Machine Unlearning under Overparameterization

Jacob L. Block* Aryan Mokhtari* Sanjay Shakkottai*

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 unlearning 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 we demonstrate that an implementation of our framework outperforms existing baselines across various unlearning experiments.

*Chandra Family Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA. {jblock@utexas.edu, mokhtari@austin.utexas.edu, sanjay.shakkottai@utexas.edu}

1 Introduction

As modern models are trained on vast datasets, the ability to remove the influence of specific data samples from a trained model is essential—both to comply with privacy regulations such as the GDPR and CCPA [GDP16; CCP18], and to correct mislabeled or biased data that may compromise model integrity [GRBTKDYZA24]. *Machine unlearning* [CY15] 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 behaves 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" solution to unlearning is to retrain the model from scratch on the retain set $\mathcal{D}_r$, which perfectly achieves the unlearning objective but is often infeasible due to high computational cost and the potential for limited access to the original training data. Hence, the goal of an unlearning algorithm 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 the exact unlearning solution 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 $\mathcal{D}$ [BNLGG22; SAKS21; GGHV20].

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 $\boldsymbol{\theta}^*$ minimizes the loss over both $\mathcal{D}$ and $\mathcal{D}_r$, and $\boldsymbol{\theta}^*$ 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 2.1). 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 performance 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 loss minimization constraint reduces to requiring interpolation of $\mathcal{D}_r$. To solve the resulting problem of minimizing R subject to interpolation, we relax the constraint via a first-order Taylor expansion around $\boldsymbol{\theta}^*$ and reparameterize as $\boldsymbol{\theta}^* + \boldsymbol{\Delta}$, where $\boldsymbol{\Delta}$ is the drift. Since $\boldsymbol{\theta}^*$ already interpolates $\mathcal{D}_r$, the linearized constraint requires $\boldsymbol{\Delta}$ to be orthogonal to model gradients at $\boldsymbol{\theta}^*$ 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 $\hat{R}(\boldsymbol{\Delta})$ to control the size and direction of the drift. The final objective minimizes $R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) + \hat{R}(\boldsymbol{\Delta})$ under the relaxed orthogonal gradient constraint, yielding updated parameters $\boldsymbol{\theta}^* + \boldsymbol{\Delta}$.

Theoretical Contributions. For linear models and linear networks, we prove there exists a regularizer $\hat{R}$ such that minimizing $R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) + \hat{R}(\boldsymbol{\Delta})$ over our constraint relaxation gives the

exact unlearning solution when R is the ℓ_2-norm of either the effective linear predictor 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 [KTHT23; PDLKSN25; MFSLK24], where OG refers to orthogonal gradient. MinNorm-OG alternates between two steps: solving for the minimizer of $R(\boldsymbol{\theta}+\boldsymbol{\Delta})+\hat{R}(\boldsymbol{\Delta})$ over $\boldsymbol{\Delta}$ 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 [Ham74], a classic statistical tool for estimating the effect of down-weighting a sample on a learned function [BNLGG22]. Extensions have explored approximate unlearning via differential privacy [SAKS21; GGHV20]. Previous works have considered different unlearning paradigms. [SAKS21] analyzed the deletion capacity an unlearning method can tolerate while maintaining adequate generalization performance. [GKKMSZ23; BCCJTZLP21] 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 [NRS21; GNG21; CS23; JBVRCCH25; ZLBM24]. All these methods rely on loss gradient perturbations, which we show yield vacuous updates under overparameterization (Theorem 2.1). In practice, they also struggle to unlearn effectively [PDLKSN25], as loss ascent encourages misfitting $\mathcal{D}_f$ rather than forgetting it.

Our framework builds on components from other contexts. We enforce parameter perturbations to be orthogonal to the gradient of the model's predictions on $\mathcal{D}_r$ to preserve loss optimality—an idea also used in continual learning to retain past performance [FAML20]. Recent unlearning methods use similar projections which mix loss ascent and descent, but their reliance on these objectives inherits prior limitations [CZYZ24; HRGV24].

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

2 Unlearning in Overparameterized Settings

We introduce notation 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 $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, with sample inputs $\boldsymbol{x}_i \in \mathbb{R}^m$ and outputs $\boldsymbol{y}_i \in \mathbb{R}^l$ 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 $\{f(\boldsymbol{\theta}, \cdot) \mid \boldsymbol{\theta} \in \mathbb{R}^d\}$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$, where $f : \mathbb{R}^{d+m} \to \mathbb{R}^l$ takes a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ and an input $\boldsymbol{x} \in \mathbb{R}^m$ and maps them to a prediction $f(\boldsymbol{\theta}, \boldsymbol{x})$ in the output space $\mathbb{R}^l$. We define the training procedure, also denoted the learning algorithm, as $\mathcal{A} : 2^{\mathcal{Z}} \to \mathbb{R}^d$, which takes in a dataset and returns the parameter vector $\boldsymbol{\theta}^*$ corresponding to the trained model. We make the minimal assumption that $\mathcal{A}$ is faithful to a known loss function $\mathcal{J}$, meaning $\mathcal{A}(\mathcal{D}) = \boldsymbol{\theta}^*$ 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 $\mathcal{L}$:

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

For our theoretical discussion, we consider sample-wise loss functions $\mathcal{L}\left(\boldsymbol{\theta}\,; \boldsymbol{x}, \boldsymbol{y}\right)$ which are minimized when $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{y}$, 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 $M\left(\mathcal{A}, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f\right)$ 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 $\boldsymbol{\theta}_r^*$, where the subscript r indicates that $\boldsymbol{\theta}_r^*$ 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 $\boldsymbol{\theta}_r^*$, 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 $\mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{D}\right)$ as well as the retain set $\mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{D}_r\right)$ admits a unique minimizer. To ensure that the unlearning solution remains consistent with the training loss, the only valid choice is to define $\boldsymbol{\theta}_r^*$ as the unique minimizer of $\mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{D}_r\right)$. However, in the *overparameterized* setting this uniqueness property fails to hold, as both $\mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{D}\right)$ and $\mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{D}_r\right)$ may admit multiple minimizers. In order to sidestep the non-uniqueness issue, one may be tempted to define *any* minimizer of $\mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{D}_r\right)$ as a valid unlearning solution, as presumably any minimizer to $\mathcal{J}\left(\boldsymbol{\theta}\,; \mathcal{D}_r\right)$ 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 $\boldsymbol{\theta}^*$ that interpolates all of $\mathcal{D}$ is itself a valid minimizer of the retain set loss $\mathcal{J}\left(\boldsymbol{\theta}; \mathcal{D}_r\right)$, but $\boldsymbol{\theta}^*$ can reflect training dynamics influenced by $\mathcal{D}_f$, revealing information that cannot be inferred from $\mathcal{D}_r$ alone (see Appendix B 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 $R(\boldsymbol{\theta})$, expressed as the output of a training algorithm $\mathcal{A}_R$:

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

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 $\boldsymbol{\theta}_r^*$ 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 [HMRT22].

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 $M\left(\mathcal{A}_R, \mathcal{I}_r, \mathcal{A}(\mathcal{D}), \mathcal{D}_f\right)$ 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 [BNLGG22; SAKS21; GGHV20], while existing empirical unlearning methods perform combinations of loss ascent over $\mathcal{D}_f$, loss descent over $\mathcal{D}_r$, and parameter noising [NRS21; GNG21; CS23; KTHT23]. We characterize these methods as *loss-gradient unlearning*, and show that they perform ineffective updates when deployed under overparameterization.

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

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

Although versions of loss-gradient unlearning have been theoretically motivated in the underparameterized setting [SAKS21; GGHV20], we show they fail to unlearn in the overparameterized setting.

Theorem 2.1. *Let $f(\boldsymbol{\theta}^*, \cdot)$ interpolate $\mathcal{D}$, so $f(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{y}$ for all $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$, and let M_{LG} be any loss-gradient unlearning method. If the sample loss $\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}\right)$ is minimized when $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{y}$, then for all $\mathcal{D}_f \subseteq \mathcal{D}$, M_{LG} simply noises $\boldsymbol{\theta}^*$ by some zero-mean random variable $\boldsymbol{\xi}$.*

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

The recovered parameters $\boldsymbol{\theta}^*$ already minimize $\mathcal{J}\left(\boldsymbol{\theta}^*; \mathcal{D}_r\right)$, so the loss gradients vanish and M_{LG} merely adds noise to $\boldsymbol{\theta}^*$. This shows the core issue with loss gradient updates in overparameterized unlearning: the loss gradient is uninformative, as both $\boldsymbol{\theta}^*$ and $\boldsymbol{\theta}_r^*$ 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 $f(\boldsymbol{\theta}, \cdot)$ where $f(\boldsymbol{\theta}, \boldsymbol{x}_i) = \boldsymbol{y}_i$ for every $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ in the retain set. This lets us replace the loss minimization in (2) with the hard constraint $f(\boldsymbol{\theta}, \boldsymbol{x}_i) = \boldsymbol{y}_i$, leading to the following formulation:

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

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 $\boldsymbol{\theta}^*$ 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 $\boldsymbol{\theta}^*$. While the constraints $f(\boldsymbol{\theta}, \boldsymbol{x}_i) = \boldsymbol{y}_i$ in (4) can be highly nonconvex and difficult to satisfy in general, we demonstrate that using the proposed first-order approximation

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

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

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

where for notational convenience, we define the drift variable as $\boldsymbol{\Delta} = \boldsymbol{\theta} - \boldsymbol{\theta}^*$. 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 $\boldsymbol{\Delta}$ 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 $\hat{R}(\boldsymbol{\Delta})$ to the loss to control the drift. The resulting objective function is $\tilde{R}(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) := R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) + \hat{R}(\boldsymbol{\Delta})$. Consequently, the optimization problem we propose to solve instead of (4) is given by

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

Indeed, by finding $\tilde{\boldsymbol{\Delta}}$ the suggested unlearned model would be $\boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$. 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 $\mathcal{I}_r = \{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^*, \boldsymbol{x})\}_{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_r}$. This is significantly less restrictive than prior work, which requires access to the inverse Hessian of the loss over $\mathcal{D}_r$ [BNLGG22; SAKS21; GGHV20], 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). For clarity, we denote the Euclidean projection onto a set $\boldsymbol{S}$ by $\mathcal{P}_{\boldsymbol{S}}(\cdot)$, and we define the penalty function $\delta_{\{a\}}$, which is $+\infty$ if condition a is satisfied and 0 otherwise.

