
Machine Unlearning under Overparameterization

Jacob L. Block
UT Austin
jblock@utexas.edu

Aryan Mokhtari
UT Austin & Google Research
mokhtari@austin.utexas.edu

Sanjay Shakkottai
UT Austin
sanjay.shakkottai@utexas.edu

Abstract

Machine unlearning algorithms aim to remove the influence of specific training samples, ideally recovering the model that would have resulted from training on the remaining data alone. We study unlearning in the overparameterized setting, where many models interpolate the data, and defining the solution as any loss minimizer over the retained set—as in prior work in the underparameterized setting—is inadequate, since the original model may already interpolate the retained data and satisfy this condition. In this regime, loss gradients vanish, rendering prior methods based on gradient perturbations ineffective, motivating both new unlearning definitions and algorithms. For this setting, we define the unlearning solution as the minimum-complexity interpolator over the retained data and propose a new algorithmic framework that only requires access to model gradients on the retained set at the original solution. We minimize a regularized objective over perturbations constrained to be orthogonal to these model gradients, a first-order relaxation of the interpolation condition. For different model classes, we provide exact and approximate unlearning guarantees and demonstrate that an implementation of our framework outperforms existing baselines across various unlearning experiments.

1 Introduction

The ability to remove the influence of specific samples from a trained model is essential to comply with privacy regulations such as the GDPR and CCPA [1, 2] and to correct mislabeling or bias that may compromise model integrity [3]. *Machine unlearning* [4] refers to algorithms that address these challenges by modifying a model trained on a dataset $\mathcal{D}$ to forget a subset of samples, termed the forget set $\mathcal{D}_f$, and produce a model that acts as if it had been trained only on the remaining data, denoted the retain set $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$. The ideal, yet costly, “gold standard” unlearning solution is to retrain the model from scratch on $\mathcal{D}_r$, which perfectly achieves the unlearning objective but is often infeasible due to high computational cost and potentially limited access to the original training data. The goal of an unlearning method is to efficiently approximate this outcome using the knowledge of the original training procedure, the samples to be forgotten, and potentially restricted side-information related to the retained data, aiming to recover a model that could result from training on $\mathcal{D}_r$ alone.

In the *underparameterized* regime, where the model class cannot fit all training data, the training loss admits a unique minimizer. Thus, the natural definition of exact unlearning is the unique minimizer to the loss on $\mathcal{D}_r$. When the loss is further strongly convex, prior work developed efficient unlearning approximations using influence functions, which estimate the effect of removing a sample via a single gradient ascent step over the loss on $\mathcal{D}_f$, preconditioned by the inverse loss Hessian on $\mathcal{D}$ [5–7].

In contrast, this paper focuses on the *overparameterized* regime, where the model class contains many interpolating solutions. Crucially, the training loss no longer admits a unique minimizer, and defining the unlearning solution by loss optimality alone no longer suffices: the original model θ^* minimizes the loss over both $\mathcal{D}$ and $\mathcal{D}_r$, and θ^* clearly encodes information about $\mathcal{D}_f$, the data to be removed. Moreover, interpolation causes the loss gradients to vanish, rendering loss-gradient-based methods such as influence functions ineffective (Theorem 1). This fundamental shift necessitates both a new definition of unlearning and new algorithmic tools tailored to the overparameterized setting.

We begin by formalizing unlearning in the overparameterized setting. Specifically, we define the exact unlearning solution as the model which minimizes a model complexity measure R , subject to minimizing the loss over $\mathcal{D}_r$; see (2). For natural choices of R , such as the parameter norm, this definition ensures that the unlearned model reveals no information about the forgotten data and maintains strong generalization using only the retain set. Given this definition of unlearning, we propose a new algorithmic framework to compute the solution. We focus on settings where the loss is minimized by any interpolating model, so the constraint reduces to requiring interpolation over $\mathcal{D}_r$. To solve the resulting problem of minimizing R subject to interpolation, we relax the constraint via a first-order Taylor expansion around θ^* and reparameterize as $\theta^* + \Delta$, where Δ is the drift. Since θ^* already interpolates $\mathcal{D}_r$, the linearized constraint requires Δ to be orthogonal to model gradients at θ^* on $\mathcal{D}_r$. This simplifies the problem, requiring only gradient access, and avoids the complex interpolation constraint. To mitigate error from this relaxation, we add a regularizer $\hat{R}(\Delta)$ to control the size and direction of the drift. The final objective minimizes $R(\theta^* + \Delta) + \hat{R}(\Delta)$ under the relaxed orthogonal gradient constraint, yielding updated parameters $\theta^* + \Delta$.

Theoretical Contributions. We prove for linear models and linear networks, there exists a regularizer $\hat{R}$ such that minimizing $R(\theta^* + \Delta) + \hat{R}(\Delta)$ over our constraint relaxation gives the exact unlearning solution when R is the ℓ_2 -norm of either the effective regressor or the full parameter vector. For two-layer perceptrons with nonlinear activations, where R measures network width, we prove that the right choice of $\hat{R}$ yields a solution to our relaxed problem which interpolates $\mathcal{D}_r$ and matches the best known upper bound on the number of neurons required to fit any dataset of a given size.

Algorithmic Contributions. We devise an iterative algorithm MinNorm-OG that accesses a subset of $\mathcal{D}_r$, aligning with data access assumptions in prior work [8–10], where OG refers to orthogonal gradient. MinNorm-OG alternates between two steps: solving for the minimizer of $R(\theta + \Delta) + \hat{R}(\Delta)$ over Δ satisfying the orthogonal gradient constraint, and descending on the loss over $\mathcal{D}_r$ (Algorithm 1). We take both R and $\hat{R}$ as scaled squared ℓ_2 norms, which apply broadly to parameterized models and yield a closed-form solution to the relaxed problem. We show strong performance of our method across three experimental settings: *Data Poisoning*, *Multi-Class Label Erasure*, and *Representation Collapse*, using natural and interpretable unlearning metrics to compare our method against existing baselines. Notably, the Multi-Class Label Erasure and Representation Collapse image-domain experiments introduce novel unlearning settings for effective evaluation.

Related work. Unlearning theory traces back to influence functions [11] which estimate the effect of down-weighting a sample on a learned function [5]. Extensions explored approximate unlearning via differential privacy [6, 7]. [6] analyzed the deletion capacity an unlearning method can tolerate while maintaining generalization. [12, 13] proposed joint learning-unlearning schemes that store information about data subsets during training for later unlearning. Several works proposed iterative unlearning methods for large-scale models, combining loss ascent, descent, and noise injection [14–18]. All these methods rely on loss gradient perturbations, which we show yield vacuous updates under overparameterization (Theorem 1). In practice, they also struggle to unlearn effectively [9], as loss ascent encourages misfitting $\mathcal{D}_f$ rather than forgetting it. Our framework enforces parameter perturbations to be orthogonal to the gradient over $\mathcal{D}_r$ to preserve loss optimality, an idea also used in continual learning contexts [19]. Recent unlearning methods use similar projections which mix loss ascent and descent, but their reliance on these objectives inherits prior limitations [20, 21].

2 Unlearning in Overparameterized Settings

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

To formalize the unlearning problem, we now define the problem setting and notation, covering both the underparameterized and overparameterized regimes. We define the full training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, with sample inputs $x_i \in \mathbb{R}^m$ and outputs $y_i \in \mathbb{R}^l$ drawn from the data domain $\mathcal{Z} = \mathbb{R}^m \times \mathbb{R}^l$. Initially, training is performed on the full dataset $\mathcal{D}$ over the model class $\{f(\theta, \cdot) \mid \theta \in \mathbb{R}^d\}$ parameterized by $\theta \in \mathbb{R}^d$, where $f : \mathbb{R}^{d+m} \rightarrow \mathbb{R}^l$ takes a parameter vector $\theta \in \mathbb{R}^d$ and an input $x \in \mathbb{R}^m$ and maps them to a prediction $f(\theta, x)$ in the output space $\mathbb{R}^l$. We define the training procedure, also denoted the learning algorithm, as $\mathcal{A} : 2^{\mathcal{Z}} \rightarrow \mathbb{R}^d$, which takes in a dataset and returns the parameter vector θ^* corresponding to the trained model. We make the minimal assumption that $\mathcal{A}$ is faithful to a known loss function $\mathcal{J}$, meaning $\mathcal{A}(\mathcal{D}) = \theta^*$ is only guaranteed to be a minimizer of $\mathcal{J}$ over $\mathcal{D}$, where $\mathcal{J}$ is defined as the average of the sample-wise loss $\mathcal{L}$:

$$\mathcal{A}(\mathcal{D}) = \theta^* \in \operatorname{argmin}_{\theta} \mathcal{J}(\theta; \mathcal{D}) = \operatorname{argmin}_{\theta} \frac{1}{n} \sum_{(x,y) \in \mathcal{D}} \mathcal{L}(\theta; x, y). \quad (1)$$

For our theoretical discussion, we consider sample-wise loss functions $\mathcal{L}(\theta; x, y)$ which are minimized when $f(\theta, x) = y$, meaning that sample interpolation implies loss minimization. For example, this is the case for ℓ_p -norm regression or classification with 0-1 loss.

With this training setup, we begin the unlearning process given a request for the model to forget a subset of the training data $\mathcal{D}_f \subseteq \mathcal{D}$. We then 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 the samples, or the Hessian of the training loss over the retained data), initial solution $\mathcal{A}(\mathcal{D})$, and forget set $\mathcal{D}_f$, and which attempts to recover the desired unlearning solution, denoted by θ_r^* , where the subscript r indicates that θ_r^* is the parameter vector that would result from training only on the retain set $\mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f$. To formally define θ_r^* , we must distinguish between underparameterized and overparameterized regimes, as the former's definition requires refinement to remain meaningful in the latter.

In the *underparameterized* setting, the loss function over both the full data set $\mathcal{J}(\theta; \mathcal{D})$ as well as the retain set $\mathcal{J}(\theta; \mathcal{D}_r)$ admits a unique minimizer. To ensure that the unlearning solution remains consistent with the training loss, the only valid choice is to define θ_r^* as the unique minimizer of $\mathcal{J}(\theta; \mathcal{D}_r)$. However, in the *overparameterized* setting this uniqueness property fails to hold, as both $\mathcal{J}(\theta; \mathcal{D})$ and $\mathcal{J}(\theta; \mathcal{D}_r)$ may admit multiple minimizers. In order to sidestep the non-uniqueness issue, one may be tempted to define *any* minimizer of $\mathcal{J}(\theta; \mathcal{D}_r)$ as a valid unlearning solution, as presumably any minimizer to $\mathcal{J}(\theta; \mathcal{D}_r)$ could be found from just training on $\mathcal{D}_r$ alone. However, following this rationale allows for seemingly valid unlearning solutions to leak information relating to $\mathcal{D}_f$. Specifically, the original solution θ^* that interpolates all of $\mathcal{D}$ is itself a valid minimizer of the retain set loss $\mathcal{J}(\theta; \mathcal{D}_r)$, but θ^* can reflect training dynamics influenced by $\mathcal{D}_f$, revealing information that cannot be inferred from $\mathcal{D}_r$ alone (see Appendix C for a concrete illustration).

2.1 Defining Unlearning Beyond Loss Optimality

As discussed above, the overparameterized setting requires a more fine-grained definition of the desired unlearning solution—one that goes beyond loss optimality. We define the unlearning solution in the overparameterized case to be the specific loss minimizer which minimizes an additional objective function $R(\theta)$, expressed as the output of a training algorithm $\mathcal{A}_R$:

$$\mathcal{A}_R(\mathcal{D}_r) = \theta_r^* \in \operatorname{argmin}_{\theta} R(\theta), \quad \text{subject to } \theta \in \operatorname{argmin}_{\theta'} \mathcal{J}(\theta'; \mathcal{D}_r). \quad (2)$$

This bilevel optimization problem searches for the model which minimizes the complexity measure R among all models which minimize the retain set loss. Indeed, 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. While different choices of R can address these issues, we ultimately want R to promote desirable model properties. In our theoretical results, we focus on R as a regularization function that penalizes model complexity. This way, the solution θ_r^* to (2) corresponds to the simplest model that interpolates $\mathcal{D}_r$ —a particularly useful property in the overparameterized regime, where the simplest interpolating model is often associated with optimal generalization performance [22].

Then given the training algorithm $\mathcal{A}_R$, side information about the retain set $\mathcal{I}_r$, a minimizer to the original training loss $\mathcal{A}(\mathcal{D})$, and the forget set $\mathcal{D}_f$, an unlearning algorithm $M(\mathcal{A}_R, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f)$ attempts to recover $\mathcal{A}_R(\mathcal{D}_r)$, the least complex loss minimizer over $\mathcal{D}_r$ as measured by R .

2.2 Loss Gradient Methods Deployed Under Overparameterization

For the characterization in (2) of the ground truth unlearning solution under overparameterization, we show that existing unlearning methods based on loss gradient perturbations fail to achieve meaningful unlearning updates. Prior theoretical works proposed gradient-ascent-style updates based on influence functions, a principled technique from robust statistics [5–7], while existing empirical unlearning methods perform combinations of loss ascent over $\mathcal{D}_f$, loss descent over $\mathcal{D}_r$, and parameter noising [14–16, 8]. We characterize these methods as *loss-gradient unlearning*, and show that they perform ineffective updates when deployed under overparameterization.

Definition 1. Let $\theta^* = \mathcal{A}(\mathcal{D})$. We say an unlearning algorithm M performs *loss-gradient unlearning* if for any positive semi-definite $\mathbf{P}_r, \mathbf{P}_f \in \mathbb{R}^{d \times d}$ and zero-mean random variable $\xi \in \mathbb{R}^d$,

$$M(\mathcal{A}, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f) = \theta^* - \mathbf{P}_r \nabla_{\theta} \mathcal{J}(\theta^*; \mathcal{D}_r) + \mathbf{P}_f \nabla_{\theta} \mathcal{J}(\theta^*; \mathcal{D}_f) + \xi \quad (3)$$

Although versions of loss-gradient unlearning have been theoretically motivated in the underparameterized setting [6, 7], we show they fail to unlearn in the overparameterized setting.

Theorem 1. Let $f(\theta^*, \cdot)$ interpolate $\mathcal{D}$, so $f(\theta^*, \mathbf{x}) = \mathbf{y}$ for all $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$, and let M_{LG} be any loss-gradient unlearning method. If the sample loss $\mathcal{L}(\theta, \mathbf{x}, \mathbf{y})$ is minimized when $f(\theta, \mathbf{x}) = \mathbf{y}$, then for all $\mathcal{D}_f \subseteq \mathcal{D}$, M_{LG} simply noises θ^* by some zero-mean random variable ξ .

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

The recovered parameters θ^* already minimize $\mathcal{J}(\theta^*; \mathcal{D}_r)$, so the loss gradients vanish and M_{LG} merely adds noise to θ^* . This shows the core issue with loss gradient updates in overparameterized unlearning: the loss gradient is uninformative, as both θ^* and θ_r^* minimize the loss on $\mathcal{D}_r$.

