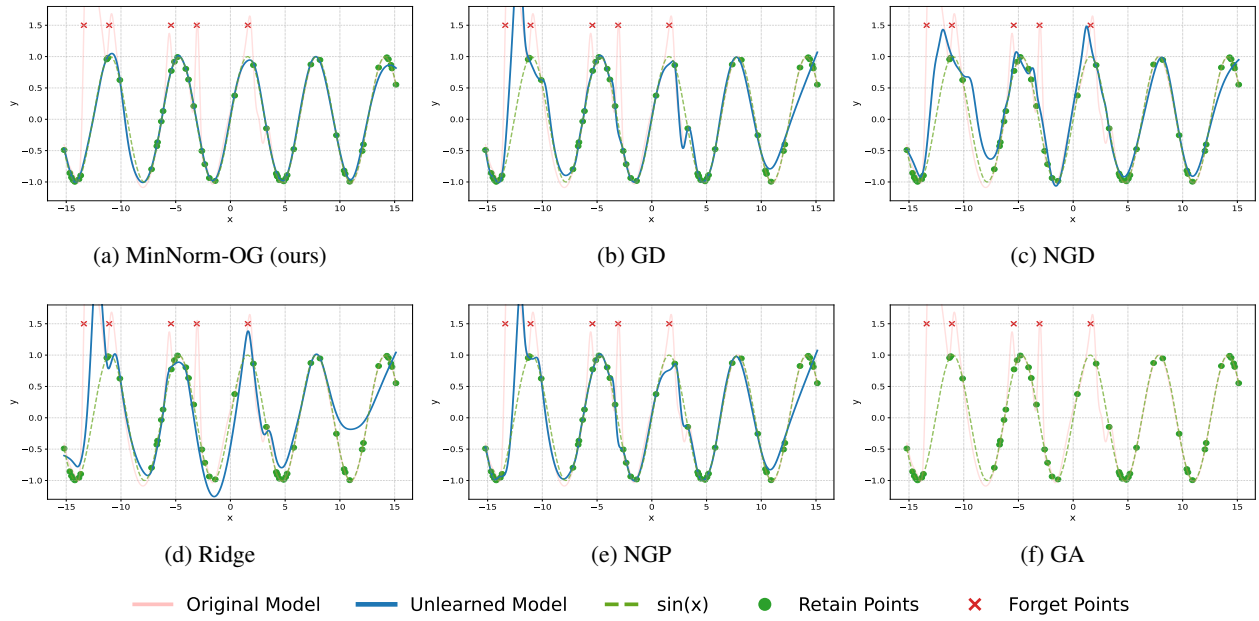
Figure 10. Example unlearned model fits when given 10 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.



(a) MinNorm-OG (ours) (b) GD (c) NGD

(d) Ridge (e) NGP (f) GA

Figure 11. Example unlearned model fits when given 100 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.



(a) MinNorm-OG (ours) (b) GD (c) NGD

(d) Ridge (e) NGP (f) GA

Figure 12. Example unlearned model fits when given 1000 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

F.4. Representation Collapse

We lastly include an experiment which tests if each unlearning method can effectively forget complex data representations which would only be learned through training on the full dataset.

We use a subset of MNIST where the retain set contains the images with digit 0 colored green and the images with digit 1 colored red. We then construct the forget set by randomly sampling 10% of the 0 and 1 digits and coloring them oppositely to the retain set coloring, so the forget set 0's are colored red and the forget set 1's are colored green. We train the initial model over 250 epochs with a learning rate of 10^{-3} and the AdamW optimizer. For MNIST, we use the same convolutional neural network architecture as in the Multi-Class Label Erasure experiment, except with a single prediction head, and we use a batch size of 3000. For CIFAR-10, we similarly use a modified ResNet-18 architecture along with a batch size of 2048. We also train ground truth unlearned models using the same settings, except we only train for 100 epochs instead of 250.

The ground truth unlearned model predicts from color alone, as color perfectly determines the label in $\mathcal{D}_r$ and is easier to learn than digit shape. In contrast, models trained on the full dataset $\mathcal{D} = \mathcal{D}_r \sqcup \mathcal{D}_f$ must rely on shape, since color is no longer predictive. For evaluation, we relabel training images by color and assess unlearning via color-label accuracy, testing if the unlearning methods can collapse the original model into just a color classifier.