4.1 Linear Model

We first consider training a linear model $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$ on data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ where $\boldsymbol{x}_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$. Given initial parameters $\boldsymbol{\theta}^*$ with $\boldsymbol{\theta}^{*\top} \boldsymbol{x}_i = y_i$ for all $(\boldsymbol{x}_i, y_i) \in \mathcal{D}$, we can easily solve the exact unlearning problem (4) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$.

Theorem 4.1. *Let $\tilde{\boldsymbol{\Delta}}$ solve (7) for $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$ and $\tilde{R}(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$. Then the recovered solution $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^* + \tilde{\boldsymbol{\Delta}}$ solves the exact unlearning problem (4) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$*

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 $f(\boldsymbol{\theta}, \boldsymbol{x}) = \boldsymbol{c}^\top \boldsymbol{A}_{L-1} \cdots \boldsymbol{A}_1 \boldsymbol{x}$, where the parameter vector is partitioned $\boldsymbol{\theta} = [\boldsymbol{c}; \mathrm{vec}(\boldsymbol{A}_1); \ldots; \mathrm{vec}(\boldsymbol{A}_{L-1})]$, with $\boldsymbol{A}_\ell \in \mathbb{R}^{h_\ell \times h_{\ell-1}}$ and $\boldsymbol{c} \in \mathbb{R}^{h_{L-1}}$ for $\ell = 1, \ldots, L-1$. The input dimension is $m = h_0$, and we assume $n < m$ to reflect the overparameterized regime. For clarity, define the effective linear predictor $\mathbf{w}(\boldsymbol{\theta}) = \boldsymbol{A}_1^\top \cdots \boldsymbol{A}_{L-1}^\top \boldsymbol{c}$, so that $f(\boldsymbol{\theta}, \boldsymbol{x}) = \mathbf{w}(\boldsymbol{\theta})^\top \boldsymbol{x}$. 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: $R(\boldsymbol{\theta}) = \left\|\boldsymbol{A}_1^\top \cdots \boldsymbol{A}_{L-1}^\top \boldsymbol{c}\right\|_2 = \|\mathbf{w}(\boldsymbol{\theta})\|_2$. Given $\boldsymbol{\theta}^* = \left[\boldsymbol{c}^*; \mathrm{vec}(\boldsymbol{A}_1^*); \ldots; \mathrm{vec}(\boldsymbol{A}_{L-1}^*)\right]$ such that $\mathbf{w}(\boldsymbol{\theta}^*)^\top \boldsymbol{x}_i = y_i$ for all $(\boldsymbol{x}_i, y_i) \in \mathcal{D}$, we aim to solve (4) for this choice of R. In this case the mapping f is non-linear with respect to $\boldsymbol{\theta}$. 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 $\tilde{\boldsymbol{\Delta}}$ that satisfies the relaxed linearized constraints and, when added to $\boldsymbol{\theta}^*$, yields an optimal solution to (4).

Lemma 1. *Denote the retain set input subspace by $\boldsymbol{S}_r = \mathrm{span}\{\boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}_r\}$ and partition the perturbation as $\tilde{\boldsymbol{\Delta}} = \left[\tilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}; \mathrm{vec}(\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}); \ldots; \mathrm{vec}(\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_{L-1}})\right]$ in the same manner as $\boldsymbol{\theta}$. Set*

$$\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1} = -\left\|\boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^*\right\|_2^{-2} \boldsymbol{A}_2^{*\top} \cdots \boldsymbol{A}_{L-1}^{*\top} \boldsymbol{c}^* \mathcal{P}_{\boldsymbol{S}_r^\perp}\left(\mathbf{w}(\boldsymbol{\theta}^*)\right)^\top \tag{8}$$

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

The above result shows that the perturbation direction defined in (8) leads to an optimal solution for (4) once added to $\boldsymbol{\theta}^*$, 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 $\tilde{\boldsymbol{\Delta}}$. In fact, we can show that without adding a proper regularization term $\hat{R}$ to the loss, $\tilde{\boldsymbol{\Delta}}$ would not be a solution of the relaxed problem (see Appendix C.4.1). We next characterize the appropriate regularization $\hat{R}(\boldsymbol{\Delta})$ needed to ensure that $\tilde{\boldsymbol{\Delta}}$ is the optimal solution to the surrogate problem in (7).

Theorem 4.2. *The solution to the relaxed unlearning problem (7) with the following choice of $\tilde{R}$ solves the exact unlearning problem (4) for $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$.*

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

4.2.2 Minimizing Parameter Norm

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

Theorem 4.3. *Let $\hat{\boldsymbol{\theta}}_r^*$ solve (4) for $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$, so $\mathbf{w}(\hat{\boldsymbol{\theta}}_r^*)$ is the min ℓ_2-norm linear predictor over $\mathcal{D}_r$. Define $\rho = \|\mathbf{w}(\hat{\boldsymbol{\theta}}_r^*)\|_2$ and let $\boldsymbol{v}_\ell \in \mathbb{R}^{h_\ell}$ for $\ell \in [L-1]$ each satisfy $\|\boldsymbol{v}_\ell\|_2 = 1$. Set*

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

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

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 $R(\boldsymbol{\theta}) = \|\mathbf{w}(\boldsymbol{\theta})\|_2$ using the constraint relaxation and then update the parameters as prescribed by Theorem 4.3.

4.3 2-Layer Perceptron

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

$$\boldsymbol{\theta}_r^* \in \operatorname*{argmin}_{\boldsymbol{\theta}} R(\boldsymbol{\theta}) \quad \text{s.t. } \boldsymbol{c}^\top \phi(\boldsymbol{A}\boldsymbol{x}_i) = y_i \quad \forall (\boldsymbol{x}_i, y_i) \in \mathcal{D}_r \tag{10}$$

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 [RSSZ07], while for ReLU activations specifically, some n_r-sample datasets need at least $n_r - 2$ neurons [YSJ19]. 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 $\boldsymbol{\theta}^*$, 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 $\boldsymbol{\Delta} = \left[\boldsymbol{\Delta}_c\,;\,\mathrm{vec}(\boldsymbol{\Delta}_A)\right]$, yielding the specific instance of the linearized problem (6) for this model class:

$$\min_{\boldsymbol{\Delta}}\; R(\boldsymbol{\theta}^* + \boldsymbol{\Delta})\;\text{ s.t. }\;\boldsymbol{\Delta}_c^\top \phi(\boldsymbol{A}^*\boldsymbol{x}_i) + \mathrm{tr}\Big\{\boldsymbol{\Delta}_A^\top\left(\phi'(\boldsymbol{A}^*\boldsymbol{x}_i)\odot \boldsymbol{c}^*\right)\boldsymbol{x}_i^\top\Big\} = 0\;\;\forall(\boldsymbol{x}_i,\boldsymbol{y}_i)\in\mathcal{D}_r, \tag{11}$$

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 $\tilde{\boldsymbol{\Delta}}$, modifying only the last layer $\boldsymbol{c}^*$, exists and yields a network satisfying (10) with at most n_r active neurons.

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

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 $\boldsymbol{A}^*$ ensures that solving (7) yields a network feasible for (10) with at most n_r active neurons.

Theorem 4.4. *For $R(\boldsymbol{\theta})$ which measures the number of active neurons of the network $f(\boldsymbol{\theta}, \cdot)$, define*

$$\tilde{R}(\boldsymbol{\theta}\,;\boldsymbol{\theta}^*) = R(\boldsymbol{\theta}) + \delta_{\{\boldsymbol{A}\neq\boldsymbol{A}^*\}} \tag{12}$$

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 $s = \dim\left(\mathrm{span}\{\phi(\boldsymbol{A}^*\boldsymbol{x})\}_{(\boldsymbol{x},y)\in\mathcal{D}_r}\right) \le n_r$ active neurons, where $n_r = |\mathcal{D}_r|$.

Theorem 4.4 shows that for general activation functions, linearizing the constraints to (10) and minimizing the sum of the complexity measure R along the appropriator 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 $\{\phi(\boldsymbol{A}^*\boldsymbol{x})\}_{(\boldsymbol{x},y)\in\mathcal{D}_r}$. 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 [RSSZ07] for minimum width interpolation.