3 Our Proposed Framework

We present a new framework to efficiently address the desired unlearning goal in overparameterized settings without full retraining. A key assumption underlying our method is the richness of the function class, allowing for perfect fitting of the retain set. This means there exist several mappings $f(\theta, \cdot)$ where $f(\theta, \mathbf{x}_i) = \mathbf{y}_i$ for every $(\mathbf{x}_i, \mathbf{y}_i)$ in the retain set. This lets us replace the loss minimization in (2) with the hard constraint $f(\theta, \mathbf{x}_i) = \mathbf{y}_i$, leading to the following formulation:

$$\theta_r^* \in \underset{\theta}{\operatorname{argmin}} R(\theta) \quad \text{s.t.} \quad f(\theta, \mathbf{x}_i) = \mathbf{y}_i \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_r \quad (4)$$

This problem can be independently solved, but this would be the equivalent of retraining on the retain set. The main goal of our proposed framework is to solve the above problem efficiently by starting from the model θ^* which fits each sample and leveraging the feasibility of this model for the above optimization problem. To do so, we simplify the problem and replace the constraints in (4) with their linear approximation around θ^* . While the constraints $f(\theta, \mathbf{x}_i) = \mathbf{y}_i$ in (4) can be highly nonconvex and difficult to satisfy in general, we demonstrate that using the proposed first-order approximation

$$f(\theta^*, \mathbf{x}_i) + \nabla f(\theta^*, \mathbf{x}_i)^\top (\theta - \theta^*) = \mathbf{y}_i \quad \Rightarrow \quad \nabla f(\theta^*, \mathbf{x}_i)^\top (\theta - \theta^*) = 0, \quad (5)$$

renders it tractable as it leads to a set of linear constraints with respect to θ . Note that in the above simplification we used the fact that θ^* perfectly fits the retain set, so $f(\theta^*, \mathbf{x}_i) = \mathbf{y}_i$. Now if we apply this constraint relaxation the resulting optimization problem would be:

$$\min_{\Delta} R(\theta^* + \Delta) \quad \text{s.t.} \quad \nabla f(\theta^*, \mathbf{x}_i)^\top \Delta = 0 \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_r, \quad (6)$$

where for notational convenience, we define the drift variable as $\Delta = \theta - \theta^*$. While this relaxation is sensible, it presents a clear limitation: approximating a general function with its linearization is only locally accurate and thus valid when the drift term Δ remains sufficiently small in some norm. To keep the surrogate solution close to that of the original problem in (4), we add a regularization term $\hat{R}(\Delta)$ to the loss to control the drift. The resulting objective function is $\tilde{R}(\theta^* + \Delta) := R(\theta^* + \Delta) + \hat{R}(\Delta)$. Consequently, the optimization problem we propose to solve instead of (4) is given by

$$\tilde{\Delta} \in \underset{\Delta}{\operatorname{argmin}} \tilde{R}(\theta^* + \Delta) \quad \text{s.t.} \quad \nabla f(\theta^*, \mathbf{x}_i)^\top \Delta = 0 \quad \forall (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_r \quad (7)$$

Indeed, by finding $\tilde{\Delta}$ the suggested unlearned model would be $\theta^* + \tilde{\Delta}$. Although (7) employs relaxed constraints, we will show that for various mapping functions f , there exists a function $\hat{R}$ such that the solution to (7) either (i) solves the original unlearning problem (4) exactly, or (ii) yields a model that both interpolates $\mathcal{D}_r$, remaining feasible for (4), and satisfies a tight upper bound on the complexity measure R . A key advantage of the formulation in (7), beyond simplifying the constraints, is its minimal information requirement: it only relies on the gradient of f evaluated at the original trained model, i.e., the side information $\mathcal{I}_r = \{\nabla_{\theta} f(\theta^*, x)\}_{(x,y) \in \mathcal{D}_r}$. This is significantly less restrictive than prior work, which requires access to the inverse Hessian of the loss over $\mathcal{D}_r$ [5–7], and makes our method substantially simpler than full retraining.

4 Theoretical Guarantees

This section provides theoretical guarantees for using our proposed relaxation (7) to solve the exact unlearning problem (4). Going forward, we denote Euclidean projection onto the set $\mathcal{S}$ as $\mathcal{P}_{\mathcal{S}}(\cdot)$ 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 θ^* with $\theta^{*\top} x_i = y_i$ for all $(x_i, y_i) \in \mathcal{D}$, we can easily solve the exact unlearning problem (4) for $R(\theta) = \|\theta\|_2$.

Theorem 2. *Let $\tilde{\Delta}$ solve (7) for $f(\theta, x) = \theta^\top x$ and $\tilde{R}(\theta) = \|\theta\|_2$. Then the recovered solution $\tilde{\theta} = \theta^* + \tilde{\Delta}$ solves the exact unlearning problem (4) for $R(\theta) = \|\theta\|_2$*

This result holds because, in the linear case, the surrogate and original constraints match exactly, and no approximation error is introduced. Thus, no additional regularizer (i.e., $\hat{R}(\cdot) = 0$) is needed.

4.2 L-Layer Linear Network

In this section, we extend our analysis to a more complex model: an L -layer linear network. Let the prediction function be $f(\theta, x) = c^\top A_{L-1} \cdots A_1 x$, where the parameter vector is partitioned $\theta = [c; \text{vec}(A_1); \dots; \text{vec}(A_{L-1})]$, with $A_\ell \in \mathbb{R}^{h_\ell \times h_{\ell-1}}$ and $c \in \mathbb{R}^{h_{L-1}}$ for $\ell = 1, \dots, L-1$. The input dimension is $m = h_0$, and we assume $n < m$ to reflect the overparameterized regime. For clarity, define the effective linear predictor $w(\theta) = A_1^\top \cdots A_{L-1}^\top c$, so that $f(\theta, x) = w(\theta)^\top x$. For this model class, we study two natural choices of regularizers in (4): (i) R as the norm of the prediction function as a linear map, and (ii) R as the norm of all model parameters.

4.2.1 Minimizing Predictor Norm

We first analyze when the R measures the ℓ_2 -norm of the effective linear predictor: $R(\theta) = \|A_1^\top \cdots A_{L-1}^\top c\|_2 = \|w(\theta)\|_2$. Given $\theta^* = [c^*; \text{vec}(A_1^*); \dots; \text{vec}(A_{L-1}^*)]$ such that $w(\theta^*)^\top x = y$ for all $(x, y) \in \mathcal{D}$, we aim to solve (4) for this choice of R . In this case the mapping f is non-linear with respect to θ . As a result, the first-order approximation for the constraints is not tight, so solving the surrogate problem in (7) does not necessarily give a solution for the problem in (4). However, we show that adding a suitable regularizer $\hat{R}$ to control model drift ensures the relaxed and original problems have the same solution. We first present an intermediate result showing the existence of a feasible perturbation $\tilde{\Delta}$ that satisfies the relaxed linearized constraints and, when added to θ^* , yields an optimal solution to (4).

Lemma 1. *Denote the retain set input subspace by $\mathcal{S}_r = \text{span}\{x \mid (x, y) \in \mathcal{D}_r\}$ and partition the perturbation as $\tilde{\Delta} = [\tilde{\Delta}_c; \text{vec}(\tilde{\Delta}_{A_1}); \dots; \text{vec}(\tilde{\Delta}_{A_{L-1}})]$ in the same manner as θ . Set*

$$\tilde{\Delta}_{A_1} = -\|A_2^{*\top} \cdots A_{L-1}^{*\top} c^*\|_2^{-2} A_2^{*\top} \cdots A_{L-1}^{*\top} c^* \mathcal{P}_{\mathcal{S}_r^\perp}(w(\theta^*))^\top \quad (8)$$

and all other components of $\tilde{\Delta}$ to zero. Then $\tilde{\Delta}$ is orthogonal to the gradient of mapping $f(\theta, x)$ evaluated at $\theta = \theta^$ for each input x in the retain set and hence feasible for the relaxed problem (7). Moreover, $\theta^* + \tilde{\Delta}$ solves the exact unlearning problem (4) for $R(\theta) = \|w(\theta)\|_2$.*

The above result shows that the perturbation direction defined in (8) leads to an optimal solution for (4) once added to θ^* , while satisfying the relaxed linear constraints of the surrogate problem. That

said, it does not imply that solving (6), which only differs in the constraints from (4), would recover $\tilde{\Delta}$. In fact, we can show that without adding a proper regularization term $\tilde{R}$ to the loss, $\tilde{\Delta}$ would not be a solution of the relaxed problem (see Appendix D.4.1). We next characterize the appropriate regularization $\tilde{R}(\Delta)$ needed to ensure that $\tilde{\Delta}$ is the optimal solution to the surrogate problem in (7).

Theorem 3. *The solution to the relaxed unlearning problem (7) with the following choice of $\tilde{R}$ solves the exact unlearning problem (4) for $R(\theta) = \|\mathbf{w}(\theta)\|_2$.*

$$\tilde{R}(\theta; \theta^*) = \|\mathbf{w}(\theta)\|_2 + \delta_{\{c \neq c^*\}} + \sum_{\ell=2}^{L-1} \delta_{\{A_\ell \neq A_\ell^*\}} \quad (9)$$

4.2.2 Minimizing Parameter Norm

Next, we analyze when the unlearning solution is the loss minimizer with the smallest parameter norm, so $R(\theta) = \|\theta\|_2$. In this case, we can construct an exact unlearning solution from the exact unlearning solution to the previously analyzed case when $R(\theta) = \|\mathbf{w}(\theta)\|_2$.

Theorem 4. *Let $\hat{\theta}_r^*$ solve (4) for $R(\theta) = \|\mathbf{w}(\theta)\|_2$, so $\mathbf{w}(\hat{\theta}_r^*)$ is the min ℓ_2 -norm linear predictor over $\mathcal{D}_r$. Define $\rho = \|\mathbf{w}(\hat{\theta}_r^*)\|_2$ and let $\mathbf{v}_\ell \in \mathbb{R}^{h_\ell}$ for $\ell \in [L-1]$ each satisfy $\|\mathbf{v}_\ell\|_2 = 1$. Set*

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

Then $\tilde{\theta} = [\tilde{c}; \text{vec}(\tilde{A}_1); \dots; \text{vec}(\tilde{A}_{L-1})]$ solves the exact unlearning problem (4) for $R(\theta) = \|\theta\|_2$.

Thus, the solution to the minimum norm predictor problem gives the solution to minimum parameter norm problem, so we can apply the previous results to find a solution for (4) with $R(\theta) = \|\mathbf{w}(\theta)\|_2$ using the constraint relaxation and then update the parameters as prescribed by Theorem 4.

4.3 2-Layer Perceptron

We lastly consider a 2-layer perceptron with a non-linear activation. Specifically, we define $f(\theta, \mathbf{x}) = \mathbf{c}^\top \phi(\mathbf{A}\mathbf{x})$, where we use the partition $\theta = [c; \text{vec}(\mathbf{A})]$ for $c \in \mathbb{R}^h$, $\mathbf{A} \in \mathbb{R}^{h \times m}$. Here, h is the total number of neurons and $\phi: \mathbb{R} \rightarrow \mathbb{R}$ is some activation function. We abuse notation and write $\phi(\mathbf{A}\mathbf{x})$ to denote the element-wise application of ϕ to $\mathbf{A}\mathbf{x}$. We analyze the case where R measures the number of active neurons, i.e., the width of the network. Formally, we denote $\mathbf{a}_i^\top$ as the i th row of $\mathbf{A}$, and we set $R(\theta) = \sum_{i=1}^h \mathbb{1}\{|\mathbf{c}_i| \|\mathbf{a}_i\|_2 > 0\}$. With this choice of R , the unlearning solution promotes recovering a sparse network which fits $\mathcal{D}_r$, where $\mathcal{D}_r$ has $n_r = |\mathcal{D}_r|$ samples. Given that $\mathbf{c}^{*\top} \phi(\mathbf{A}^* \mathbf{x}) = y$ for all $(\mathbf{x}, y) \in \mathcal{D}$, we chase the minimum neuron interpolating solution to $\mathcal{D}_r$:

$$\theta_r^* \in \underset{\theta}{\operatorname{argmin}} R(\theta) \quad \text{s.t.} \quad \mathbf{c}^\top \phi(\mathbf{A}\mathbf{x}) = y \quad \forall (\mathbf{x}, y) \in \mathcal{D}_r \quad (10)$$

While we aim to solve (10) for any retain set $\mathcal{D}_r$, the exact minimal-width solution remains unknown. Prior work shows that $n_r + 1$ neurons suffice for general activations [23], while for ReLU, some n_r -sample datasets need at least $n_r - 2$ neurons [24]. Here, we apply our framework to recover feasible unlearned networks with width at most n_r , improving the best known worst-case bound.

We begin by linearizing the constraints of problem (10) around θ^* , as directly solving this problem may be intractable due to the non-linear activation ϕ , especially since we assume access to only the model gradients over $\mathcal{D}_r$, not the samples in $\mathcal{D}_r$ themselves. We define the drift as $\Delta = [\Delta_c; \text{vec}(\Delta_A)]$, yielding the specific instance of the linearized problem (6) for this model class:

$$\min_{\Delta} R(\theta^* + \Delta) \quad \text{s.t.} \quad \Delta_c^\top \phi(\mathbf{A}^* \mathbf{x}_i) + \text{tr}\{\Delta_A^\top (\phi'(\mathbf{A}^* \mathbf{x}_i) \odot \mathbf{c}^*) \mathbf{x}_i^\top\} = 0 \quad \forall (\mathbf{x}_i, y_i) \in \mathcal{D}_r, \quad (11)$$

where $\odot$ denotes element-wise product. Due to the layered structure and non-linear activation ϕ , solving (11) may not ensure feasibility for (10), as the relaxed constraints are loose. We first show that a feasible perturbation $\tilde{\Delta}$, modifying only the last layer $\mathbf{c}^*$, exists and yields a network satisfying (10) with at most n_r active neurons.

Lemma 2. *Assume the finite-width network $f(\theta^*, \mathbf{x}) = \mathbf{c}^{*\top} \phi(\mathbf{A}^* \mathbf{x})$ interpolates $\mathcal{D}_r$, where $n_r = |\mathcal{D}_r|$ is the number of retain set samples. Let $\dim(\text{span}\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r}) = s \leq n_r$. Then, there exists a feasible perturbation $\tilde{\Delta}$ satisfying the linear constraints in (11), such that $f(\theta^* + \tilde{\Delta}, \cdot)$ interpolates $\mathcal{D}_r$, $R(\theta^* + \tilde{\Delta}) \leq s$, and $\tilde{\Delta}_A = 0$.*

Algorithm 1 MinNorm-OG

```
1: Input:  $\theta^*$ , loss  $\mathcal{J}(\theta)$ ,  $\mathcal{D}'_r \subseteq \mathcal{D}_r$ , step size  $\eta_t$ , regularization constant  $\lambda_t \geq 0$ , subsample batch size  $n_{\text{pert}}$ 
2: Initialize  $\theta \leftarrow \theta^*$ 
3: for  $t = 1, \dots, n_{\text{epochs}}$  do
4:   for each batch  $\mathcal{B}$  from  $\mathcal{D}'_r$  do
5:     if  $\lambda_t < \infty$  then
6:       Compute function gradients  $\mathbf{g}_i = \nabla_{\theta} f(\theta, \mathbf{x}_i)$  for  $\mathbf{x}_i \in \mathcal{B}$ ,  $i = 1, \dots, n_{\text{pert}}$ 
7:       Solve  $\tilde{\Delta} \leftarrow \operatorname{argmin}_{\Delta} \|\theta + \Delta\|_2^2 + \lambda_t \|\Delta\|_2^2$  s.t.  $\Delta \perp \mathbf{g}_i$  for all  $i \leq n_{\text{pert}}$ 
8:       Update  $\theta \leftarrow \theta + \tilde{\Delta}$ 
9:     Loss descent:  $\theta \leftarrow \theta - \eta_t \nabla_{\theta} \mathcal{J}(\theta; \mathcal{B})$ 
10: return  $\theta$ 
```

While Lemma 2 provides a feasible point for (11), it is not the solution, as the relaxed problem linearizes the interpolation constraint without limiting drift size, potentially losing interpolation over $\mathcal{D}_r$. The following theorem shows that choosing $\hat{R}$ to restrict perturbations in $\mathbf{A}^*$ ensures that solving (7) yields a network feasible for (10) with at most n_r active neurons.