We apply each unlearning algorithm for a set number of unlearning epochs T as well as a fixed proportion of the retain set which is accessible, which we denote $p_{\text{retain}} \in [0, 1]$. Just as in the Multi-Class Label Erasure experiment, during each unlearning epoch the algorithms iterate over batches from the forget set and sample a corresponding batch of the same size from the available retained data. The epoch ends once all forget set batches have been processed, regardless of whether there are unused retain set samples remaining. Any unused retain batches are not discarded—they will be sampled in subsequent epochs. Once all available retain set batches have been used at least once, the sampling process begins again from the start of the available retain set samples.

We search over hyperparameters and report the best results for each algorithm in each setting in Table 13. We write Retain % to denote $100 \times p_{\text{retain}}$. We observed that the results can exhibit a bimodal distribution across trials, as each method must transition from an initial model that classifies digits perfectly to one that achieves the same retain set accuracy using only color. When this transition fails, the model often reverts to digit-based predictions, leading to high variance in the results. To reflect this behavior robustly, Table 13 reports median color accuracy over 5 trials, along with the range of the central 3 values. We note that MinNorm-OG consistently performs best. For each setting of the number of epochs and the Retain %,

Table 13. Unlearning performance across constraints on the number of epochs and percentage of accessible retain set samples for the Representation Collapse experiment. Evaluation is measured as accuracy on duplicate training images labeled by color only (higher is better). We report medians over 5 trials, along with the range of the central 3 values.

Retain %	**Epochs**	GD	GA	NGD	NGP	NPO	Scrub	MinNorm-OG	Ridge
1	5	0.60 (0.52, 0.70)	0.50 (0.50, 0.50)	0.50 (0.50, 0.50)	0.90 (0.77, 0.97)	0.50 (0.50, 0.50)	0.80 (0.74, 0.85)	**1.00** (1.00, 1.00)	0.73 (0.53, 0.73)
	8	0.72 (0.53, 0.74)	0.50 (0.50, 0.50)	0.50 (0.50, 0.50)	**1.00** (0.99, 1.00)	0.50 (0.50, 0.50)	0.96 (0.79, 0.97)	**1.00** (1.00, 1.00)	0.73 (0.66, 0.73)
	10	0.76 (0.73, 0.79)	0.50 (0.50, 0.50)	0.50 (0.50, 0.50)	**1.00** (1.00, 1.00)	0.50 (0.50, 0.50)	**1.00** (1.00, 1.00)	**1.00** (1.00, 1.00)	0.75 (0.73, 0.82)
10	5	0.73 (0.52, 0.73)	0.50 (0.50, 0.58)	0.50 (0.50, 0.50)	0.91 (0.82, 0.92)	0.52 (0.50, 0.57)	0.76 (0.73, 0.83)	**1.00** (0.85, 1.00)	0.73 (0.52, 0.73)
	8	0.72 (0.65, 0.74)	0.50 (0.50, 0.50)	0.50 (0.50, 0.50)	**1.00** (1.00, 1.00)	0.50 (0.50, 0.50)	**1.00** (0.99, 1.00)	**1.00** (1.00, 1.00)	0.77 (0.70, 0.81)
	10	0.73 (0.69, 0.80)	0.50 (0.50, 0.50)	0.50 (0.50, 0.50)	**1.00** (1.00, 1.00)	0.50 (0.50, 0.50)	**1.00** (1.00, 1.00)	**1.00** (1.00, 1.00)	0.92 (0.81, 0.92)

we show the hyperparameters we considered in Tables 14,15,16,17,18,and 19 before reporting the best performance out of each combination for each algorithm.

All training was performed on a cluster of NVIDIA GH200 GPUs. For example, sweeping through all hyperparameter combinations listed in Table 14 for each algorithm completed in about 10 minutes using 8 nodes.

Table 14. Hyperparameter values considered for the Representation Collapse Experiment with $T = 5$ and $p_{\text{retain}} = 0.01$.

Hyperparameter	**Values**
η	$\{10^{-2},\ 8 \times 10^{-3},\ 3 \times 10^{-3}\}$
λ_{GA}	$\{10^{-3},\ 10^{-2},\ 0.1,\ 1.0\}$
λ_{reg}	$\{0.1,\ 0.3,\ 0.5,\ 1.0,\ 3.0\}$
σ	$\{0.1,\ 0.5,\ 1.0\}$
T_{GD}	$\{1,\ 2\}$
γ_{reg}	$\{0.3,\ 0.5,\ 0.9,\ 1.0\}$
T_{Proj}	$\{1,\ 2\}$
n_{pert}	$\{50\}$

Table 15. Hyperparameter values considered for the Representation Collapse Experiment with $T = 5$ and $p_{\text{retain}} = 0.1$.