The drift regularizer $\hat{R}$ only allows perturbations to $\boldsymbol{c}^*$, so the solution to (7) reduces width via sparsity in the updated last layer $\boldsymbol{c}^* + \tilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}$, while leaving the first layer $\boldsymbol{A}^*$ unchanged. Although $\boldsymbol{c}^* + \tilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}$ relies on a small set of features, the feature map $\phi(\boldsymbol{A}^*\boldsymbol{x})$ still reflects representations learned from all of $\mathcal{D}$. We show, however, that the sparsity of $\boldsymbol{c}^* + \tilde{\boldsymbol{\Delta}}_{\boldsymbol{c}}$ can be propagated into $\boldsymbol{A}^*$, 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 4.4.

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

Algorithm 1 MinNorm-OG

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

Thus, for any parameters $\boldsymbol{\theta}$, we can apply a simple update to recover new parameters $\hat{\boldsymbol{\theta}}$ which behave like an $R(\boldsymbol{\theta})$-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 $\tilde{\boldsymbol{\theta}} = [\tilde{\boldsymbol{c}}; \operatorname{vec}(\boldsymbol{A}^*)]$ solve (7) for $\tilde{R}$ defined in (12), and define the updated first layer as $\hat{\boldsymbol{A}} = (\mathbf{1}_{\tilde{\boldsymbol{c}} \neq 0} \mathbf{1}^\top) \odot \boldsymbol{A}^*$. Then network parameterized by $\hat{\boldsymbol{\theta}} = [\tilde{\boldsymbol{c}}; \operatorname{vec}(\hat{\boldsymbol{A}})]$ similarly interpolates $\mathcal{D}_r$, has the same number of active neurons $R(\tilde{\boldsymbol{\theta}}) = R(\hat{\boldsymbol{\theta}})$, and $\hat{\boldsymbol{A}}$ has at most $R(\hat{\boldsymbol{\theta}})$ non-zero rows.*

Thus, solving the relaxed problem (7) and updating $\boldsymbol{A}^*$ 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 unlearning algorithm MinNorm-OG (Algorithm 1). At epoch t, we alternate between solving a version of the relaxed unlearning problem (7) and descending the loss 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 $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$ and $\hat{R}(\boldsymbol{\Delta}) = \lambda_t \|\boldsymbol{\Delta}\|_2^2$ 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} of each batch. For this R and $\hat{R}$, the solution to (7) perturbs $\boldsymbol{\theta}$ toward its projection onto the span of model gradients over this subsample (see Appendix D), which can be interpreted as a proximal update under the orthogonal gradient constraint. The main overhead relative to gradient descent comes from solving for $\tilde{\boldsymbol{\Delta}}$ via a QR decomposition with complexity $O(dn_{\text{pert}}^2)$, which is negligible compared to the $O(dn_B)$ cost of gradient descent when $n_{\text{pert}} < \sqrt{n_B}$, where $n_B = |\mathcal{B}|$ is the batch size. Moreover, we often set $\lambda_t = \infty$ for many epochs in practice, skipping this cost entirely.

5.1 Experiments

We test our algorithm against the following existing methods. GD [NRS21] runs gradient descent on $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}_r)$, while Noisy GD (NGD) [CS23] adds gradient noise to the GD steps. GA [GNG21] runs gradient ascent on $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}_f)$. NegGrad+ (NGP) [KTHT23] minimizes a weighted combination of the GD and GA objectives. SCRUB [KTHT23] optimizes three objectives: minimizing $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}_r)$, minimizing KL divergence of model outputs on $\mathcal{D}_r$ relative to the original model, and maximizing

Table 1: Data Poisoning experiment results, measured as the sup-norm distance between the retain set trend $y = \sin(x)$ and the outputs of the unlearning algorithms (smaller is better). We report medians over 20 trials, along with the range of the central 10 values

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

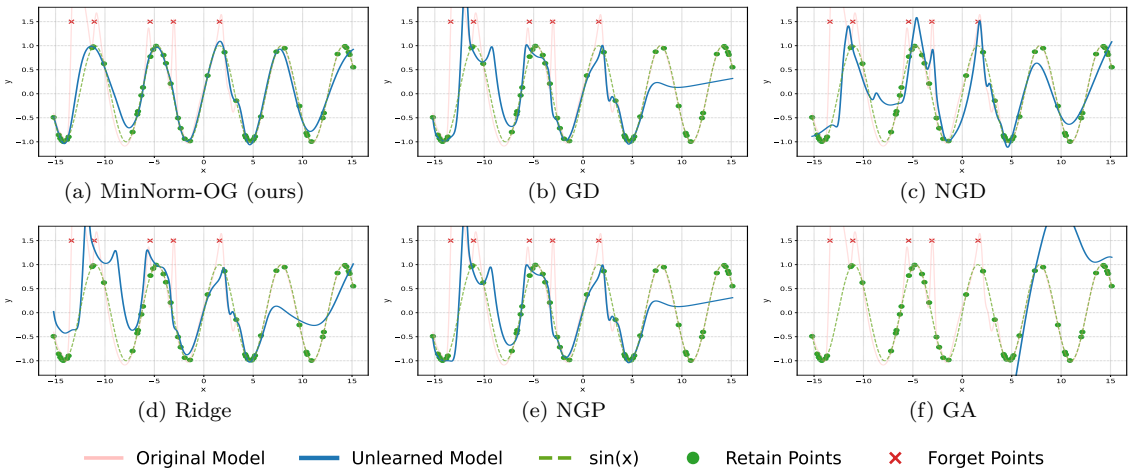

Figure 1: Example unlearned model fits when given 100 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

KL divergence on $\mathcal{D}_f$. Negative Preference Optimization (NPO) [ZLBM24] runs a form of gradient ascent over $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}_f)$ inspired by preference optimization. NPO and SCRUB only apply to models which output a class distribution. To highlight the performance of our algorithm, we also compare to ridge regression, which approximates our unlearning objective (4) for $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2$ by minimizing $\mathcal{J}(\boldsymbol{\theta}; \mathcal{D}_r) + \lambda_t \|\boldsymbol{\theta}\|_2^2$. The minimizer of this regularized objective converges to the minimum-ℓ_2-norm loss minimizer as $\lambda_t \to 0$ [HMRT22].

While recent work proposed various unlearning benchmarks, especially for LLMs [CS23; SLH-MZHLZSZ25; RWJCBVCHG25b; RWJCBVCHG25a], they often rely on opaque metrics that emphasize suppressing forget-set generation. In contrast, we present the following experiments with interpretable quantitative metrics. See Appendix E for full details.

Data Poisoning. We train a shallow 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 $\mathcal{X} = [-15, 15] \subseteq \mathbb{R}$. We evaluate the output $\boldsymbol{\theta}$ of each unlearning method by measuring the deviation from the retain set trend, given by $\sup_{\boldsymbol{x} \in \mathcal{X}} |f(\boldsymbol{\theta}, \boldsymbol{x}) - \sin(\boldsymbol{x})|$. Results are reported in Table 1 as the median over 20 trials along with the range of the central 10 trials, with visualizations in Figure 1. With just 10 epochs, the results vary widely, but they become more consistent as the number of unlearning epochs increases. We observe in general that the methods which mainly descend the loss on $\mathcal{D}_r$ (GD, NGD, Ridge) struggle to escape from the initial solution which fits the poisoned samples, while the methods which include ascent (NGP, GA) diverge from the sine curve in regions unrelated to the forget points.

Multi-Class Label Erasure. We use MNIST and CIFAR-10 [LCB10; Kri09], creating red, green,

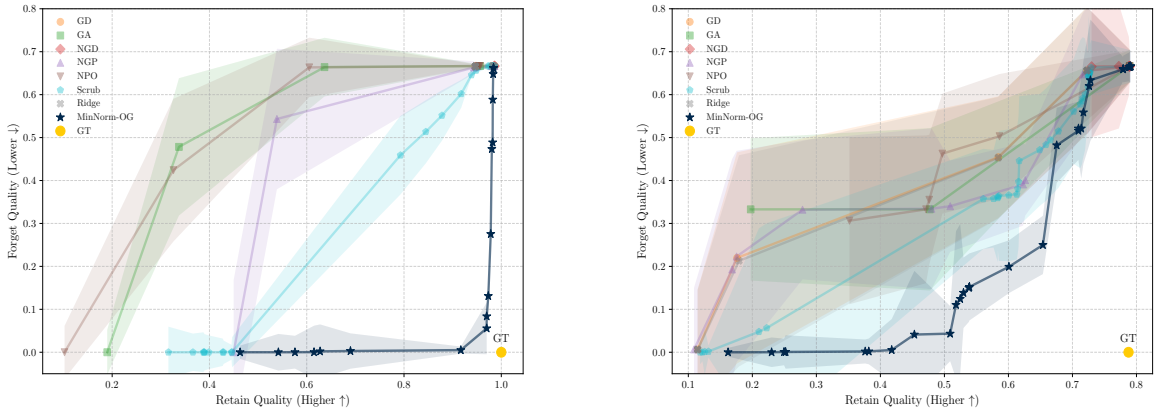

Figure 2: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST (left) and CIFAR-10 (right). Models predict both color and content, but the retain set contains only gray images. The x-axis shows accuracy on gray test images (higher is better), and the y-axis shows mean squared error between predicted probability of gray on all inputs and the target of 1 (lower is better). The ground truth unlearned model (GT) performs well on gray inputs but always predicts gray with probability 1. MinNorm-OG (ours) strictly dominates the other methods.