Theorem 5. For $R(\theta)$ which measures the number of active neurons of the network $f(\theta, \cdot)$, define

$$\tilde{R}(\theta; \theta^*) = R(\theta) + \delta_{\{\mathbf{A} \neq \mathbf{A}^*\}} \quad (12)$$

as the surrogate objective. Then the solution to the relaxed unlearning problem (7) with this choice of $\tilde{R}$ results in a network which interpolates $\mathcal{D}_r$, achieving feasibility for the exact unlearning problem (10), and admits at most $s = \dim(\operatorname{span}\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r}) \leq n_r$ active neurons, where $n_r = |\mathcal{D}_r|$.

Theorem 5 shows that for general activation functions, linearizing the constraints to (10) and minimizing the sum of the complexity measure R along the appropriate regularizer $\hat{R}$ for the drift term recovers a network that interpolates $\mathcal{D}_r$ with at most s active neurons, where s is the dimension of the span of the learned representations $\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r}$. Since s can never exceed $n_r = |\mathcal{D}_r|$ our method guarantees a worst-case interpolation width of at most n_r , thereby improving the general bound of $n_r + 1$ implied by [23] for minimum width interpolation.

The drift regularizer $\hat{R}$ only allows perturbations to $\mathbf{c}^*$, so the solution to (7) reduces width via sparsity in the updated last layer $\mathbf{c}^* + \tilde{\Delta}_{\mathbf{c}}$, while leaving the first layer $\mathbf{A}^*$ unchanged. Although $\mathbf{c}^* + \tilde{\Delta}_{\mathbf{c}}$ relies on a small set of features, the feature map $\phi(\mathbf{A}^* \mathbf{x})$ still reflects representations learned from all of $\mathcal{D}$. We show, however, that the sparsity of $\mathbf{c}^* + \tilde{\Delta}_{\mathbf{c}}$ can be propagated into $\mathbf{A}^*$, producing a network with a new, sparser feature map that is less expressive and no longer consistent with having been trained on the full dataset $\mathcal{D}$, yet still satisfies all unlearning guarantees in Theorem 5.

Proposition 1. Let $\theta = [\mathbf{c}; \operatorname{vec}(\mathbf{A})]$ be any parameter vector, and define $\hat{\mathbf{A}} = (\mathbf{1}_{\mathbf{c} \neq 0}, \mathbf{1}^\top) \odot \mathbf{A}$. Then the updated parameters $\hat{\theta} = [\hat{\mathbf{c}}; \operatorname{vec}(\hat{\mathbf{A}})]$ satisfy: (i) $f(\theta, \mathbf{x}) = f(\hat{\theta}, \mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^m$, (ii) $R(\theta) = R(\hat{\theta})$, and (iii) $\hat{\mathbf{A}}$ has at most $R(\hat{\theta})$ number of nonzero rows.

Thus, for any parameters θ , we can apply a simple update to recover new parameters $\hat{\theta}$ which behave like an $R(\theta)$ -neuron network in terms of both the function outputs and at the parameter level. We apply this result to the solution to the relaxed unlearning problem (7) in the following corollary.

Corollary 1. Let $\tilde{\theta} = [\tilde{\mathbf{c}}; \operatorname{vec}(\mathbf{A}^*)]$ solve (7) for $\tilde{R}$ defined in (12), and define the updated first layer as $\hat{\mathbf{A}} = (\mathbf{1}_{\tilde{\mathbf{c}} \neq 0}, \mathbf{1}^\top) \odot \mathbf{A}^*$. Then network parameterized by $\hat{\theta} = [\hat{\mathbf{c}}; \operatorname{vec}(\hat{\mathbf{A}})]$ similarly interpolates $\mathcal{D}_r$, has the same number of active neurons $R(\tilde{\theta}) = R(\hat{\theta})$, and $\hat{\mathbf{A}}$ has at most $R(\hat{\theta})$ non-zero rows.

Thus, solving the relaxed problem (7) and updating $\mathbf{A}^*$ via Proposition 1 yields a network that reveals no trace of having been trained on the larger dataset $\mathcal{D} = \mathcal{D}_r \sqcup \mathcal{D}_f$, even at the representation level.

5 From Theory to Practice

We translate our framework into a practical algorithm MinNorm-OG (Algorithm 1). At epoch t , we alternate between solving a version of the relaxed unlearning problem (7) and descending the loss

Table 1: Data Poisoning experiment results, measured as the median sup-norm distance between the retain set trend $y = \sin(x)$ and the unlearned model outputs over 10 trials (smaller is better).

Epochs	Retrain	MinNorm-OG	GD	GA	NGP	NGD	Ridge	ℓ_1 -Sparse
10	1.50	1.50	3.23	2.56	2.50	2.73	2.27	3.28
100	1.36	1.08	2.76	23.8	2.62	2.85	2.13	1.79
1000	1.17	0.63	2.61	1400	2.75	2.45	1.96	1.37

on $\mathcal{D}_r$ to maintain feasibility for the exact unlearning problem (4), leveraging access to samples in $\mathcal{D}_r$. Steps 6-8 of Algorithm 1 denote solving (7) for $R(\theta) = \|\theta\|_2^2$ and $\hat{R}(\Delta) = \lambda_t \|\Delta\|_2^2$ where $\lambda_t \geq 0$ is a scaling parameter, and step 9 shows the loss descent step. To handle batched data and large models, we enforce the orthogonality constraint in (7) over a subsample of size n_{pert} within each batch. For this R and $\hat{R}$, the solution to (7) perturbs θ toward its projection onto the span of model gradients over this subsample (see Appendix E), which can be interpreted as a proximal update under the orthogonal gradient constraint. The main overhead relative to gradient descent comes from solving for $\tilde{\Delta}$ via a QR decomposition with complexity $\mathcal{O}(dn_{\text{pert}}^2)$, which is negligible compared to the $\mathcal{O}(dn_B)$ cost of gradient descent when $n_{\text{pert}} < \sqrt{n_B}$, where $n_B = |\mathcal{B}|$ is the batch size.

5.1 Experiments

We test our algorithm against the following baseline methods. Retrain erases the initial model and trains from scratch on $\mathcal{D}_r$. GD [14] runs gradient descent on $\mathcal{J}(\theta; \mathcal{D}_r)$, while Noisy GD (NGD) [16] adds gradient noise to the GD steps. GA [15] runs gradient ascent on $\mathcal{J}(\theta; \mathcal{D}_f)$. NegGrad+ (NGP) [8] minimizes a weighted combination of the GD and GA objectives. ℓ_1 -Sparse [25] runs GD with an ℓ_1 -norm regularizer. SCRUB [8] optimizes three objectives: minimizing $\mathcal{J}(\theta; \mathcal{D}_r)$, 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) [18] runs a form of gradient ascent over $\mathcal{J}(\theta; \mathcal{D}_f)$ inspired by preference optimization. SalUn [26] minimizes $\mathcal{J}(\theta; \mathcal{D}_r)$ while fitting random labels over $\mathcal{D}_f$, updating only the parameters that initially have large gradients with respect to $\mathcal{J}(\theta; \mathcal{D}_f)$. We apply SCRUB, NPO, and SalUn only in our classification-based experiments, as they are not directly designed for general regression tasks. To highlight the performance of our algorithm, we also compare to ridge regression, which approximates our unlearning objective (4) for $R(\theta) = \|\theta\|_2^2$ by minimizing $\mathcal{J}(\theta; \mathcal{D}_r) + \lambda_t \|\theta\|_2^2$. The minimizer of this regularized objective converges to the minimum- ℓ_2 -norm loss minimizer in the ridgeless limit as $\lambda_t \rightarrow 0$ [22].

While recent work proposed various unlearning benchmarks, especially for LLMs [16, 27–29], they often rely on opaque metrics that emphasize suppressing forget-set generation. In contrast, we present the following experiments with interpretable quantitative metrics. In each experiment, we evaluate all methods over a fixed number of unlearning epochs. In the Data Poisoning experiment, each epoch corresponds to a single unlearning update over the full dataset, $\mathcal{D} = \mathcal{D}_f \sqcup \mathcal{D}_r$. In contrast, for the larger-scale Multi-Class Label Erasure and Representation Collapse experiments, each method processes batches from the entire forget set $\mathcal{D}_f$ and only a subset of the retain set, $\mathcal{D}'_r \subset \mathcal{D}_r$. Here, one epoch is defined as a full pass over the forget set batches, where each forget batch is paired with a retain batch of equal size sampled from $\mathcal{D}'_r$. Complete details of our experimental setup and full results with uncertainty estimates are provided in Appendix F.

Data Poisoning. We train a shallow neural network on retain samples $(x_r, y_r) \in \mathcal{D}_r$ with $y_r = \sin(x_r)$ and forget samples $(x_f, y_f) \in \mathcal{D}_f$ with $y_f = 1.5$, over input domain $\mathcal{X} = [-5\pi, 5\pi] \subseteq \mathbb{R}$. We evaluate the output θ of each unlearning method by measuring the deviation from the retain set trend, given by $\sup_{x \in \mathcal{X}} |f(\theta, x) - \sin(x)|$. Results are reported in Table 1 as the median over 10 trials with visualizations in Figure 1. Retrain fails to closely capture the retain set trend $y = \sin(x)$, while the (regularized) loss descent methods (GD, NGD, Ridge) struggle to simultaneously escape the influence of the forget set and fit to the retain set. GA is unstable and diverges from the retain set. Due to this instability, we found that NGP performed best when the loss-ascent coefficient was near zero, causing it to behave like GD.

Multi-Class Label Erasure. We use the CIFAR-10 [30] and Tiny ImageNet [31] datasets, creating red, green, and gray copies of each image. We train modified ResNet-18 and ResNet-50 models on

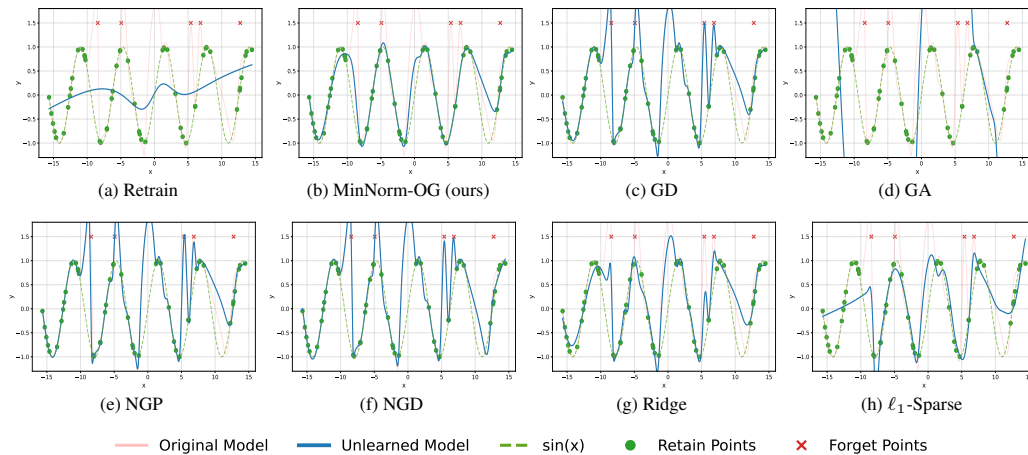


Figure 1: Example unlearned model fits when given 100 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

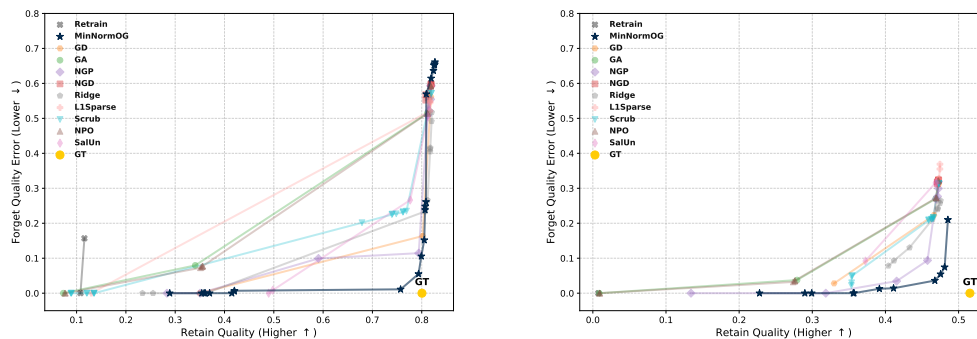


Figure 2: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure experiment on colored versions of CIFAR-10 (left) and Tiny ImageNet (right). Models predict color and content, but the retain set contains only gray images. The ground truth unlearned model (GT) performs well on gray inputs but always predicts gray with probability 1. 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) best approaches the ground-truth unlearned model's performance relative to the other baselines.

CIFAR-10 and Tiny ImageNet, respectively, to jointly predict image class and color. The retain set $\mathcal{D}_r$ contains all image content classes only in gray, while the forget set $\mathcal{D}_f$ contains all colors. The ground truth unlearned model predicts gray content well and always predicts gray color with probability 1, regardless of input. We evaluate retain quality by accuracy on gray-colored test samples, and we measure forget quality error as the mean squared error between the model's predicted probability of an image being gray and the target value of 1 for all colored inputs.

Figure 2 presents the Pareto frontier of the mean performance across 5 trials for each method under different hyperparameter settings. The optimal point $(1, 0)$ indicates perfect retain quality and zero forget quality error. The ground truth unlearned model is labeled GT. Our method, MinNorm-OG, performs best across both datasets, as the other methods struggle to unlearn the influence of the colored forget set samples without harming accurate classification of gray test images. Notably, retraining from scratch yields a model with nearly perfect forget quality error, as it is trained only on gray images, but performs no better than random guessing on gray test images due to the limited training epochs and restricted retain set access.

Representation Collapse. We again use the CIFAR-10 dataset where each of the 10 classes is assigned a unique color. The retain set $\mathcal{D}_r$ contains the CIFAR-10 images colored according to their unique color, while the forget set $\mathcal{D}_f$ comprises randomly colored images. The ground truth

Table 2: Representation Collapse experiment results across constraints on the number of unlearning epochs and percentage of accessible retain set samples. Models are trained on colored images where color perfectly predicts the label in the retain set but not in the full dataset $\mathcal{D}$. Evaluation is measured as the mean accuracy (%) on test images labeled by color over 5 trials (higher is better).