Hyperparameter	**Values**
η	$\{9 \times 10^{-3},\ 7 \times 10^{-3},\ 3 \times 10^{-3}\}$
λ_{GA}	$\{10^{-3},\ 10^{-2},\ 0.1,\ 1.0\}$
λ_{reg}	$\{0.1,\ 0.3,\ 0.6,\ 1.0,\ 3.0\}$
σ	$\{0.1,\ 0.5,\ 1.0\}$
T_{GD}	$\{1,\ 2\}$
γ_{reg}	$\{0.3,\ 0.5,\ 0.9,\ 1.0\}$
T_{Proj}	$\{1,\ 2\}$
n_{pert}	$\{50\}$

Table 16. Hyperparameter values considered for the Representation Collapse Experiment with $T = 8$ and $p_{\text{retain}} = 0.01$.

Hyperparameter	Values
η	$\{8 \times 10^{-3},\ 3 \times 10^{-3},\ 8 \times 10^{-4}\}$
λ_{GA}	$\{10^{-3},\ 10^{-2},\ 0.1,\ 1.0\}$
λ_{reg}	$\{0.1,\ 0.3,\ 0.6,\ 1.0,\ 3.0\}$
σ	$\{0.1,\ 0.5,\ 1.0\}$
T_{GD}	$\{4,\ 6\}$
γ_{reg}	$\{0.3,\ 0.5,\ 0.9,\ 1.0\}$
T_{Proj}	$\{1,\ 2\}$
n_{pert}	$\{50\}$

Table 17. Hyperparameter values considered for the Representation Collapse Experiment with $T = 8$ and $p_{\text{retain}} = 0.1$.

Hyperparameter	Values
η	$\{8 \times 10^{-3},\ 3 \times 10^{-3},\ 8 \times 10^{-4}\}$
λ_{GA}	$\{10^{-3},\ 10^{-2},\ 0.1,\ 1.0\}$
λ_{reg}	$\{0.1,\ 0.3,\ 0.6,\ 1.0,\ 3.0\}$
σ	$\{0.1,\ 0.5,\ 1.0\}$
T_{GD}	$\{4,\ 6\}$
γ_{reg}	$\{0.3,\ 0.5,\ 0.9,\ 1.0\}$
T_{Proj}	$\{1,\ 2\}$
n_{pert}	$\{50\}$

Table 18. Hyperparameter values considered for the Representation Collapse Experiment with $T = 10$ and $p_{\text{retain}} = 0.01$.

Hyperparameter	Values
η	$\{8 \times 10^{-3},\ 3 \times 10^{-3},\ 8 \times 10^{-4}\}$
λ_{GA}	$\{10^{-3},\ 10^{-2},\ 0.1,\ 1.0\}$
λ_{reg}	$\{0.1,\ 0.3,\ 0.6,\ 1.0,\ 3.0\}$
σ	$\{0.1,\ 0.5,\ 1.0\}$
T_{GD}	$\{4,\ 7\}$
γ_{reg}	$\{0.3,\ 0.5,\ 0.9,\ 1.0\}$
T_{Proj}	$\{1,\ 2\}$
n_{pert}	$\{50\}$

Table 19. Hyperparameter values considered for the Representation Collapse Experiment with $T = 10$ and $p_{\text{retain}} = .1$.

Hyperparameter	Values
η	$\{8 \times 10^{-3},\ 3 \times 10^{-3},\ 8 \times 10^{-4}\}$
λ_{GA}	$\{10^{-3},\ 10^{-2},\ 0.1,\ 1.0\}$
λ_{reg}	$\{0.1,\ 0.3,\ 0.6,\ 1.0,\ 3.0\}$
σ	$\{0.1,\ 0.5,\ 1.0\}$
T_{GD}	$\{4,\ 7\}$
γ_{reg}	$\{0.3,\ 0.5,\ 0.9,\ 1.0\}$
T_{Proj}	$\{1,\ 2\}$
n_{pert}	$\{50\}$

F.5. Asset Information

We use the MNIST (LeCun et al., 2010) and CIFAR-10 (Krizhevsky, 2009) datasets in our experiments. CIFAR-10 is publicly available but does not specify an explicit license. MNIST is also publicly available and is typically distributed under the Creative Commons Attribution-ShareAlike 3.0 License. Additionally, we use the ResNet-18 (He et al., 2016) architecture and pretrained weights from PyTorch's `torchvision` library, which are licensed under the BSD 3-Clause License.