Table 2: Unlearning performance across constraints on the number of epochs and percentage of accessible retain set samples for the Representation Collapse experiment. 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 accuracy on duplicate training images labeled by color only (higher is better). We report medians over 5 trials, along with the range of the central 3 values.

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

and gray copies of each image. The model is trained to predict both content (digit or object) and color. The retain set $\mathcal{D}_r$ contains all 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 forget quality by the mean squared error between the predicted gray probability and the ideal value of 1 across all colored inputs. Figure 2 shows the Pareto frontier for each method. Each point is a median over 5 trials for one hyperparameter setting, with shaded uncertainty as half the interquartile range. The optimal point $(1, 0)$ indicates perfect retain accuracy and zero gray prediction error. The ground truth unlearned model is labeled GT. Our method MinNorm-OG performs best in both tasks, though all methods struggle to preserve accuracy on CIFAR-10, a harder task than to MNIST. We observe that descent-based methods (GD, NGD, Ridge) often remain near the initial model (upper right region), which is already near-optimal on $\mathcal{D}_r$ and provides weak gradients for unlearning.

Representation Collapse. We use a subset of MNIST where the digits 0 and 1 are assigned a unique color. The retain set $\mathcal{D}_r$ contains the digits colored uniquely, while the forget set $\mathcal{D}_f$ contains

digits with mismatched colors. The ground truth unlearned model predicts from color alone, as it perfectly determines the label in $\mathcal{D}_r$ and is easier to learn than digit shape. In contrast, models trained on the full dataset $\mathcal{D} = \mathcal{D}_r \sqcup \mathcal{D}_f$ must rely on shape, since color is no longer predictive. For evaluation, we relabel training images by color and assess unlearning via color-label accuracy, testing if the unlearning methods can collapse the original model into just a color classifier. Results exhibit a bimodal distribution across trials, as each method must transition from an initial model that classifies digits perfectly to one that achieves the same retain accuracy using only color. When this transition fails, the model often reverts to digit-based predictions, leading to high variance. To reflect this behavior robustly, Table 2 reports median color accuracy over 5 trials, along with the range of the central 3 values. We note that MinNorm-OG consistently performs best.

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

[BNLGG22] J. Bae, N. Ng, A. Lo, M. Ghassemi, and R. B. Grosse. "If influence functions are the answer, then what is the question?" *Advances in Neural Information Processing Systems* 35 (2022), pp. 17953–17967 (pages 2, 3, 5, 6).

[BCCJTZLP21] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot. "Machine Unlearning". In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 141–159. DOI: 10.1109/SP40001.2021.00019 (page 3).

[CY15] Y. Cao and J. Yang. "Towards making systems forget with machine unlearning". In: *2015 IEEE symposium on security and privacy*. IEEE. 2015, pp. 463–480 (page 2).

[CCP18] CCPA. *California Consumer Privacy Act of 2018 (CCPA)*. 2018 (page 2).

[CZYZ24] H. Chen, T. Zhu, X. Yu, and W. Zhou. "Machine unlearning via null space calibration". In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*. 2024, pp. 358–366 (page 3).

[CS23] R. Chourasia and N. Shah. "Forget unlearning: Towards true data-deletion in machine learning". In: *International conference on machine learning*. PMLR. 2023, pp. 6028–6073 (pages 3, 5, 10, 11).

[FAML20] M. Farajtabar, N. Azizan, A. Mott, and A. Li. "Orthogonal gradient descent for continual learning". In: *International conference on artificial intelligence and statistics*. PMLR. 2020, pp. 3762–3773 (pages 3, 28).

[GRBTKDYZA24] I. O. Gallegos, R. A. Rossi, J. Barrow, M. M. Tanjim, S. Kim, F. Dernoncourt, T. Yu, R. Zhang, and N. K. Ahmed. "Bias and fairness in large language models: A survey". *Computational Linguistics* 50.3 (2024), pp. 1097–1179 (page 2).

[GDP16] GDPR. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data (General Data Protection Regulation)*. 2016 (page 2).

[GKKMSZ23] B. Ghazi, P. Kamath, R. Kumar, P. Manurangsi, A. Sekhari, and C. Zhang. "Ticketed learning–unlearning schemes". In: *The Thirty Sixth Annual Conference on Learning Theory*. PMLR. 2023, pp. 5110–5139 (page 3).

[GNG21] L. Graves, V. Nagisetty, and V. Ganesh. "Amnesiac machine learning". *Proceedings of the AAAI Conference on Artificial Intelligence* 35.13 (2021), pp. 11516–11524 (pages 3, 5, 10).

[GGHV20] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten. "Certified data removal from machine learning models". In: *Proceedings of the 37th International Conference on Machine Learning*. 2020, pp. 3832–3842 (pages 2, 3, 5, 6).

[Ham74] F. R. Hampel. "The influence curve and its role in robust estimation". *Journal of the american statistical association* 69.346 (1974), pp. 383–393 (page 3).

[HMRT22] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. "Surprises in high-dimensional ridgeless least squares interpolation". *Annals of statistics* 50.2 (2022), p. 949 (pages 5, 11).

[HZRS16] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (pages 32, 41).

[HRGV24] T. Hoang, S. Rana, S. Gupta, and S. 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*. 2024, pp. 4819–4828 (page 3).

[JBVRCCH25] X. Jin, Z. Bu, B. Vinzamuri, A. Ramakrishna, K.-W. Chang, V. Cevher, and M. 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)*. Ed. by L. Chiruzzo, A. Ritter, and L. Wang. Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 11278–11294. ISBN: 979-8-89176-189-6 (page 3).

[Kri09] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. Technical Report. University of Toronto, 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (pages 11, 32, 41).

[KTHT23] M. Kurmanji, P. Triantafillou, J. Hayes, and E. Triantafillou. "Towards unbounded machine unlearning". *Advances in neural information processing systems* 36 (2023), pp. 1957–1987 (pages 3, 5, 10).

[LCB10] Y. LeCun, C. Cortes, and C. J. Burges. *MNIST handwritten digit database*. http://yann.lecun.com/exdb/mnist/. 2010 (pages 11, 32, 41).

[LH19] I. Loshchilov and F. Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2019 (page 26).

[MFSLK24] P. Maini, Z. Feng, A. Schwarzschild, Z. C. Lipton, and J. Z. Kolter. "TOFU: A Task of Fictitious Unlearning for LLMs". In: *First Conference on Language Modeling*. 2024 (page 3).

[NRS21] S. Neel, A. Roth, and S. Sharifi-Malvajerdi. "Descent-to-delete: Gradient-based methods for machine unlearning". In: *Algorithmic Learning Theory*. PMLR. 2021, pp. 931–962 (pages 3, 5, 10).

[PDLKSN25] M. Pawelczyk, J. Z. Di, Y. Lu, G. Kamath, A. Sekhari, and S. Neel. "Machine Unlearning Fails to Remove Data Poisoning Attacks". In: *The Thirteenth International Conference on Learning Representations*. 2025 (page 3).

[RWJCBVCHG25a] A. Ramakrishna, Y. Wan, X. Jin, K.-W. Chang, Z. Bu, B. Vinzamuri, V. Cevher, M. Hong, and R. Gupta. "Lume: Llm unlearning with multitask evaluations". *arXiv preprint arXiv:2502.15097* (2025) (page 11).

[RWJCBVCHG25b] A. Ramakrishna, Y. Wan, X. Jin, K.-W. Chang, Z. Bu, B. Vinzamuri, V. Cevher, M. Hong, and R. Gupta. "Semeval-2025 task 4: Unlearning sensitive content from large language models". *arXiv preprint arXiv:2504.02883* (2025) (page 11).

[RSSZ07] S. Rosset, G. Swirszcz, N. Srebro, and J. Zhu. "l1 regularization in infinite dimensional feature spaces". In: *Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings 20*. Springer. 2007, pp. 544–558 (pages 8, 9).

[SAKS21] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh. "Remember what you want to forget: Algorithms for machine unlearning". *Advances in Neural Information Processing Systems* 34 (2021), pp. 18075–18086 (pages 2, 3, 5, 6).

[SLHMZHLZSZ25] W. Shi, J. Lee, Y. Huang, S. Malladi, J. Zhao, A. Holtzman, D. Liu, L. Zettlemoyer, N. A. Smith, and C. Zhang. "MUSE: Machine Unlearning Six-Way Evaluation for Language Models". In: *The Thirteenth International Conference on Learning Representations*. 2025 (page 11).

[YSJ19] C. Yun, S. Sra, and A. Jadbabaie. "Small relu networks are powerful memorizers: a tight analysis of memorization capacity". *Advances in neural information processing systems* 32 (2019) (page 8).

[ZLBM24] R. Zhang, L. Lin, Y. Bai, and S. Mei. "Negative Preference Optimization: From Catastrophic Collapse to Effective Unlearning". In: *First Conference on Language Modeling*. 2024 (pages 3, 11, 27).