Retain %	Epochs	Retrain	MinNorm-OG	GD	GA	NGP	NGD	Ridge	ℓ_1 -Sparse	Scrub	NPO	SalUn
0.1	5	79.1	49.1	24.0	34.5	45.9	12.6	20.9	12.4	37.3	40.6	29.4
	10	95.5	78.7	55.2	38.3	73.7	33.0	55.2	23.0	61.9	43.2	53.4
	15	96.3	93.2	78.2	39.8	81.5	46.9	78.2	25.1	74.1	44.6	76.3
1	5	92.8	59.5	40.7	34.2	58.3	32.9	33.5	12.5	47.8	42.5	42.5
	10	98.5	94.7	73.6	38.2	92.4	70.5	63.4	31.4	75.8	43.5	68.7

Table 3: Average time in seconds to perform a single unlearning epoch on the Multi-Class Label Erasure experiment on Tiny ImageNet using ResNet-50, averaged over 5 trials.

Method	Retrain	MinNorm-OG	GD	GA	NGP	NGD	Ridge	ℓ_1 -Sparse	Scrub	NPO	SalUn
Time (s)	0.90	0.78	0.67	0.68	0.74	0.69	0.68	0.69	0.96	0.94	1.09

unlearned model predicts from color alone, as the color feature completely determines the class label and is easier to learn than the image content features. In contrast, models trained on the full dataset $\mathcal{D} = \mathcal{D}_r \sqcup \mathcal{D}_f$ must predict based on content, since color is no longer fully predictive of the label. For evaluation, we label heldout test 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.

Table 2 presents mean results over 5 trials. We observe that Retrain achieves much higher color classification accuracy than any unlearning method, as image color is an easy feature to learn from scratch. However, this result is specific to this setup: since the target function over the retain set is so simple, it is easier to learn from a freshly initialized model than by adapting the original model, which has learned to rely on complex image content features. Our prior experiments demonstrate that Retrain is not a viable unlearning strategy in general, so we report its results separately in the leftmost column. We bold the MinNorm-OG results, as they achieve the highest accuracy for each combination of constraints on the percentage of accessible retain set samples and number of unlearning epochs.

5.2 Runtime Comparison

In the above experiments, we compare all unlearning methods under a fixed number of epochs. To assess per-epoch computational efficiency, we measure the runtime required by each method to complete a single unlearning epoch in the Multi-Class Label Erasure experiment using Tiny ImageNet with ResNet-50, our largest-scale setting. The results are reported in Table 3.

We observe that methods which compute loss gradients with simple regularizers on a single data split (GD, GA, NGD, Ridge, ℓ_1 -Sparse) are the fastest. NGP is slower, as it requires forward and backward passes over both retain and forget set batches. Our method, MinNorm-OG, operates only on the retain set but solves an additional subproblem on top of the loss descent step, while maintaining a runtime comparable to NGP. Scrub and NPO are slower, as they require extra forward passes using the original model to evaluate their loss functions, while Retrain is slowed by the need to initialize a new model from scratch. Lastly, SalUn is the slowest due to an initial pass over the entire dataset to compute the threshold for deciding which parameters to freeze during unlearning.

6 Conclusion

We proposed a new unlearning framework under overparameterization by seeking the simplest solution consistent with the retain set. We proved guarantees on solving the exact unlearning problem through a tractable relaxed formulation. A practical implementation of our framework outperformed baselines, as the simplest solution aligns with unlearning goals and removes artifacts unrelated to the retain set. While our theoretical guarantees open the door for unlearning analysis beyond the underparameterized setting, we focused on model classes like linear networks and two-layer perceptrons. We naturally aim to analyze unlearning in more complex settings like deep networks in future work, as well as experiment within broader domains at larger scale.

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.

References

- [1] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016, 2016.
- [2] California State Legislature. California Consumer Privacy Act of 2018. Statute, 2018. Cal. Civ. Code § 1798.100 et seq.
- [3] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. Bias and fairness in large language models: A survey. *Computational Linguistics*, 50(3):1097–1179, 2024.
- [4] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015. doi: 10.1109/SP.2015.35.
- [5] Juhan Bae, Nathan Ng, Alston Lo, Marzyeh Ghassemi, and Roger B Grosse. If influence functions are the answer, then what is the question? *Advances in Neural Information Processing Systems*, 35:17953–17967, 2022.
- [6] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.
- [7] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In *International Conference on Machine Learning*, pages 3832–3842. PMLR, 2020.
- [8] Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, 36:1957–1987, 2023.
- [9] Martin Pawelczyk, Jimmy Z Di, Yiwei Lu, Gautam Kamath, Ayush Sekhari, and Seth Neel. Machine unlearning fails to remove data poisoning attacks. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [10] Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary Chase Lipton, and J Zico Kolter. TOFU: A task of fictitious unlearning for LLMs. In *First Conference on Language Modeling*, 2024.
- [11] Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974.
- [12] Badih Ghazi, Pritish Kamath, Ravi Kumar, Pasin Manurangsi, Ayush Sekhari, and Chiyuan Zhang. Ticketed learning–unlearning schemes. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 5110–5139. PMLR, 2023.
- [13] Lucas Bourtole, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE symposium on security and privacy (SP)*, pages 141–159. IEEE, 2021.
- [14] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR, 2021.

- [15] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021.
- [16] Rishav Chourasia and Neil Shah. Forget unlearning: Towards true data-deletion in machine learning. In *International Conference on Machine Learning*, pages 6028–6073. PMLR, 2023.
- [17] Xiaomeng Jin, Zhiqi Bu, Bhanukiran Vinzamuri, Anil Ramakrishna, Kai-Wei Chang, Volkan Cevher, and Mingyi Hong. Unlearning as multi-task optimization: A normalized gradient difference approach with an adaptive learning rate. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11278–11294, 2025.
- [18] Ruiqi Zhang, Licong Lin, Yu Bai, and Song Mei. Negative preference optimization: From catastrophic collapse to effective unlearning. In *First Conference on Language Modeling*, 2024.
- [19] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.
- [20] Huiqiang Chen, Tianqing Zhu, Xin Yu, and Wanlei Zhou. Machine unlearning via null space calibration. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 358–366, 2024.
- [21] Tuan Hoang, Santu Rana, Sunil Gupta, and Svetha Venkatesh. Learn to unlearn for deep neural networks: Minimizing unlearning interference with gradient projection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4819–4828, 2024.
- [22] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, April 2022. doi: 10.1214/21-AOS2144.
- [23] Saharon Rosset, Grzegorz Swirszcz, Nathan Srebro, and Ji Zhu. ℓ_1 regularization in infinite dimensional feature spaces. In *International Conference on Computational Learning Theory*, pages 544–558. Springer, 2007.
- [24] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. *Advances in Neural Information Processing Systems*, 32, 2019.
- [25] Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsity can simplify machine unlearning. *Advances in Neural Information Processing Systems*, 36:51584–51605, 2023.
- [26] Chongyu Fan, Jiancheng Liu, Yihua Zhang, Eric Wong, Dennis Wei, and Sijia Liu. Salun: Empowering machine unlearning via gradient-based weight saliency in both image classification and generation. In *The Twelfth International Conference on Learning Representations*, 2023.
- [27] Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A Smith, and Chiyuan Zhang. Muse: Machine unlearning six-way evaluation for language models. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [28] Anil Ramakrishna, Yixin Wan, Xiaomeng Jin, Kai Wei Chang, Zhiqi Bu, Bhanukiran Vinzamuri, Volkan Cevher, Mingyi Hong, and Rahul Gupta. SemEval-2025 task 4: Unlearning sensitive content from large language models. In *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*, pages 2584–2596. Association for Computational Linguistics, July 2025.
- [29] Anil Ramakrishna, Yixin Wan, Xiaomeng Jin, Kai-Wei Chang, Zhiqi Bu, Bhanukiran Vinzamuri, Volkan Cevher, Mingyi Hong, and Rahul Gupta. LUME: LLM unlearning with multitask evaluations. *arXiv preprint arXiv:2502.15097*, 2025.
- [30] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. Technical Report.

- [31] Stanford CS231n. Tiny imagenet visual recognition challenge, 2015. Course project from Stanford University's CS231n: Convolutional Neural Networks for Visual Recognition.
- [32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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 $\mathbf{x} \in \mathbb{R}^d$ and $A \subseteq \mathbb{R}^d$, $\mathcal{P}_A(\mathbf{x})$ is the Euclidean projection onto A . For $\mathbf{Z} \in \mathbb{R}^{m \times n}$, $\text{vec}(\mathbf{Z}) \in \mathbb{R}^{mn}$ is the columnwise vectorization. $\text{im}(\mathbf{Z})$, $\text{ker}(\mathbf{Z})$, and $\text{row}(\mathbf{Z})$ denote the image, kernel, and rowspace. $\|\mathbf{Z}\|_F$ is the Frobenius norm, $\|\mathbf{Z}\|_*$ is the nuclear norm, and $\|\mathbf{Z}\|_2$ is the spectral norm. For $\mathbf{Y} \in \mathbb{R}^{m \times n}$, $\langle \mathbf{Z}, \mathbf{Y} \rangle$ is the Frobenius inner product and $\mathbf{Z} \odot \mathbf{Y}$ is the element-wise product. $\text{tr}\{\cdot\}$ is the trace. For $\mathbf{x} \in \mathbb{R}^{d_x}$ and $\mathbf{y} \in \mathbb{R}^{d_y}$, $[\mathbf{x}; \mathbf{y}] \in \mathbb{R}^{d_x+d_y}$ stacks $\mathbf{x}$ and $\mathbf{y}$. $\|\mathbf{x}\|_p$ is the ℓ_p norm. $[n] = \{1, \dots, n\}$. 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 $\mathbf{x} \in \mathbb{R}^d$ and $c \in \mathbb{R}$, let $\mathbf{1}_{\mathbf{x} \neq c}$ denote the vector which is 1 in each entry of $\mathbf{x}$ 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 $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with sample inputs $\mathbf{x}_i \in \mathbb{R}^m$ and outputs $y_i \in \mathbb{R}$. We consider training a linear model $f(\boldsymbol{\theta}, \mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$ 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 $\mathcal{S} = \text{span}\{\mathbf{x} \mid (\mathbf{x}, y) \in \mathcal{D}\}$, and assume $\dim(\mathcal{S}) = n$ so the regression problem is realizable. Consider solving the following problem for finding the linear regression solution with minimum ℓ_2 norm:

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

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

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 \text{ s.t. } \mathbf{y} = \mathbf{X}\boldsymbol{\theta} \quad (13)$$

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

Lemma 3. $\boldsymbol{\theta}^*$ is the unique vector in $\text{row}(\mathbf{X})$ which is feasible for (13)

Proof. The objective (13) 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.

$$\begin{aligned} \min_{\boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 \quad \text{s.t.} \quad \mathbf{y} = \mathbf{X}\boldsymbol{\theta} &= \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\lambda} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 + \boldsymbol{\lambda}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \\ &= \max_{\boldsymbol{\lambda}} \min_{\boldsymbol{\theta}} \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 + \boldsymbol{\lambda}^\top (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \\ &= \max_{\boldsymbol{\lambda}} \frac{1}{2} \|\mathbf{X}^\top \boldsymbol{\lambda}\|_2^2 + \boldsymbol{\lambda}^\top (\mathbf{y} - \mathbf{X}\mathbf{X}^\top \boldsymbol{\lambda}) \quad \text{s.t.} \quad \boldsymbol{\theta} = \mathbf{X}^\top \boldsymbol{\lambda} \\ &= \max_{\boldsymbol{\lambda}} -\frac{1}{2} \|\mathbf{X}^\top \boldsymbol{\lambda}\|_2^2 + \boldsymbol{\lambda}^\top \mathbf{y} \quad \text{s.t.} \quad \boldsymbol{\theta} = \mathbf{X}^\top \boldsymbol{\lambda} \\ &\implies \mathbf{X}\mathbf{X}^\top \boldsymbol{\lambda}^* = \mathbf{y} \text{ and } \boldsymbol{\theta}^* = \mathbf{X}^\top \boldsymbol{\lambda}^* \end{aligned} \quad (14)$$

Thus the primal solution $\boldsymbol{\theta}^*$ must be of the form $\mathbf{X}^\top \boldsymbol{\lambda}^* \in \text{row}(\mathbf{X})$. To show uniqueness, consider $\boldsymbol{\theta}_1^*, \boldsymbol{\theta}_2^* \in \text{row}(\mathbf{X})$ that are both feasible for (13). Then,

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

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

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

□

Using the same analysis, we can characterize the entire feasible set in terms of θ^* .

Lemma 4. *The feasible set to (13) $\{\theta \mid y = X\theta\} = \theta^* + \ker(X)$.*

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

To show the converse, take any $z \in \ker(X)$. Then $X(\theta^* + z) = X\theta^* + Xz = X\theta^* = y$. $\square$

Using this characterization of θ^* and the feasible set, we can clearly 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 5. *Consider any subset $\mathcal{D}_r \subseteq \mathcal{D}$, and define θ_r^* as the linear regression solution over just $\mathcal{D}_r$ with minimum norm:*

$$\theta_r^* = \underset{\theta}{\operatorname{argmin}} \|\theta\|_2 \text{ s.t. } f(\theta, x) = y \quad \forall (x, y) \in \mathcal{D}_r \quad (15)$$

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

Proof. θ^* already satisfies the feasibility constraint over the whole dataset $\mathcal{D}$, so it must be feasible for (15). Applying Lemmas 3 and 4 to the minimum norm problem over just $\mathcal{D}_r$ (15), we must have that $\theta_r^* \in \mathcal{S}_r$ and $\theta^* = \theta_r^* + z$ for some $z \in \mathcal{S}_r^\perp$. Then,

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

$\square$

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 $f(\theta, x) = \theta^\top x$ parameterized by $\theta \in \mathbb{R}^m$. Further suppose the original dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ has n samples with $x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$. Denote the subspace $\mathcal{S} = \operatorname{span}\{x \mid (x, y) \in \mathcal{D}\}$, and assume $\dim(\mathcal{S}) = n$ 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

$$\mathcal{J}(\theta; \mathcal{D}) = \frac{1}{n} \sum_{(x, y) \in \mathcal{D}} (y - \theta^\top x)^2$$

Consider when the learning algorithm $\mathcal{A}$ runs gradient descent on the loss, initialized at 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 $\mathcal{S}$.

Proposition 2. *Let $\mathcal{A}^k(\mathcal{D})$ be a learning algorithm which runs k steps of gradient descent on $\mathcal{J}(\theta; \mathcal{D})$ initialized at 0 , and define $\mathcal{S} = \operatorname{span}\{x \mid (x, y) \in \mathcal{D}\}$. If $\lim_{k \rightarrow \infty} \mathcal{A}^k(\mathcal{D})$ converges to some θ^* , then*

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

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

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

D Proofs

D.1 Proof of Theorem 1

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

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

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

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

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

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

D.2 Proof of Theorem 2

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

$$\theta_r^* = \underset{\theta}{\operatorname{argmin}} \|\theta\|_2 \quad \text{s.t. } \theta^\top x = y \quad \forall (x, y) \in \mathcal{D}_r \quad (16)$$

Consider solving the relaxed unlearning problem (7) for $\tilde{R}(\theta) = \|\theta\|_2$:

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

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

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

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

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

Thus the updated unlearned vector is

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

Then, $\mathcal{P}_{\mathcal{S}_r}(\theta^*) = \theta_r^*$ by Lemma 5.

D.3 Proof of Lemma 1

Recall that in this case we are interested in minimizing $R(\theta) = \|\mathbf{w}(\theta)\|_2$, where $\mathbf{w}(\theta) = A_1^\top \cdots A_{L-1}^\top c$ returns the effective linear predictor parameterized by θ .

We first show that $\tilde{\Delta}$ is feasible for the relaxed problem (7). Firstly, $\tilde{\Delta}$ is zero in all entries except those corresponding to the perturbation of A_1 , so we only need to ensure that $\tilde{\Delta}_{A_1}$ is orthogonal to $\nabla_{A_1} f(\theta^*, x)$ for each $(x, y) \in \mathcal{D}_r$. Recall we denote the retain set input space as $S_r = \text{span}\{x \mid (x, y) \in \mathcal{D}_r\}$, and $\tilde{\Delta}_{A_1}$ is defined as

$$\tilde{\Delta}_{A_1} = -\|A_2^{*\top} \cdots A_{L-1}^{*\top} c^*\|_2^{-2} A_2^{*\top} \cdots A_{L-1}^{*\top} c^* \mathcal{P}_{S_r^\perp} (w(\theta^*))^\top.$$

Further, the gradients are computed as

$$\nabla_{A_1} f(\theta^*, x) = A_2^{*\top} \cdots A_{L-1}^{*\top} c^* x^\top$$

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

$$\begin{aligned} \langle \tilde{\Delta}_{A_1}, \nabla_{A_1} f(\theta^*, x) \rangle &= \text{tr} \left\{ \left(\tilde{\Delta}_{A_1} \right)^\top \nabla_{A_1} f(\theta^*, x) \right\} \\ &= \text{tr} \left\{ \nabla_{A_1} f(\theta^*, x) \left(\tilde{\Delta}_{A_1} \right)^\top \right\} \\ &= -\|A_2^{*\top} \cdots A_{L-1}^{*\top} c^*\|_2^{-2} \text{tr} \left\{ A_2^{*\top} \cdots A_{L-1}^{*\top} c^* x^\top \mathcal{P}_{S_r^\perp} (w(\theta^*)) c^{*\top} A_{L-1}^* \cdots A_2^* \right\} \\ &= 0, \end{aligned}$$

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

We now show that $\theta^* + \tilde{\Delta}$ achieves the optimal unlearning solution θ^* . By construction of $\tilde{\Delta}$, the only entries of θ^* that are perturbed are those which correspond to A_1 . Thus, we compute the effective linear predictor after the perturbation:

$$\begin{aligned} w(\theta^* + \tilde{\Delta}) &= w(\theta^*) + \tilde{\Delta}_{A_1}^\top A_2^{*\top} \cdots A_{L-1}^{*\top} c^* \\ &= w(\theta^*) - \|A_2^{*\top} \cdots A_{L-1}^{*\top} c^*\|_2^{-2} \mathcal{P}_{S_r^\perp} (w(\theta^*)) c^{*\top} A_{L-1}^* \cdots A_2^* A_2^{*\top} \cdots A_{L-1}^{*\top} c^* \\ &= w(\theta^*) - \|A_2^{*\top} \cdots A_{L-1}^{*\top} c^*\|_2^{-2} \mathcal{P}_{S_r^\perp} (w(\theta^*)) (A_2^{*\top} \cdots A_{L-1}^{*\top} c^*)^\top A_2^{*\top} \cdots A_{L-1}^{*\top} c^* \\ &= w(\theta^*) - \mathcal{P}_{S_r^\perp} (w(\theta^*)) \\ &= \mathcal{P}_{S_r} (w(\theta^*)) \end{aligned}$$

Since the linear predictor $w(\theta^*)$ already interpolated $\mathcal{D}$, $\mathcal{P}_{S_r} (w(\theta^*))$ must be the minimum norm linear predictor over $\mathcal{D}_r$ by Lemma 5. Thus, the effective predictor of the perturbed parameters $w(\theta^* + \tilde{\Delta})$ solves the exact unlearning problem (4) when $R(\theta) = \|w(\theta)\|_2$, so $\theta^* + \tilde{\Delta}$ achieves the optimal unlearning solution.

D.4 Proof of Theorem 3

Recall for this theorem we analyze $R(\theta) = \|w(\theta)\|_2$. Let $\tilde{\Delta}$ be the perturbation which satisfies the conditions in Lemma 1. Then, $\tilde{\Delta}$ is feasible for the relaxed problem (7), and further $\theta^* + \tilde{\Delta}$ solves the exact unlearning problem (4).

Now, let Δ^* minimize the relaxed problem (7) for this $\tilde{R}$ defined in (9). Then because $\tilde{R}$ ensures that all elements of Δ^* which do not correspond to A_1 are zero, we must have that for any $(x, y) \in \mathcal{D}_r$:

$$\begin{aligned} w(\theta^* + \Delta^*)^\top x &= c^{*\top} A_{L-1}^* \cdots A_2^* (A_1^* + \Delta_{A_1}^*) x \\ &= y + c^{*\top} A_{L-1}^* \cdots A_2^* \Delta_{A_1}^* x \\ &= y + \langle \Delta^*, \nabla_{\theta} f(\theta^*, x) \rangle \\ &= y, \end{aligned}$$

where the last equality follows from the feasibility of Δ^* to (7). Thus, $\theta^* + \Delta^*$ interpolates $\mathcal{D}_r$, so $\theta^* + \Delta^*$ is feasible for the exact unlearning problem (4). We now show this point is also optimal for (4).

Since $\theta^* + \tilde{\Delta}$ solves the exact unlearning problem (4) and $\theta^* + \Delta^*$ is another feasible point, we must have that

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

Further, both $\tilde{\Delta}$ and Δ^* are feasible for (7) and Δ^* is defined as the solution to (7), so we must have that

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

But, since both $\tilde{\Delta}$ and Δ^* are non-zero only in the entries corresponding to A_1 , applying R and $\tilde{R}$ yields the same value:

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

Thus, $R(\theta^* + \Delta^*) = R(\theta^* + \tilde{\Delta})$, so $\theta^* + \Delta^*$ achieves the optimal objective value of (4). Since we established feasibility and optimality, $\theta^* + \Delta^*$ must solve (4).

D.4.1 Necessity of Additional Regularizer $\hat{R}$ for Theorem 3

In this section, we show that minimizing just R over the relaxed constraints, i.e. solving (6), for R which measures the linear network predictor norm does not solve the exact unlearning solution. Because there is no control the size and direction of the perturbation Δ , we can construct a simple example where Δ satisfies just the linearization of the data interpolation constraints but the updated network $\theta^* + \Delta$ no longer interpolates $\mathcal{D}_r$.

Consider a dataset of two samples $\mathcal{D} = \{(e_1, 1), (e_2, 1)\}$, where $e_i \in \mathbb{R}^m$ is the i th standard basis vector for any $m \geq 3$. Consider the original 2-layer interpolating network trained on $\mathcal{D}$ defined by parameters $\theta^* = [c^*; \text{vec}(A^*)]$, where $c^* = e_1 + e_2 \in \mathbb{R}^m$ and A^* is the $m \times m$ identity matrix $A^* = I_m$, so $f(\theta^*, x) = c^{*\top} A^* x = (e_1 + e_2)^\top x$.

We set $\mathcal{D}_r = \{(e_1, 1)\}$ and $\mathcal{D}_f = \{(e_2, 1)\}$, and define the perturbation variable $\Delta = [\Delta_c; \text{vec}(\Delta_A)]$. Translating the constraints of (6) to this specific problem instance, we have that

$$\Delta_c^\top e_1 + \text{tr}\{\Delta_A^\top (e_1 + e_2) e_1^\top\} = 0$$

We then select the values $\Delta_c = -e_3$ and $\Delta_A = e_3 e_1^\top - e_2 e_2^\top - e_3 e_3^\top$. It is easy to see that these choices satisfy the above constraint. Further, they achieve exact minimization of (6). We show below that the resulting network's predictor $(A^* + \Delta_A)^\top (c^* + \Delta_c) = 0$.

$$\begin{aligned} R(\theta^* + \Delta) &= \|(A^* + \Delta_A)^\top (c^* + \Delta_c)\|_2 \\ &= \|(I + e_3 e_1^\top - e_2 e_2^\top - e_3 e_3^\top)^\top (e_1 + e_2 - e_3)\|_2 \\ &= \|(I + e_1 e_3^\top - e_2 e_2^\top - e_3 e_3^\top)(e_1 + e_2 - e_3)\|_2 \\ &= \|e_1 + e_2 - e_3 - e_2 - e_1 + e_3\|_2 \\ &= \|0\|_2 = 0 \end{aligned}$$

Thus, the updated network which solves (6) predicts the constant function at 0 for all inputs x , as $f(\theta^* + \Delta, x) = ((A^* + \Delta_A)^\top (c^* + \Delta_c))^\top x = 0^\top x = 0$.

This clearly does not interpolate $\mathcal{D}_r$, and this example as a whole demonstrates that failing to control the size and direction of the drift term Δ beyond just the linearized constraints does not lead to the exact unlearning solution.

D.5 Proof of Theorem 4

Denote the minimum ℓ_2 norm solution $w(\hat{\theta}_r^*)$ to $y = Xw$ as just w_r^* for brevity. Using w_r^* , we construct a solution to the exact unlearning problem (4) for $R(\theta) = \|\theta\|_2$, which we restate below:

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

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

$$\underset{c, \mathbf{A}_1, \dots, \mathbf{A}_{L-1}}{\text{argmin}} \quad \|\mathbf{c}\|_2^2 + \sum_{\ell=1}^{L-1} \|\mathbf{A}_\ell\|_F^2 \quad \text{s.t.} \quad \mathbf{y}_r = \mathbf{X}_r \mathbf{A}_1^\top \dots \mathbf{A}_{L-1}^\top \mathbf{c} \quad (17)$$

Let $\mathbf{c}^*, \mathbf{A}_1^*, \dots, \mathbf{A}_{L-1}^*$ be a solution to (17). Then, $\mathbf{A}_1^{*\top} \dots \mathbf{A}_{L-1}^{*\top} \mathbf{c}^*$ interpolates $\mathcal{D}_r$, so $\mathbf{A}_1^{*\top} \dots \mathbf{A}_{L-1}^{*\top} \mathbf{c}^* = \mathbf{w}_r^* + \mathbf{z}$ where $\mathbf{w}_r^* \in \text{row}(\mathbf{X}_r)$ and $\mathbf{z} \in \ker(\mathbf{X}_r)$ by Lemma 4.

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

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

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

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

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

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

Then, (17) reduces to

$$\begin{aligned} \min_{\lambda_i, \mathbf{v}_\ell} \quad & \|\lambda_L \mathbf{v}_{L-1}\|_2^2 + \|\lambda_1 \mathbf{v}_1 \mathbf{w}_r^{*\top}\|_F^2 + \sum_{\ell=2}^{L-1} \|\lambda_\ell \mathbf{v}_\ell \mathbf{v}_{\ell-1}^\top\|_F^2 \\ \text{s.t.} \quad & (\lambda_1 \mathbf{w}_r^* \mathbf{v}_1^\top)(\lambda_2 \mathbf{v}_1 \mathbf{v}_2^\top) \dots (\lambda_{L-1} \mathbf{v}_{L-2} \mathbf{v}_{L-1}^\top)(\lambda_L \mathbf{v}_{L-1}) = \mathbf{w}_r^* \text{ and } \|\mathbf{v}_\ell\|_2 = 1 \\ = \min_{\lambda_i} \quad & \|\mathbf{w}_r^*\|_2^2 \lambda_1^2 + \sum_{\ell=2}^L \lambda_\ell^2 \quad \text{s.t.} \quad \lambda_1 \lambda_2 \dots \lambda_L = 1 \end{aligned} \quad (18)$$

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

$$\min_{\gamma_i > 0} \quad \|\mathbf{w}_r^*\|_2^2 \gamma_1 + \sum_{\ell=2}^L \gamma_\ell \quad \text{s.t.} \quad \gamma_1 \gamma_2 \dots \gamma_L = 1 \quad (19)$$

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

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

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

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

Then $g(\gamma^*) = \|\mathbf{w}_r^*\|_2^{2/L}$ achieves the lower bound, so it must solve (19). Thus, the optimal values $\lambda_1^*, \dots, \lambda_L^*$ to (18) result from taking square roots of γ_ℓ^* . Then, the following values for the network parameters must be optimal for (17):

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

D.6 Proof of Theorem 5

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

Lemma 6. For $\mathbf{c} \in \mathbb{R}^h$ and subspace $\mathcal{G} \subseteq \mathbb{R}^h$ such that $\dim(\mathcal{G}) = s$, there exists $\Delta_c \in \mathcal{G}_r^\perp$ such that $\|\mathbf{c} + \Delta_c\|_0 \leq s$, where the ℓ_0 -“norm” $\|\cdot\|_0$ counts the number of non-zero elements.

Because $\hat{R}$ does not allow any perturbation of $\mathbf{A}^*$, any solution to (12) must only perturb $\boldsymbol{\theta}^*$ in the entries corresponding to $\mathbf{c}^*$.

Let $s = \dim(\text{span}\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r})$. Note that by definition we have that $s \leq |\mathcal{D}_r|$. Apply the lemma to $\mathbf{c}^*$ and $\text{span}\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r}$ so that there exists $\tilde{\Delta}_c \in \text{span}(\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r})^\perp$ such that $\|\mathbf{c}^* + \tilde{\Delta}_c\|_0 \leq s$. Define $\tilde{\Delta} = [\tilde{\Delta}_c; \mathbf{0}]$.

Then the network defined by $\boldsymbol{\theta}^* + \tilde{\Delta}$ has at most s active neurons since any zero element of $\mathbf{c}^* + \tilde{\Delta}_c$ cannot contribute an active neuron. Further, $\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r} = \{\nabla_{\mathbf{c}} f(\boldsymbol{\theta}^*, \mathbf{x})\}_{\mathbf{x}, y \in \mathcal{D}_r}$, so the perturbation $\tilde{\Delta}$ is feasible for the relaxed problem (7). But, f is linear in $\mathbf{c}$, so this perturbation must preserve function value on $\mathcal{D}_r$, since the constraints of the relaxed problem are tight when just perturbing $\mathbf{c}^*$. Thus, the resulting network defined by $\boldsymbol{\theta}^* + \tilde{\Delta}$ both interpolates $\mathcal{D}_r$ and has at most $s = \dim(\text{span}\{\phi(\mathbf{A}^* \mathbf{x})\}_{(\mathbf{x}, y) \in \mathcal{D}_r})$ active neurons.

Note that this construction of $\tilde{\Delta}$ satisfies the conditions of Lemma 2, so we do not include a separate proof of the lemma since it is contained within the larger proof of the theorem.

Proof of Lemma 6:

Let the columns of some $\mathbf{P} \in \mathbb{R}^{h \times (h-s)}$ form a basis for $\mathcal{G}^\perp$ so that $\text{im}(\mathbf{P}) = \mathcal{G}^\perp$. Consider the reduced column echelon form of $\mathbf{P}$ denoted $\text{rcef}(\mathbf{P}) = \tilde{\mathbf{P}}$. By definition, $\text{im}(\tilde{\mathbf{P}}) = \text{im}(\mathbf{P}) = \mathcal{G}^\perp$, so $\text{rank}(\tilde{\mathbf{P}}) = h - s$ and thus each of the $h - s$ columns of $\tilde{\mathbf{P}}$ has a leading one. Let $\tilde{\mathbf{p}}_i$ be the i th column of $\tilde{\mathbf{P}}$ and let j_i denote the index of the leading one in $\tilde{\mathbf{p}}_i$ for all $i \in [h - s]$.