Appendix

A Minimum Norm Solutions to Linear Regression

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

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

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

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \, \frac{1}{2} \|\boldsymbol{\theta}\|_2^2 \;\text{ s.t. }\; \boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} \tag{13}$$

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

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

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

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

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

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

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

□

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

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

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

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

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

Lemma 5. *Consider any subset $\mathcal{D}_r \subseteq \mathcal{D}$, and define $\boldsymbol{\theta}_r^*$ as the linear regression solution over just $\mathcal{D}_r$ with minimum norm:*

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

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

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

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

□

B Loss Minimization Does not Protect Against Data Leakage

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

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

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

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

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

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

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

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

C Proofs

C.1 Proof of Theorem 2.1

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

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

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

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

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

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

C.2 Proof of Theorem 4.1

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

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

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

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

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

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

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

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

Thus the updated unlearned vector is

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

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

C.3 Proof of Lemma 1

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

We first show that $\tilde{\boldsymbol{\Delta}}$ is feasible for the relaxed problem (7). Firstly, $\tilde{\boldsymbol{\Delta}}$ is zero in all entries except those corresponding to the perturbation of $\boldsymbol{A}_1$, so we only need to ensure that $\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}$ is orthogonal to $\nabla_{\boldsymbol{A}_1} f(\boldsymbol{\theta}^*, \boldsymbol{x})$ for each $(\boldsymbol{x}, y) \in \mathcal{D}_r$. Recall we denote the retain set input space as $\boldsymbol{S}_r = \text{span}\{\, \boldsymbol{x} \mid (\boldsymbol{x}, y) \in \mathcal{D}_r \,\}$, and $\tilde{\boldsymbol{\Delta}}_{\boldsymbol{A}_1}$ is defined as

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

Further, the gradients are computed as

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

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

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

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

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

effective linear predictor after the perturbation:

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

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

C.4 Proof of Theorem 4.2

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

Now, let $\boldsymbol{\Delta}^*$ minimize the relaxed problem (7) for this $\tilde{R}$ defined in (9). Then because $\tilde{R}$ ensures that all elements of $\boldsymbol{\Delta}^*$ which do not correspond to $\boldsymbol{A}_1$ are zero, we must have that for any $(\boldsymbol{x}, y) \in \mathcal{D}_r$:

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

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

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

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

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

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

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

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

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

C.4.1 Necessity of Additional Regularizer $\hat{R}$ for Theorem 4.2

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 $\boldsymbol{\Delta}$, we can construct a simple example where $\boldsymbol{\Delta}$ satisfies just the linearization of the data interpolation constraints but the updated network $\boldsymbol{\theta}^* + \boldsymbol{\Delta}$ no longer interpolates $\mathcal{D}_r$.

Consider a dataset of two samples $\mathcal{D} = \{(\boldsymbol{e}_1, 1), (\boldsymbol{e}_2, 1)\}$, where $\boldsymbol{e}_i \in \mathbb{R}^m$ is the ith standard basis vector for any $m \geq 3$. Consider the original 2-layer interpolating network trained on $\mathcal{D}$ defined by parameters $\boldsymbol{\theta}^* = \left[\boldsymbol{c}^* \,; \mathrm{vec}(\boldsymbol{A}^*)\right]$, where $\boldsymbol{c}^* = \boldsymbol{e}_1 + \boldsymbol{e}_2 \in \mathbb{R}^m$ and $\boldsymbol{A}^*$ is the $m \times m$ identity matrix $\boldsymbol{A}^* = \boldsymbol{I}_m$, so $f(\boldsymbol{\theta}^*, \boldsymbol{x}) = \boldsymbol{c}^{*\top} \boldsymbol{A}^* \boldsymbol{x} = (\boldsymbol{e}_1 + \boldsymbol{e}_2)^\top \boldsymbol{x}$.

We set $\mathcal{D}_r = \{(\boldsymbol{e}_1, 1)\}$ and $\mathcal{D}_f = \{(\boldsymbol{e}_2, 1)\}$, and define the perturbation variable $\boldsymbol{\Delta} = \left[\boldsymbol{\Delta}_{\boldsymbol{c}} \,; \mathrm{vec}(\boldsymbol{\Delta}_{\boldsymbol{A}})\right]$. Translating the constraints of (6) to this specific problem instance, we have that

$$\boldsymbol{\Delta}_c^\top \boldsymbol{e}_1 + \mathrm{tr}\{\boldsymbol{\Delta}_{\boldsymbol{A}}^\top (\boldsymbol{e}_1 + \boldsymbol{e}_2)\boldsymbol{e}_1^\top\} = 0$$

We then select the values $\boldsymbol{\Delta}_{\boldsymbol{c}} = -\boldsymbol{e}_3$ and $\boldsymbol{\Delta}_{\boldsymbol{A}} = \boldsymbol{e}_3\boldsymbol{e}_1^\top - \boldsymbol{e}_2\boldsymbol{e}_2^\top - \boldsymbol{e}_3\boldsymbol{e}_3^\top$. 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 $(\boldsymbol{A}^* + \boldsymbol{\Delta}_{\boldsymbol{A}})^\top (\boldsymbol{c}^* + \boldsymbol{\Delta}_{\boldsymbol{c}}) = \boldsymbol{0}$.

$$\begin{aligned}
R(\boldsymbol{\theta}^* + \boldsymbol{\Delta}) &= \left\| (\boldsymbol{A}^* + \boldsymbol{\Delta}_{\boldsymbol{A}})^\top (\boldsymbol{c}^* + \boldsymbol{\Delta}_{\boldsymbol{c}}) \right\|_2 \\
&= \left\| (\boldsymbol{I} + \boldsymbol{e}_3\boldsymbol{e}_1^\top - \boldsymbol{e}_2\boldsymbol{e}_2^\top - \boldsymbol{e}_3\boldsymbol{e}_3^\top)^\top (\boldsymbol{e}_1 + \boldsymbol{e}_2 - \boldsymbol{e}_3) \right\|_2 \\
&= \left\| (\boldsymbol{I} + \boldsymbol{e}_1\boldsymbol{e}_3^\top - \boldsymbol{e}_2\boldsymbol{e}_2^\top - \boldsymbol{e}_3\boldsymbol{e}_3^\top)(\boldsymbol{e}_1 + \boldsymbol{e}_2 - \boldsymbol{e}_3) \right\|_2 \\
&= \left\| \boldsymbol{e}_1 + \boldsymbol{e}_2 - \boldsymbol{e}_3 - \boldsymbol{e}_2 - \boldsymbol{e}_1 + \boldsymbol{e}_3 \right\|_2 \\
&= \left\| \boldsymbol{0} \right\|_2 = 0
\end{aligned}$$

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

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 $\boldsymbol{\Delta}$ beyond just the linearized constraints does not lead to the exact unlearning solution.

C.5 Proof of Theorem 4.3

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

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

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

$$\underset{\boldsymbol{c}, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_{L-1}}{\operatorname{argmin}} \; \|\boldsymbol{c}\|_2^2 + \sum_{\ell=1}^{L-1} \|\boldsymbol{A}_\ell\|_F^2 \;\text{ s.t. }\; \boldsymbol{y}_r = \boldsymbol{X}_r \boldsymbol{A}_1^\top \ldots \boldsymbol{A}_{L-1}^\top \boldsymbol{c} \tag{17}$$

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

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

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

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

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

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

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

Then, (17) reduces to

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

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

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

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

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

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

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

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

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

C.6 Proof of Theorem 4.4

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

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

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

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

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

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

Proof of Lemma 6:

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

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

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

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

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

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

C.7 Proof of Proposition 1

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

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

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

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

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

D 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 $\mathcal{G}_r = \mathrm{span}\{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{x})\}_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}_r}$ and consider any $\lambda \geq 0$. We then solve the following problem:

$$\tilde{\boldsymbol{\Delta}} = \operatorname*{argmin}_{\boldsymbol{\Delta}} \|\boldsymbol{\theta} + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2 \quad \text{s.t. } \boldsymbol{\Delta} \in \mathcal{G}_r^\perp. \tag{20}$$

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).

E 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 3: Hyperparameter definitions and their associated methods.

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

E.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 [LH19] optimizer with learning rate η on different effective loss functions. We express the loss functions below.

E.1.1 GD

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

E.1.2 GA

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

E.1.3 NGD

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

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

E.1.4 NGP

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

E.1.5 NPO

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

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

where $\pi_{\boldsymbol{\theta}}(y_f\mid\boldsymbol{x}_f)$ denotes the model's predicted probability of class y_f for input $\boldsymbol{x}_f$ for parameter vector $\boldsymbol{\theta}$. Note that this is equivalent to setting the parameter β in the original NPO paper [ZLBM24] to λ_{reg}.

E.1.6 Scrub

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

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

E.1.7 MinNorm-OG

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

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

Following the AdamW update for this loss, we then (depending on the epoch) perform the model update corresponding to solving the relaxed unlearning problem (7) for $\tilde{R}(\boldsymbol{\theta}+\boldsymbol{\Delta})=\|\boldsymbol{\theta}+\boldsymbol{\Delta}\|_2^2+\lambda\|\boldsymbol{\Delta}\|_2^2$, where λ is a saved parameter of the algorithm. We use the parameters T_{Proj} and T_{GD} to determine which epochs to perform the unlearning update. For the T_{GD} last epochs, we only perform the descent step and skip the unlearning update, similar to Scrub. In the first $T-T_{\text{GD}}$ epochs, we perform the unlearning update every T_{Proj} epochs.