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

$$\Delta_c = \sum_{i=1}^{h-s} \gamma_i \tilde{\mathbf{p}}_i$$

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

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

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

D.7 Proof of Proposition 1

Consider any parameter vector $\boldsymbol{\theta} = [\mathbf{c}; \text{vec}(\mathbf{A})]$. Then for any input $\mathbf{x}$, we can write $f(\boldsymbol{\theta}, \mathbf{x}) = \sum_{i=1}^h \mathbf{c}_i \phi(\mathbf{a}_i^\top \mathbf{x})$ where $\mathbf{c}_i$ is the i th element of $\mathbf{c}$ and $\mathbf{a}_i^\top$ is the i th row of $\mathbf{A}$. Consider the updated parameters $\hat{\boldsymbol{\theta}} = [\hat{\mathbf{c}}; \text{vec}(\hat{\mathbf{A}})]$ for $\hat{\mathbf{A}} = (\mathbf{1}_{\mathbf{c} \neq 0}, \mathbf{1}^\top) \odot \mathbf{A}$. Then,

$$f(\boldsymbol{\theta}, \mathbf{x}) = \sum_{i=1}^h \mathbf{c}_i \phi(\mathbf{a}_i^\top \mathbf{x}) = \sum_{i=1}^h \mathbf{c}_i \phi(\mathbb{1}\{\mathbf{c}_i \neq 0\} \mathbf{a}_i^\top \mathbf{x}) = f(\hat{\boldsymbol{\theta}}, \mathbf{x}),$$

where the second equality follows from the fact that we can set $\mathbf{a}_i$ to be zero whenever $c_i = 0$ since that neuron does not contribute to the function output whenever $c_i = 0$. Further, changing $\mathbf{a}_i$ for any i where $c_i = 0$ does not change the number of neurons, since if for the i th neuron we have $c_i = 0$, then this neuron can never be active no matter the value of $\mathbf{a}_i$:

$$R(\boldsymbol{\theta}) = \sum_{i=1}^h \mathbb{1}\{|\mathbf{c}_i| \|\mathbf{a}_i\|_2 > 0\} = \sum_{i: \mathbf{c}_i \neq \mathbf{0}} \mathbb{1}\{\mathbf{a}_i \neq \mathbf{0}\} = \sum_{i: \mathbf{c}_i \neq \mathbf{0}} \mathbb{1}\{\hat{\mathbf{a}}_i \neq \mathbf{0}\} = R(\hat{\boldsymbol{\theta}}),$$

where $\hat{\mathbf{a}}_i^\top$ is the i th row of $\hat{\mathbf{A}}$. Lastly, since $\hat{\mathbf{a}}_i$ is always equal to $\mathbf{0}$ when $c_i = 0$, we must have that $\hat{\mathbf{A}}$ has at most $R(\hat{\boldsymbol{\theta}})$ number of nonzero rows.

E MinNorm-OG Algorithm

We derive the closed form solution of (7) for the specific choice $\tilde{R}(\boldsymbol{\theta} + \boldsymbol{\Delta}) = \|\boldsymbol{\theta} + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2$.

Define the span of the model gradients over $\mathcal{D}_r$ as the subspace $\mathcal{G}_r = \text{span}\{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \mathbf{x})\}_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_r}$ and consider any $\lambda \geq 0$. We then solve the following problem:

$$\tilde{\boldsymbol{\Delta}} = \underset{\boldsymbol{\Delta}}{\text{argmin}} \|\boldsymbol{\theta} + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2 \quad \text{s.t. } \boldsymbol{\Delta} \in \mathcal{G}_r^\perp. \quad (20)$$

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 (20).

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 4: Hyperparameter definitions and their associated methods.

Symbol	Methods	Description
T	All	Number of epochs
η	All	Learning rate
λ_{GA}	NGP, Scrub, SalUn	Loss ascent coefficient
λ_{reg}	NPO, Scrub, MinNorm-OG, Ridge, ℓ_1 -Sparse	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 Retrain and GD

Retrain and GD share the same loss function, but Retrain initializes a new model from scratch before optimizing this loss.

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

F.1.2 GA

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

F.1.3 NGD

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

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

F.1.4 NGP

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

F.1.5 Ridge

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

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

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

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

Note that 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 training loss minimizer.

F.1.6 ℓ_1 -Sparse

For each epoch $t = 1, \dots, T$, we define the ℓ_1 -Sparse loss as

$$\mathcal{J}_{\ell_1\text{-Sparse}}(\boldsymbol{\theta}; \mathcal{B}_r) = \mathcal{J}(\boldsymbol{\theta}; \mathcal{B}_r) + 2\left(1 - \frac{t-1}{T}\right) \lambda_{\text{reg}} \|\boldsymbol{\theta}\|_1.$$

This follows the linearly decaying regularization schedule proposed in [25].

F.1.7 Scrub

The Scrub loss decomposes into different terms depending on the epoch. Let $\pi_{\boldsymbol{\theta}}(\mathbf{y} | \mathbf{x})$ denote the model's predicted distribution over classes $\mathbf{y}$ for input $\mathbf{x}$ for parameter vector $\boldsymbol{\theta}$, and define $\text{KL}(\cdot \| \cdot)$ as the Kullback-Leiber divergence. Recall $\boldsymbol{\theta}^*$ denotes the initial trained model parameters, and denote the current epoch $t \in \{0, \dots, T-1\}$. 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:

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

F.1.8 NPO

Recall that θ^* denotes the initial trained model parameters. Then, the NPO loss is

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

where $\pi_{\theta}(y_f | \mathbf{x}_f)$ denotes the model's predicted probability of class y_f for input $\mathbf{x}_f$ for parameter vector θ . Note that this is equivalent to setting the parameter β in [18] to λ_{reg} .

F.1.9 SalUn

Before performing any unlearning, SalUn computes the median of the absolute values of the elements of the model gradient with respect to the loss over the forget set at the original parameter vector θ^* . Formally, define

$$\mathbf{g}_f = \text{abs}(\nabla_{\theta} \mathcal{J}(\theta^*; \mathcal{D}_f)),$$

where $\text{abs}(\cdot)$ denotes the element-wise absolute value. This step requires an initial pass over the forget set which we do not count toward the number of unlearning epochs T .

The "saliency mask" $\mathbf{m}_S$ is then defined as the binary vector that selects parameters whose corresponding gradient magnitudes exceed the median of $\mathbf{g}_f$, denoted $\text{median}(\mathbf{g}_f)$:

$$\mathbf{m}_S = \mathbb{1}\{\mathbf{g}_f > \text{median}(\mathbf{g}_f)\}.$$

After computing $\mathbf{m}_S$, SalUn minimizes the loss

$$\mathcal{J}_{\text{SalUn}}(\theta; \mathcal{B}_r, \mathcal{B}_f) = \mathcal{J}(\theta; \mathcal{B}_r) + \lambda_{\text{GA}} \mathcal{J}(\theta; \mathcal{B}'_f),$$

where $\mathcal{B}'_f$ is a modified version of the forget set batch $\mathcal{B}_f$ in which each true label is replaced with a random incorrect label. During unlearning, SALUN treats only the parameters where $\mathbf{m}_S = 1$ as trainable and freezes the others. Thus, the gradient update is applied only to $\mathbf{m}_S \odot \theta$.

F.1.10 MinNorm-OG

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

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

Following the AdamW update for this loss, we then (depending on the epoch) perform the model update corresponding to solving the relaxed unlearning problem (7) for $\tilde{R}(\theta + \Delta) = \|\theta + \Delta\|_2^2 + \lambda \|\Delta\|_2^2$, 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 smaller, 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 $\mathcal{G}'_r = \text{span}\{\nabla_{\theta} f(\theta, \mathbf{x})\}_{(\mathbf{x}, \mathbf{y}) \in \mathcal{B}'_r}$. As we showed in Appendix E, for any value of λ , the optimal perturbation is then $\tilde{\Delta} = -\frac{1}{1+\lambda} \mathcal{P}_{\mathcal{G}'_r^{\perp}}(\theta)$. Thus, the initial value $\lambda = \frac{1}{\lambda_{\text{reg}}} - 1$ leads to the perturbation $\tilde{\Delta} = -\lambda_{\text{reg}} \mathcal{P}_{\mathcal{G}'_r^{\perp}}(\theta)$. Further, the coefficient update $\lambda = \frac{\lambda'+1}{\gamma_{\text{reg}}} - 1$ leads to a more conservative unlearning perturbation $\tilde{\Delta} = -\gamma_{\text{reg}} \frac{1}{1+\lambda'} \mathcal{P}_{\mathcal{G}'_r^{\perp}}(\theta)$, as it is down-weighted by γ_{reg} . Thus, λ_{reg} is the initial strength of the perturbation and γ_{reg} represents a multiplicative decay of this strength through each update to λ .

Table 5: Data Poisoning experiment results showing the median sup-norm distance between the retain set trend $y = \sin(x)$ and the unlearned model outputs across 10 trials (smaller is better). The parentheses indicate the central range of values over the 10 trials.

Epochs	Retrain	MinNorm-OG	GD	GA	NGP	NGD	Ridge	ℓ_1 -Sparse
10	1.50 (1.34, 1.85)	1.50 (1.45, 2.96)	3.23 (2.52, 6.65)	2.56 (2.19, 3.45)	2.50 (2.20, 3.94)	2.73 (2.26, 7.23)	2.27 (1.90, 3.02)	3.28 (2.52, 3.62)
100	1.36 (1.27, 1.44)	1.08 (1.00, 1.63)	2.76 (2.48, 3.54)	23.8 (18.1, 30.0)	2.62 (2.31, 7.87)	2.85 (2.47, 3.57)	2.13 (1.75, 3.12)	1.79 (1.54, 2.48)
1000	1.17 (1.08, 1.26)	0.63 (0.42, 0.96)	2.61 (2.39, 3.38)	1400 (1025, 1869)	2.75 (2.07, 9.80)	2.45 (1.96, 3.67)	1.96 (1.61, 3.84)	1.37 (1.19, 1.55)

We formally write the unlearning update at epoch t as follows, where θ_0 is the current parameter vector, θ_{new} is the updated vector, and mod denotes the modulo operation.

$$\begin{aligned}
 & \text{if } t \bmod T_{\text{Proj}} \neq 0 \text{ or } t \geq T - T_{\text{GD}} \\
 & \quad \theta_{\text{new}} = \theta_0 \\
 & \text{else} \\
 & \quad \tilde{\Delta} = \underset{\Delta \in \mathcal{G}'_r{}^\perp}{\operatorname{argmin}} \|\theta_0 + \Delta\|_2^2 + \lambda \|\Delta\|_2^2 \\
 & \quad \theta_{\text{new}} = \theta_0 + \tilde{\Delta} \\
 & \quad \lambda \leftarrow \frac{\lambda + 1}{\gamma_{\text{reg}}} - 1
 \end{aligned}$$

Gradients for Classification. We make a special note of how we compute the gradient sub-space $\mathcal{G}'_r$ for classification tasks. At the parameter value θ_0 , the model prediction is $f(\theta_0, \mathbf{x}) = \operatorname{argmax} \mathbf{z}_{\theta_0}(\mathbf{y} | \mathbf{x})$ where $\mathbf{z}_{\theta}(\mathbf{y} | \mathbf{x})$ denotes the model’s unnormalized logits over the classes $\mathbf{y}$ for input $\mathbf{x}$ for parameter vector θ . This is not a continuous function of θ , so we cannot compute its gradient directly. However, following prior works [19], we use the gradient $\nabla_{\theta} (\mathbf{z}_{\theta_0}(\mathbf{y} | \mathbf{x}))_j$, where $j = f(\theta_0, \mathbf{x})$ is the model’s predicted class for input $\mathbf{x}$. 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 θ .

F.2 Data Poisoning

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 θ of each unlearning method by measuring the deviation from the retain set trend, given by $\sup_{\mathbf{x} \in \mathcal{X}} |f(\theta, \mathbf{x}) - \sin(\mathbf{x})|$. 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 10 trials. For the number of unlearning epochs $T \in \{10, 100, 1000\}$, we report the best performance of each algorithm in Table 5 along with the central range of each method’s performance over the 10 trials, discarding the two best and worst trials. We report the best hyperparameters for each method in Table 6, and the corresponding search spaces in Table 7. We also include visualizations of the recovered models from each unlearning method in Figures 3, 4, and 5. All experiments were run on either a single NVIDIA A40 GPU or a single NVIDIA H200 GPU.

Table 6: Hyperparameter settings for each entry in Table 5. 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	Retrain	1e-4							
	MinNorm-OG	1e-3		.1		2	.9	1	50
	GD	1e-4							
	GA	1e-4							
	NGP	1e-4	1.0						
	NGD	1e-2			.5				
	Ridge	1e-2		3.0			.6		
	ℓ_1 -Sparse	1e-2		.1					
100	Retrain	1e-4							
	MinNorm-OG	1e-3		.1		50	.9	2	50
	GD	1e-3							
	GA	1e-4							
	NGP	5e-4	.01						
	NGD	1e-3			.1				
	Ridge	1e-3		3.0			.9		
	ℓ_1 -Sparse	1e-3		.1					
1000	Retrain	1e-4							
	MinNorm-OG	1e-3		.1		500	.9	10	50
	GD	1e-3							
	GA	1e-4							
	NGP	1e-3	.001						
	NGD	1e-3			1.0				
	Ridge	1e-3		3.0			.9		
	ℓ_1 -Sparse	1e-3		.1					

Table 7: Hyperparameter values tested in the experiments corresponding to Table 5.