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

To interpret these values, first recall that we solve the relaxed unlearning problem over a subsample of each batch $\mathcal{B}_r'\subseteq\mathcal{B}_r$ where $|\mathcal{B}_r'|=n_{\text{pert}}$. For convenience, define the gradient subspace $\mathcal{G}_r'=\text{span}\{\nabla_{\boldsymbol{\theta}}f(\boldsymbol{\theta},\boldsymbol{x})\}_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{B}_r'}$. As we showed in Appendix D, for any value of λ, the optimal perturbation is then $\tilde{\boldsymbol{\Delta}}=-\frac{1}{1+\lambda}\mathcal{P}_{\mathcal{G}_r'^{\perp}}(\boldsymbol{\theta})$. Thus, the initial value $\lambda=\frac{1}{\lambda_{\text{reg}}}-1$ leads to the perturbation $\tilde{\boldsymbol{\Delta}}=$

$-\lambda_{\text{reg}}\mathcal{P}_{\mathcal{G}_r'^{\perp}}(\boldsymbol{\theta})$. Further, the coefficient update $\lambda = \frac{\lambda'+1}{\gamma_{\text{reg}}} - 1$ leads to a more conservative unlearning perturbation $\tilde{\boldsymbol{\Delta}} = -\gamma_{\text{reg}}\frac{1}{1+\lambda'}\mathcal{P}_{\mathcal{G}_r'^{\perp}}(\boldsymbol{\theta})$, as it is down-weighted by γ_{reg}. Thus, λ_{reg} is the initial strength of the perturbation, normalized to the range $[0, 1]$, and γ_{reg} represents a multiplicative decay of this strength through each update to λ.

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

$$
\begin{aligned}
&\textbf{if } t \bmod T_{\text{Proj}} \neq 0 \text{ or } t \geq T - T_{\text{GD}} \\
&\quad \boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_0 \\
&\textbf{else} \\
&\quad \tilde{\boldsymbol{\Delta}} = \operatorname*{argmin}_{\boldsymbol{\Delta} \in \mathcal{G}_r'^{\perp}} \|\boldsymbol{\theta}_0 + \boldsymbol{\Delta}\|_2^2 + \lambda \|\boldsymbol{\Delta}\|_2^2 \\
&\quad \boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_0 + \tilde{\boldsymbol{\Delta}} \\
&\quad \lambda \leftarrow \frac{\lambda + 1}{\gamma_{\text{reg}}} - 1
\end{aligned}
$$

Gradients for Classification. We make a special note of how we compute the gradient subspace $\mathcal{G}_r'$ for classification tasks. At the parameter value $\boldsymbol{\theta}_0$, the model prediction is $f(\boldsymbol{\theta}_0, \boldsymbol{x}) = \operatorname{argmax} \boldsymbol{z}_{\boldsymbol{\theta}_0}(\boldsymbol{y} \mid \boldsymbol{x})$ where $\boldsymbol{z}_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$ denotes the model's unnormalized logits over the classes $\boldsymbol{y}$ for input $\boldsymbol{x}$ for parameter vector $\boldsymbol{\theta}$. This is not a continuous function of $\boldsymbol{\theta}$, so we cannot compute its gradient directly. However, following prior works [FAML20], we use the gradient $\nabla_{\boldsymbol{\theta}} \left(\boldsymbol{z}_{\boldsymbol{\theta}_0}(\boldsymbol{y} \mid \boldsymbol{x})\right)_j$, where $j = f(\boldsymbol{\theta}_0, \boldsymbol{x})$ is the model's predicted class for input $\boldsymbol{x}$. In other words, we take the gradient of the the unnormalized logits at the index of the maximum value, where we do not treat the index as a function of $\boldsymbol{\theta}$.

E.1.8 Ridge

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

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

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

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

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

E.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 $\mathcal{X} = [-15, 15] \subseteq \mathbb{R}$. 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.

Table 4: Data Poisoning experiment results, measured as the sup-norm distance between the retain set trend $y = \sin(x)$ and the outputs of the unlearning algorithms (smaller is better). We report medians over 20 trials, along with the range of the central 10 values

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

Table 5: Hyperparameter settings for each entry in Table 4. Blank entries indicate that the hyperparameter is not applicable to the corresponding method.

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

Given these poisoned models, we apply each of the unlearning algorithms over a sweep of hyperparameters and evaluate the output $\boldsymbol{\theta}$ of each unlearning method by measuring the deviation from the retain set trend, given by $\sup_{\boldsymbol{x}\in\mathcal{X}} |f(\boldsymbol{\theta}, \boldsymbol{x}) - \sin(\boldsymbol{x})|$. We fix the number of epochs for each algorithm and allow full data access, so each method has access to all of $\mathcal{D}_r$ during unlearning. We repeat the entire process over 20 trials. For the number of unlearning epochs $T \in \{10, 100, 1000\}$, we report the best performance of each algorithm in Table 4 along with the associated hyperparameters in Table 5. We select the parameters for each method by finding the best performing parameters from the possible values in Table 6 using the first 5 trials. We then evaluate over the full 20 trials to obtain our results. We also include visualizations of the recovered models from each unlearning method in Figures 3, 4, and 5. All experiments were run on either a single NVIDIA A40 GPU or a single NVIDIA GH200 GPU.

Table 6: Hyperparameter values tested in the experiments corresponding to Table 4. We denote the total number of epochs T.

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

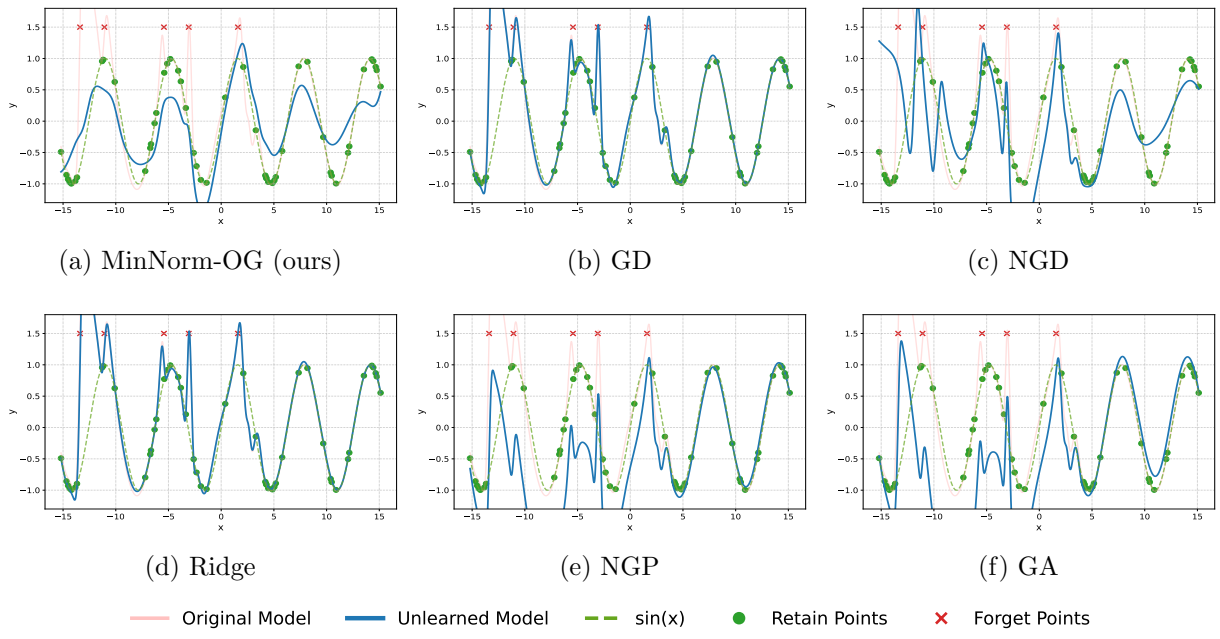

Figure 3: Example unlearned model fits when given 10 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

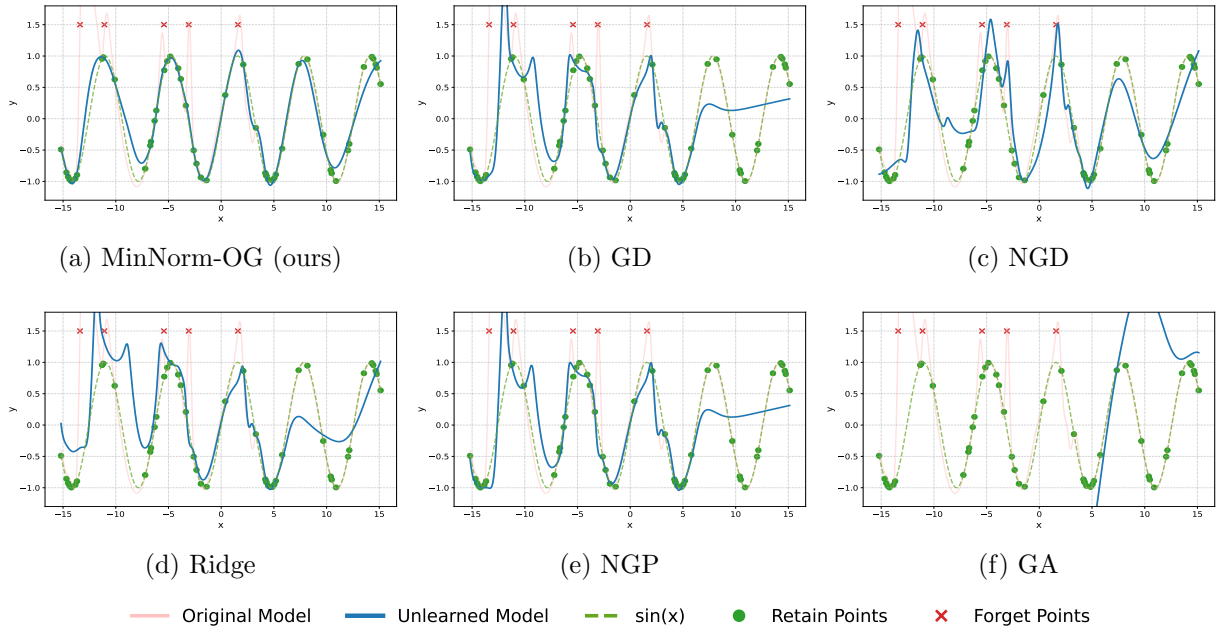

Figure 4: Example unlearned model fits when given 100 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

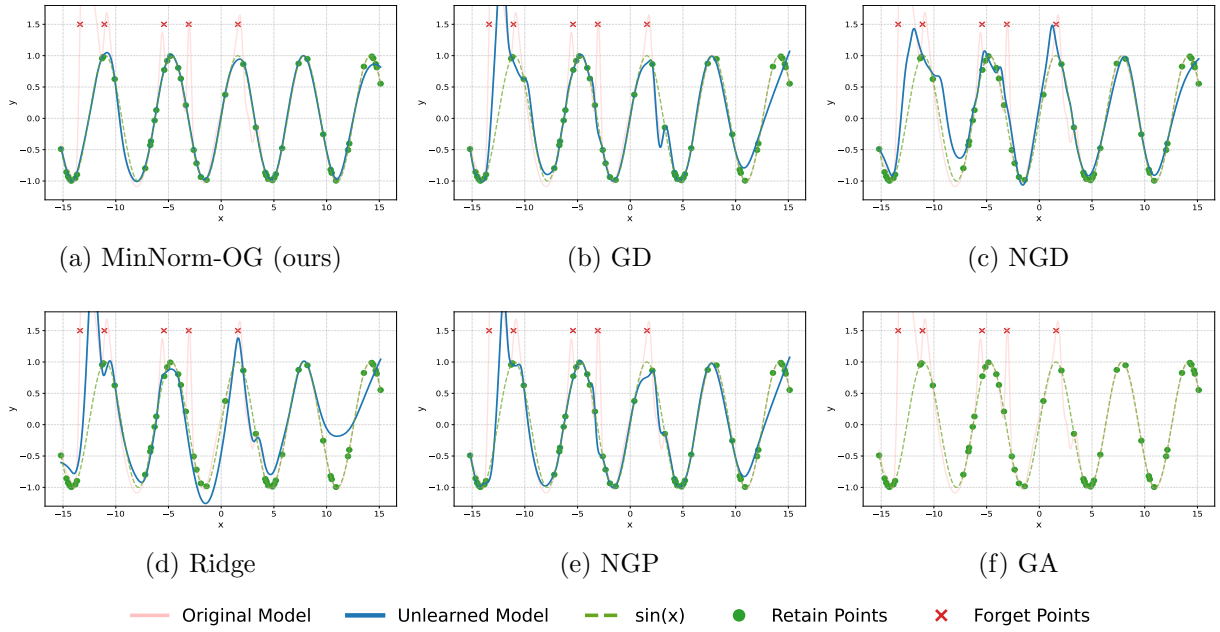

Figure 5: Example unlearned model fits when given 1000 unlearning epochs for the Data Poisoning experiment, where the forget points distort the retain set trend $y = \sin(x)$.

E.3 Multi-Class Label Erasure

We use the MNIST and CIFAR-10 [LCB10; Kri09] datasets, creating red, green, and gray copies of each image in the training sets. We construct the retain set as the entire gray copy, and the forget set as a random subset of the red and green copies. Specifically, we construct the forget set as a random sample of 5 percent of the red samples combined with a random sample of the same size of the green samples. We then train a model to predict both the image content class (digit for MNIST, object for CIFAR) as well as the color on the combined data to serve as the initial model for unlearning. For MNIST, we use a CNN with two convolutional layers, one fully connected layer, and then a separate fully connected prediction head for the color and content class. We train for 100 epochs using an initial learning rate of 10^{-3} and a batch size of 3000 along with the AdamW optimizer. For CIFAR-10, we use a modified ResNet-18 [HZRS16] architecture also with separate prediction heads for the two class types. In this case, we train for 120 epochs using stochastic gradient descent (SGD) with momentum and weight decay. We set the learning rate to 0.02, momentum to 0.9, and weight decay to 5×10^{-4}, and we use a batch size of 256. We also apply a learning rate scheduler which applies a multiplicative decay of 0.1 every 50 epochs. For each dataset, the ground truth models are trained on the gray images alone using the same training parameters.

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

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

We plot the Pareto frontiers and report the hyperparameters used to obtain the optimal curves in the figures and tables below. We do not necessarily sweep over every combination of the reported settings for every algorithm, as we selected some hyperparameter choices to fill out different areas of the frontier when needed. For example, we often had to set larger learning rates for Scrub to trace a full curve from the upper right to the bottom left. Without doing so, the Scrub results did not reach the bottom half of the plot as the unlearned models remained too close to the initial trained model. Similarly, for some of the CIFAR-10 experiments our algorithm MinNorm-OG needed smaller learning rates and small values of λ_{reg} than usual to sweep through full range through the top right

corner, as this area represents models which remain close to the original trained model.

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

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

Table 7: Hyperparameter values tested for the results in Figure 6 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .05$ and $T = 5$.

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

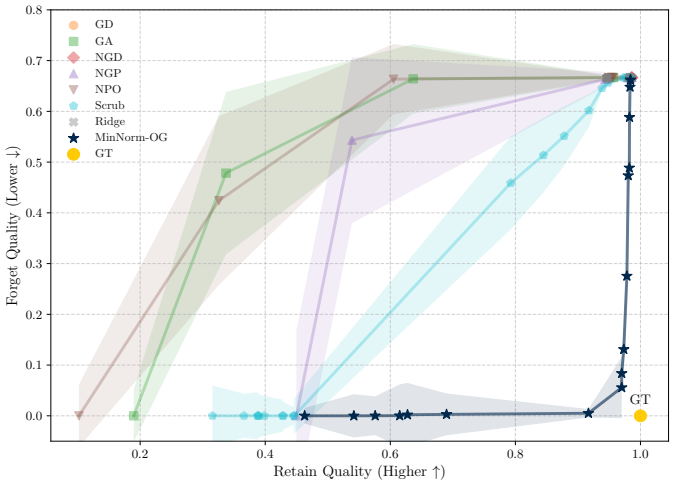

Figure 6: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .05$ and $T = 5$. This is an enlarged version of the left subfigure in Figure 2.

Table 8: Hyperparameter values tested for the results in Figure 7 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .01$ and $T = 5$.

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

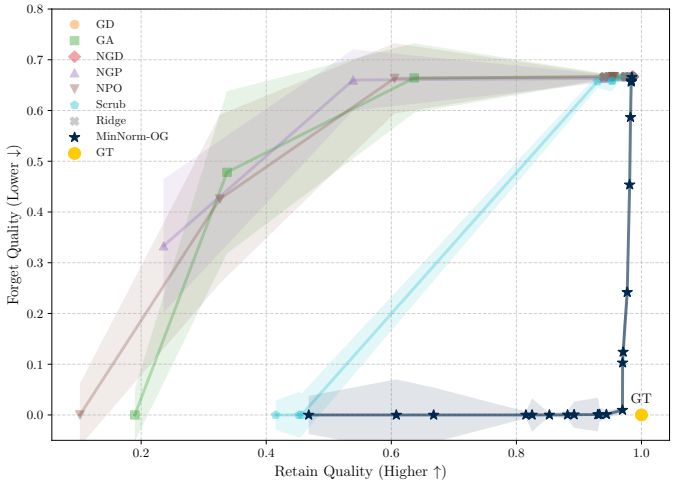

Figure 7: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .01$ and $T = 5$.

Table 9: Hyperparameter values tested for the results in Figure 8 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .05$ and $T = 2$.