Epochs	η	σ	λ_{reg}	γ_{reg}	λ_{GA}	T_{GD}	T_{Proj}	n_{pert}
10	{1e-2, 1e-3, 1e-4}	{0.1, 0.5, 1.0}	{0.1, 1.0, 3.0}	{0.3, 0.6, 0.9}	{0.001, 0.01, 0.1, 1.0}	{1, 2, 5}	{1, 2, 5}	{50}
100	{1e-3, 5e-4, 1e-4}	{0.1, 0.5, 1.0}	{0.1, 1.0, 3.0}	{0.3, 0.6, 0.9}	{0.001, 0.01, 0.1, 1.0}	{25, 50}	{1, 2}	{50}
1000	{1e-3, 5e-4, 1e-4}	{0.1, 0.5, 1.0}	{0.1, 1.0, 3.0}	{0.3, 0.6, 0.9}	{0.001, 0.01, 0.1, 1.0}	{100, 200, 500}	{10, 50, 100}	{50}

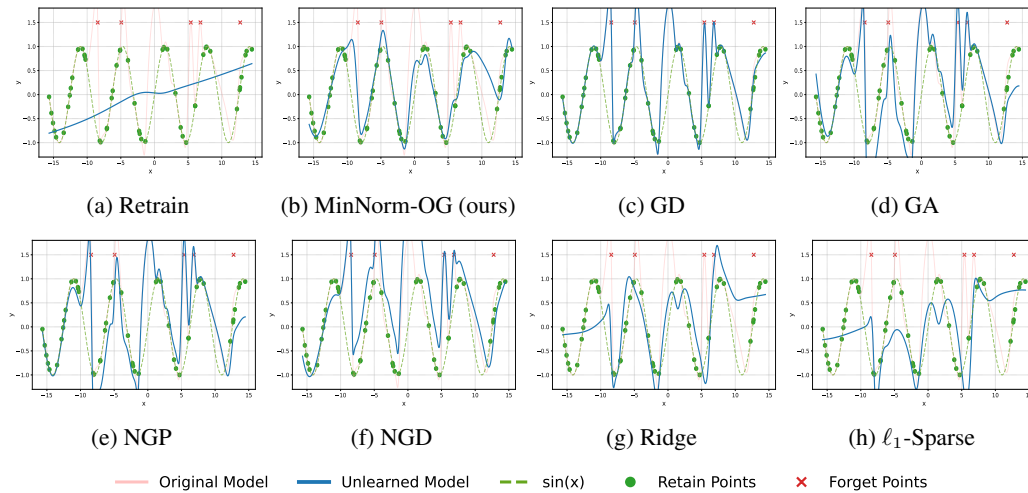


Figure 3: Example unlearned model fits when given 10 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

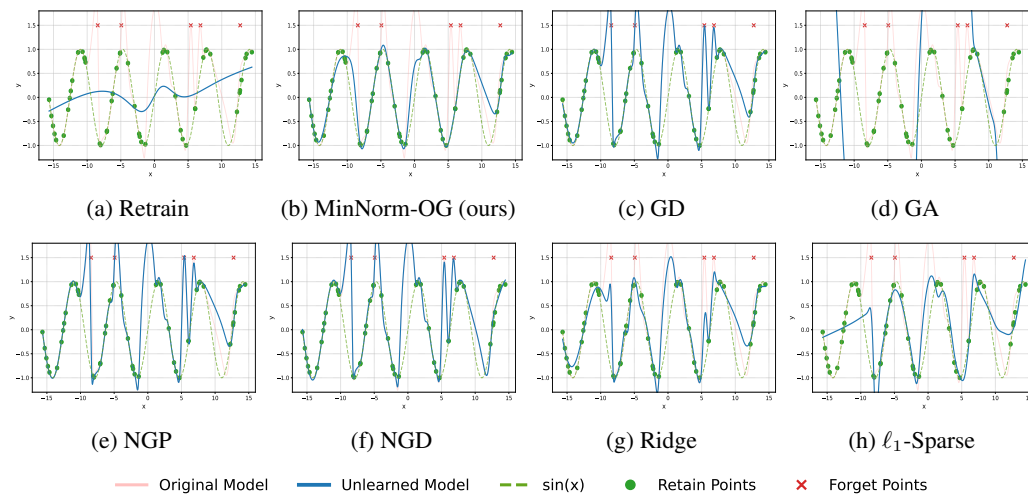


Figure 4: Example unlearned model fits when given 100 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

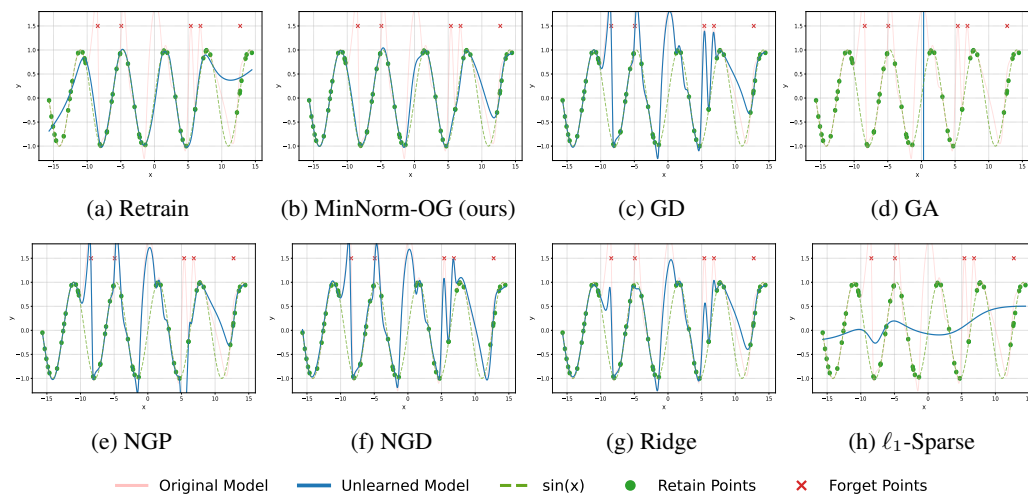


Figure 5: Example unlearned model fits when given 1000 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

E.3 Multi-Class Label Erasure

We use the CIFAR-10 [30] and Tiny ImageNet [31] datasets, creating red, green, and gray copies of each image in the training sets. The retain set consists of all gray images, while the forget set is formed by randomly sampling a proportion $p_{\text{color}} \in [0, 1]$ of the red and green copies. We then train modified ResNet-18 and ResNet-50 models [32] on CIFAR-10 and Tiny ImageNet, respectively, to jointly predict image class and color. Each model includes two separate prediction heads, one for image class and one for color.

For CIFAR-10, we train for 100 epochs using the SGD optimizer with an initial learning rate of 3×10^{-2} , weight decay of 5×10^{-4} , momentum of 0.9, and batch size of 256. The learning rate is reduced to 3×10^{-3} at epoch 50. The ground-truth unlearned model is trained on the gray images alone using the same parameters.

For Tiny ImageNet, we initialize the ResNet-50 architecture with ImageNet-pretrained weights [33] from the `torchvision` library. We apply standard data augmentations to reduce overfitting and train for 100 epochs using a batch size of 512, initial learning rate of 0.1, weight decay of 10^{-4} , and momentum of 0.9. The learning rate is decayed by a factor of 0.3 every 25 epochs. During the first 10 epochs, we update the model only using the class prediction loss to adapt the pretrained weights to the colored-image domain. For the remaining epochs, we optimize both the class and color prediction losses. The ground-truth unlearned model is trained on the gray images alone using the same 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 measure retain quality as accuracy on gray-colored test samples, and forget quality error 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 corresponding Pareto frontier across the two metrics, where the optimal point at $(1, 0)$ which indicates perfect retain quality and zero forget quality error. Each point in the frontier for a given method represents the mean results over 5 trials of a single hyperparameter combination, with error bars representing the standard error for each metric. We label the performance of the ground truth unlearned model as GT. All training and parameter searches were performed on a cluster of NVIDIA GH200 GPUs.

Table 8: Hyperparameter values tested for the results in Figure 6 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$ and $T = 2$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{0\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

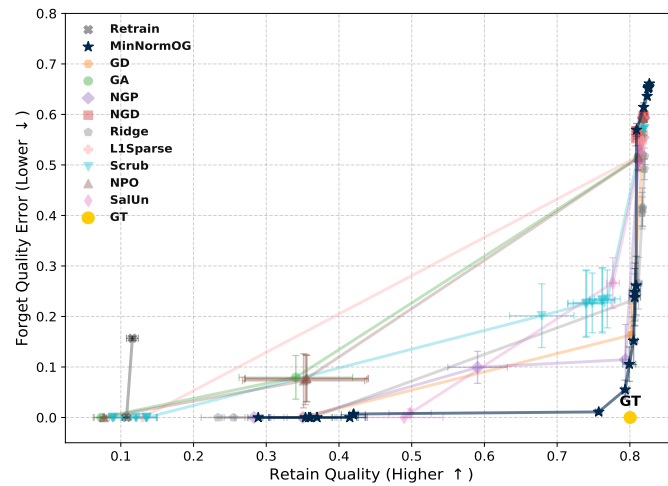


Figure 6: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$ and $T = 2$. This is an enlarged version of the left subfigure in Figure 2 with added error bars.

Table 9: Hyperparameter values tested for the results in Figure 7 running the Multi-Label Class Erasure experiment on Tiny ImageNet with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$ and $T = 5$.

Hyperparameter	Sweep Values
η	$\{5 \times 10^{-4}, 10^{-4}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{2, 3\}$
γ_{reg}	$\{0.2, 0.4, 0.8, 0.9\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

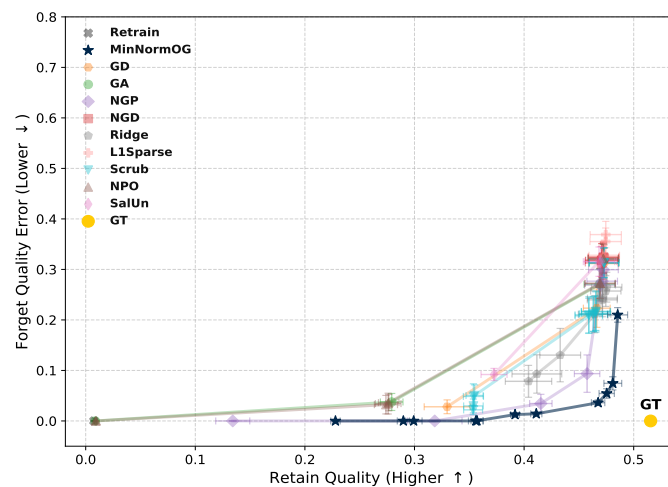


Figure 7: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on Tiny ImageNet with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$ and $T = 5$. This is an enlarged version of the right subfigure in Figure 2 with added error bars.

Table 10: Hyperparameter values tested for the results in Figure 8 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$ and $T = 1$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{0\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

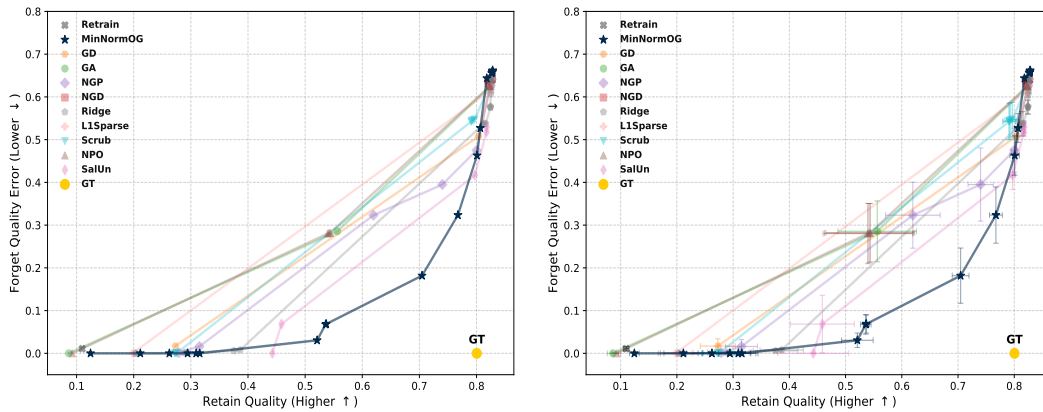


Figure 8: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$ and $T = 1$. The left panel omits error bars for visual clarity.

Table 11: Hyperparameter values tested for the results in Figure 9 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .001$ and $T = 2$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{0\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

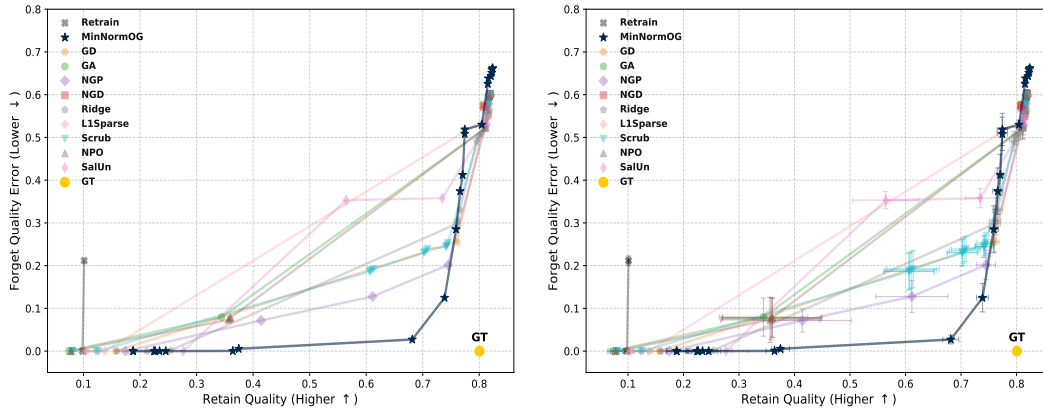


Figure 9: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .001$ and $T = 2$. The left panel omits error bars for visual clarity.

Table 12: Hyperparameter values tested for the results in Figure 10 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .001$ and $T = 5$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 5 \times 10^{-4}, 10^{-4}, 5 \times 10^{-5}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{1, 2, 3\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

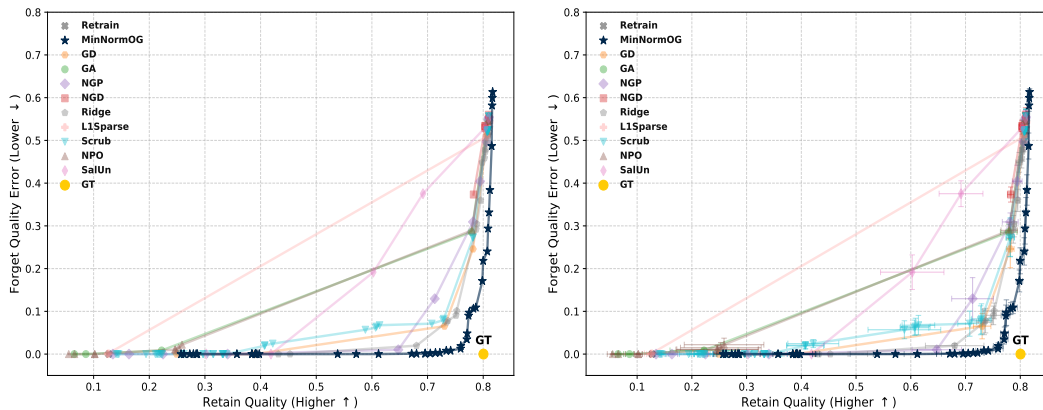


Figure 10: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{color}} = .01$, $p_{\text{retain}} = .001$ and $T = 5$. The left panel omits error bars for visual clarity.

Table 13: Hyperparameter values tested for the results in Figure 11 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{color}} = .001$, $p_{\text{retain}} = .001$ and $T = 10$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{0, 1, 2, 5\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

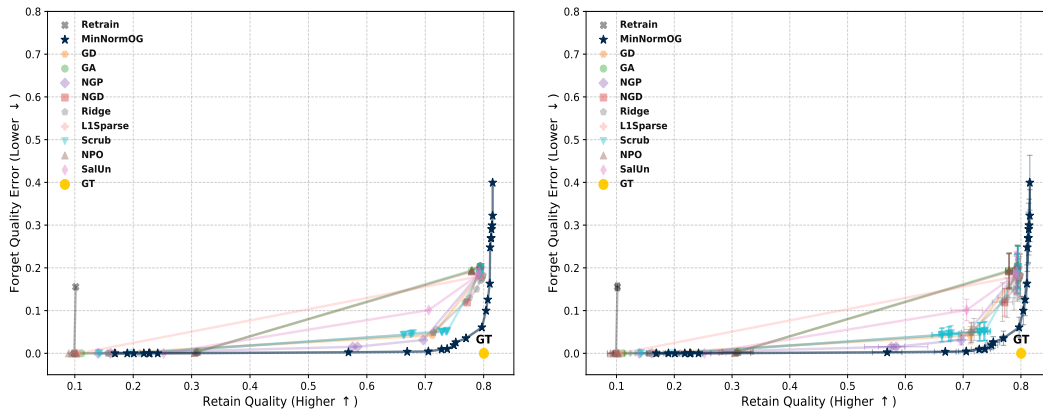


Figure 11: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{color}} = .001$, $p_{\text{retain}} = .001$, and $T = 10$. The left panel omits error bars for visual clarity.

Table 14: Hyperparameter values tested for the results in Figure 12 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{color}} = .001$, $p_{\text{retain}} = .01$ and $T = 10$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{0, 1, 2, 5\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

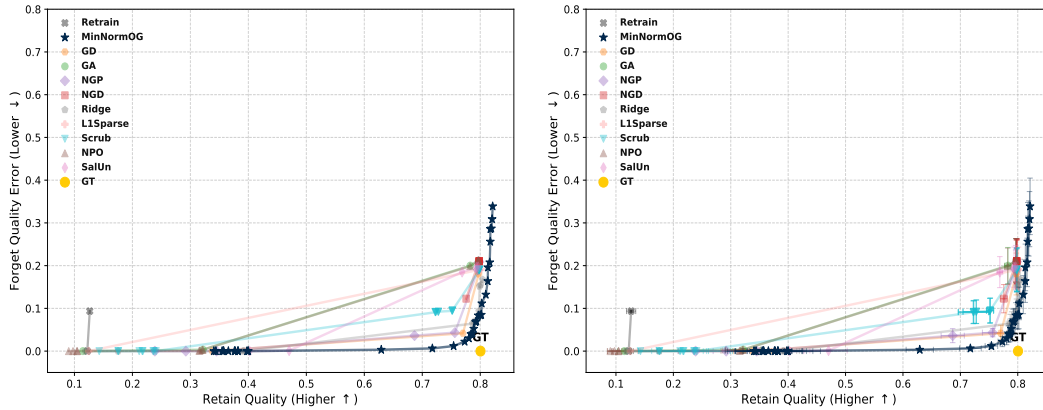


Figure 12: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{color}} = .001$, $p_{\text{retain}} = .01$, and $T = 10$. The left panel omits error bars for visual clarity.

Table 15: Hyperparameter values tested for the results in Figure 13 running the Multi-Label Class Erasure experiment on Tiny ImageNet with $p_{\text{color}} = .01$, $p_{\text{retain}} = .001$ and $T = 10$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{0, 1, 2\}$
γ_{reg}	$\{0.2, 0.4, 0.8, 0.9\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

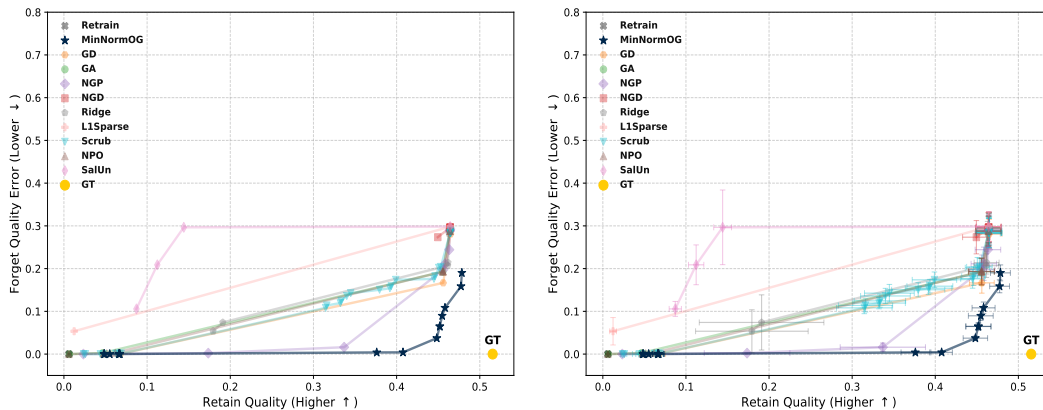


Figure 13: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on Tiny ImageNet with $p_{\text{color}} = .01$, $p_{\text{retain}} = .001$, and $T = 10$. The left panel omits error bars for visual clarity.

Table 16: Hyperparameter values tested for the results in Figure 14 running the Multi-Label Class Erasure experiment on Tiny ImageNet with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$ and $T = 10$.

Hyperparameter	Sweep Values
η	$\{10^{-3}, 10^{-4}, 10^{-5}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{10^{-3}, 10^{-2}, 5 \times 10^{-2}, 10^{-1}\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{0, 1, 2\}$
γ_{reg}	$\{0.2, 0.4, 0.8, 0.9\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

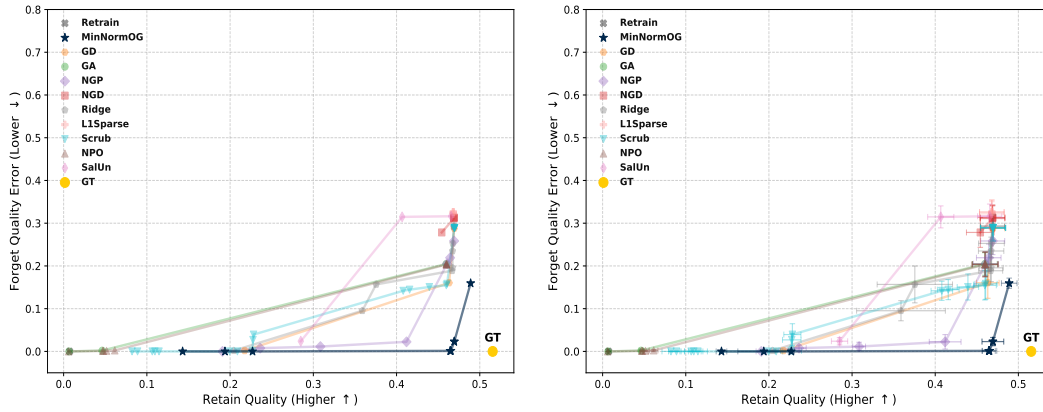


Figure 14: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on Tiny ImageNet with $p_{\text{color}} = .01$, $p_{\text{retain}} = .01$, and $T = 10$. The left panel omits error bars for visual clarity.

F.4 Representation Collapse

We again use the CIFAR-10 dataset with a modified ResNet-18 architecture, assigning each of the 10 image classes a unique color. The retain set $\mathcal{D}_r$ consists of images colored according to their assigned class color, while the forget set $\mathcal{D}_f$ contains randomly colored images. The ground-truth unlearned model predicts class labels based solely on color, as color perfectly determines the label and is easier to learn than image content. In contrast, models trained on the full dataset $\mathcal{D} = \mathcal{D}_r \sqcup \mathcal{D}_f$ must rely on content features, since color is no longer fully predictive of the label. Thus, for unlearning evaluation, we label heldout test 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.

To construct $\mathcal{D}_f$, we randomly select 1% of the training images per class and assign each a random color chosen from the remaining nine class colors. For the initial model, we first warm-start by training for 20 epochs on gray images (a color not used in the class assignments) to encourage learning general image structure. We then train for 50 epochs on the combined retain and forget sets, up-weighting the loss contribution of off-colored samples to promote fast color-invariant learning. We used the SGD optimizer with an initial learning rate of 3×10^{-2} , weight decay of 5×10^{-4} , momentum of 0.9, a batch size of 256, and a multiplicative learning rate decay of 0.1 every 20 epochs. The ground-truth unlearned model is trained for 50 epochs on the retain set using the same optimizer settings. All training was performed on a cluster of NVIDIA H200 GPUs.

We report results mean results over 5 trials in Table 17, which presents the same results as Table 2 along with standard errors for each entry. We test each method under different constraints on the number of unlearning epochs and percentage of accessible retain set samples. Specifically, for each random trial we randomly sample a proportion of $p_{\text{retain}} \in [0, 1]$ samples from $\mathcal{D}_r$ which is accessible to each unlearning method. The "Retain %" column Table 17 represents $100 \times p_{\text{retain}}$. Just as in the Multi-Class Label Erasure experiment, during each unlearning epoch the algorithms iterate over

Table 17: Representation Collapse experiment results across constraints on the number of unlearning epochs and percentage of accessible retain set samples. Models are trained on colored images where color perfectly predicts the label in the retain set but not in the full dataset $\mathcal{D}$. Reported values are mean test accuracies (%) on test images labeled by color (higher is better), averaged over 5 trials $\pm$ standard error.

Retain %	Epochs	Retrain	MinNorm-OG	GD	GA	NGP	NGD	Ridge	ℓ_1 -Sparse	Scrub	NPO	SalUn
0.1	5	79.1 $\pm$ 4.5	49.1 $\pm$ 6.1	24.0 $\pm$ 3.2	34.5 $\pm$ 4.4	45.9 $\pm$ 3.4	12.6 $\pm$ 0.2	20.9 $\pm$ 3.7	12.4 $\pm$ 0.3	37.3 $\pm$ 3.7	40.6 $\pm$ 3.7	29.4 $\pm$ 5.5
	10	95.5 $\pm$ 1.1	78.7 $\pm$ 3.4	55.2 $\pm$ 4.3	38.3 $\pm$ 5.4	73.7 $\pm$ 3.8	33.0 $\pm$ 2.2	60.0 $\pm$ 3.7	23.0 $\pm$ 6.5	61.9 $\pm$ 3.4	43.2 $\pm$ 6.9	53.4 $\pm$ 7.9
	15	96.3 $\pm$ 0.8	93.2 $\pm$ 1.3	78.2 $\pm$ 3.3	39.8 $\pm$ 5.1	81.5 $\pm$ 1.6	46.9 $\pm$ 3.4	81.3 $\pm$ 2.7	25.1 $\pm$ 4.7	74.1 $\pm$ 3.9	44.6 $\pm$ 7.6	76.3 $\pm$ 5.3
1	5	92.8 $\pm$ 2.6	59.5 $\pm$ 6.1	40.7 $\pm$ 2.4	34.2 $\pm$ 4.7	58.3 $\pm$ 6.0	32.9 $\pm$ 3.6	33.5 $\pm$ 6.1	12.5 $\pm$ 0.2	47.8 $\pm$ 4.9	42.5 $\pm$ 4.1	42.5 $\pm$ 2.7
	10	98.5 $\pm$ 0.1	94.7 $\pm$ 1.1	73.6 $\pm$ 3.0	38.2 $\pm$ 5.4	92.4 $\pm$ 2.5	70.5 $\pm$ 3.2	63.4 $\pm$ 5.0	31.4 $\pm$ 6.6	75.8 $\pm$ 2.8	43.5 $\pm$ 6.9	68.7 $\pm$ 4.0

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.

Retrain achieves much higher color classification accuracy than any unlearning method, as image color is an easy feature to learn from scratch. This result is specific to this setup, as our prior experiments demonstrate that Retrain is not a viable unlearning strategy in general. For this reason, we report the Retrain results separately in the leftmost column and highlight the MinNorm-OG results in bold, as they represent the best performance among the unlearning methods in each row.

For each constraint setting on the number of epochs and the Retain %, we tested different hyperparameters before reporting the best performance for each method. Tables 18, 19, 20, 21, and 22 report the set of hyperparameter values we tested for each row of Table 17.

Table 18: Hyperparameter values considered for the Representation Collapse Experiment with $T = 5$ and $p_{\text{retain}} = 0.001$.

Hyperparameter	Values
η	$\{10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{0.1, 0.2, 0.5, 0.9\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{1, 2, 3\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

Table 19: Hyperparameter values considered for the Representation Collapse Experiment with $T = 10$ and $p_{\text{retain}} = 0.001$.

Hyperparameter	Values
η	$\{5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{0.01, 0.1, 0.5, 0.9\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{3, 5, 7\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

Table 20: Hyperparameter values considered for the Representation Collapse Experiment with $T = 15$ and $p_{\text{retain}} = 0.001$.

Hyperparameter	Values
η	$\{5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{0.01, 0.1, 0.5, 0.9\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{3, 5, 7\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

Table 21: Hyperparameter values considered for the Representation Collapse Experiment with $T = 5$ and $p_{\text{retain}} = 0.01$.

Hyperparameter	Values
η	$\{5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{0.01, 0.1, 0.5, 0.9\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{1, 2, 3\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

Table 22: Hyperparameter values considered for the Representation Collapse Experiment with $T = 10$ and $p_{\text{retain}} = 0.01$.

Hyperparameter	Values
η	$\{5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 10^{-4}\}$
λ_{GA}	$\{10^{-3}, 10^{-2}, 10^{-1}\}$
λ_{reg}	$\{0.01, 0.1, 0.5, 0.9\}$
σ	$\{0.1, 1.0, 10.0\}$
T_{GD}	$\{3, 5, 7\}$
γ_{reg}	$\{0.2, 0.4, 0.8\}$
T_{Proj}	$\{1\}$
n_{pert}	$\{50\}$

F.5 Asset Information

We use the CIFAR-10 [30] and Tiny ImageNet [31] datasets in our experiments. Both datasets are publicly available but do not specify an explicit license. Additionally, we use the ResNet-18 and ResNet-50 [32] architectures and pretrained weights from PyTorch’s `torchvision` library, which are licensed under the BSD 3-Clause License.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Yes, our main claim centers on the failure of prior definitions and algorithms for unlearning when applied in the overparameterized setting, which we show in Section 2, and the theoretical guarantees of our new framework and algorithm, which we show in Section 4. We clearly state the model classes we analyze (linear models, linear networks, and 2-layer perceptrons) and we provide specific guarantees for each. We lastly provide strong empirical results in Section 5, supporting our claim that our proposed method can outperform prior methods in practice.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We explicitly state the model classes we analyze as well as the exact guarantees for our theoretical results in Section 4. When we describe our theoretical contributions in the Introduction, we mention the specific model classes and the corresponding results we achieve for each of them. Empirically, we discuss the computational efficiency of our proposed method in Algorithm 1 relative to other gradient-descent-based methods which we compare against. In the conclusion, we mention future work includes extending our analysis to more complex model classes and performing experiments in broader domains at larger scale.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: For each statement of theorems, propositions, and lemmas, we explicitly state the assumptions we rely on. The proofs of all statements are in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We explain the experiments in the main body, and we include a further discussion in the appendix. Also, all code for generating our empirical results is publicly released.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

- (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
- (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
- (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
- (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: Code is publicly released and the datasets we use (Tiny ImageNet, CIFAR-10) are publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: We describe the experimental methodology for each setting in the main paper, while full experimental details are reported in Appendix F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For our experiments, we either report means along with standard errors or medians with ranges of values across trials. For clarity, the main text presents only the central values (means or medians), while the appendix includes full results with corresponding uncertainty estimates.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report this in Appendix F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: This paper meets all ethical guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This work helps develop theory and algorithms for unlearning in the overparameterized setting, which has no direct societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper does not release any new data or models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: We cite the datasets we use in the main body, and include license information in the appendix.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: We release code for reproducing our experiments which contains documentation.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: We do not use human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [NA]

Justification: LLMs were not used for any important, original, or core components of this paper.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.