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

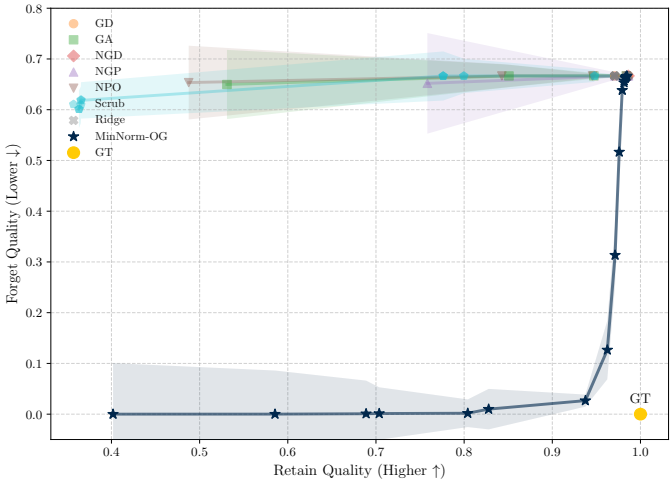

Figure 8: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .05$ and $T = 2$.

Table 10: Hyperparameter values tested for the results in Figure 9 running the Multi-Label Class Erasure experiment on MNIST with $p_{\text{retain}} = .01$ and $T = 8$.

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

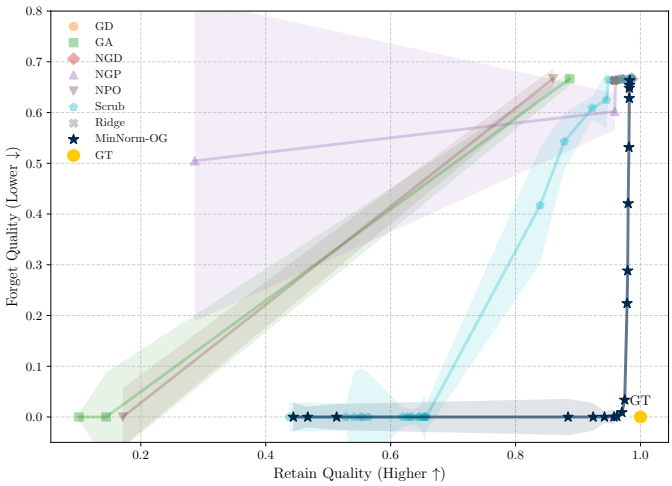

Figure 9: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on MNIST with $p_{\text{retain}} = .01$ and $T = 8$.

Table 11: Hyperparameter values tested for the results in Figure 10 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 5$.

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

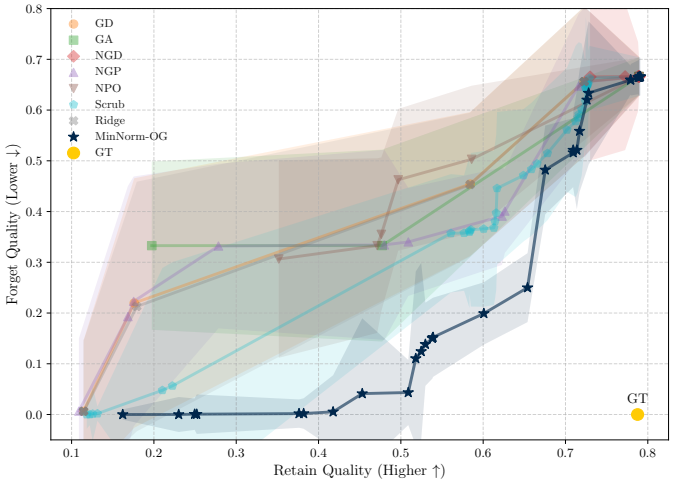

Figure 10: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 5$. This is an enlarged version of the right subfigure in Figure 2.

Table 12: Hyperparameter values tested for the results in Figure 11 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 10$.

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

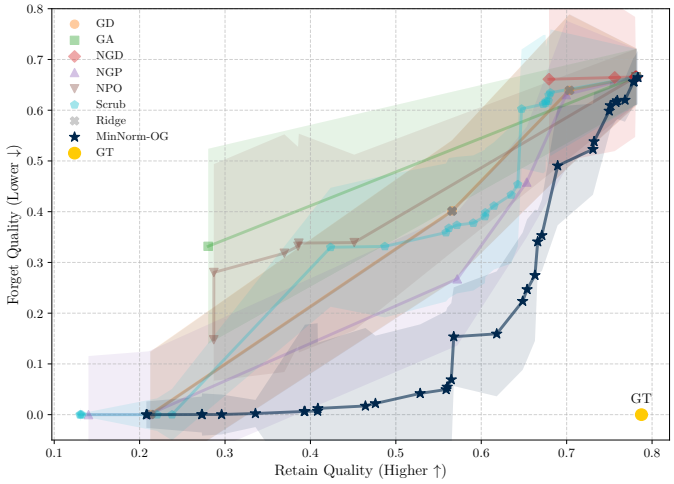

Figure 11: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .001$ and $T = 10$.

Table 13: Hyperparameter values tested for the results in Figure 12 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 5$.

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

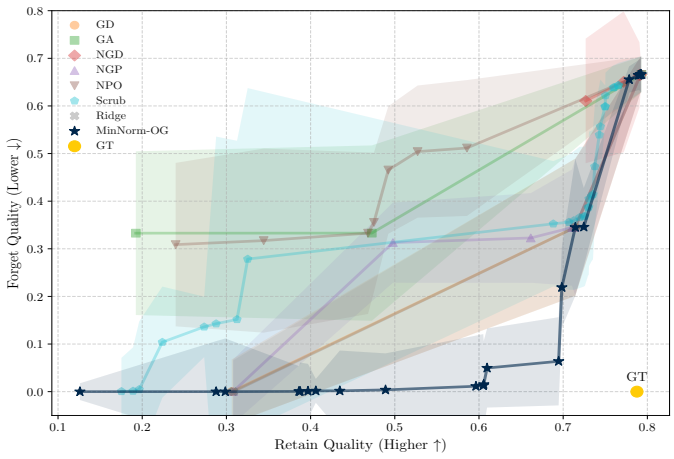

Figure 12: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 5$.

Table 14: Hyperparameter values tested for the results in Figure 13 running the Multi-Label Class Erasure experiment on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 10$.

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

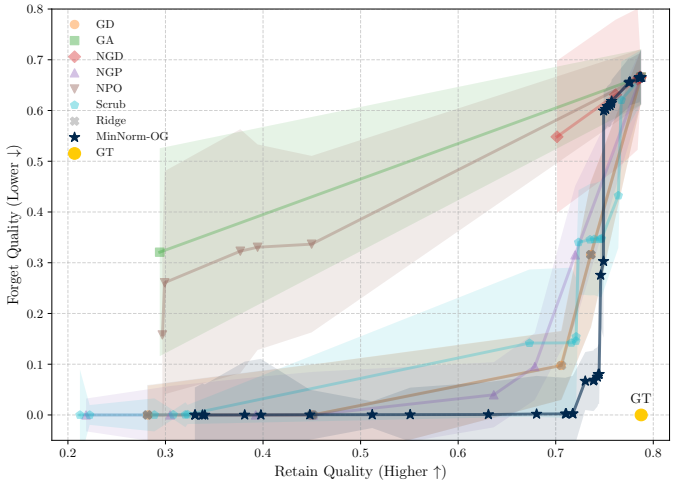

Figure 13: Pareto frontiers for each method across hyperparameter settings in the Multi-Class Label Erasure task on CIFAR-10 with $p_{\text{retain}} = .01$ and $T = 10$.

E.4 Representation Collapse

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

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

Table 15: Unlearning performance across constraints on the number of epochs and percentage of accessible retain set samples for the Representation Collapse experiment. Evaluation is measured as accuracy on duplicate training images labeled by color only (higher is better). We report medians over 5 trials, along with the range of the central 3 values.

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

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

We search over hyperparameters and report the best results for each algorithm in each setting in Table 15. We write Retain % to denote $100 \times p_{\text{retain}}$. We observed that the results can exhibit a bimodal distribution across trials, as each method must transition from an initial model that classifies digits perfectly to one that achieves the same retain accuracy using only color. When this transition fails, the model often reverts to digit-based predictions, leading to high variance in the results. To reflect this behavior robustly, Table 15 reports median color accuracy over 5 trials, along with the range of the central 3 values. We note that MinNorm-OG consistently performs best. For each setting of the number of epochs and the Retain %, we show the hyperparameters we considered in Tables 16,17,18,19,20,and 21 before reporting the best performance out of each combination for each algorithm.

All training was performed on a cluster of NVIDIA GH200 GPUs. For example, sweeping through all hyperparameter combinations listed in Table 16 for each algorithm completed in about 10 minutes using 8 nodes.

Table 16: Hyperparameter values considered for the Representation Collapse Experiment with $T = 5$ and $p_{\text{retain}} = 0.01$.

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

Table 17: Hyperparameter values considered for the Representation Collapse Experiment with $T = 5$ and $p_{\text{retain}} = 0.1$.

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

Table 18: Hyperparameter values considered for the Representation Collapse Experiment with $T = 8$ and $p_{\text{retain}} = 0.01$.

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

Table 19: Hyperparameter values considered for the Representation Collapse Experiment with $T = 8$ and $p_{\text{retain}} = 0.1$.

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

Table 20: Hyperparameter values considered for the Representation Collapse Experiment with $T = 10$ and $p_{\text{retain}} = 0.01$.

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

Table 21: Hyperparameter values considered for the Representation Collapse Experiment with $T = 10$ and $p_{\text{retain}} = .1$.

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

E.5 Asset Information

We use the MNIST [LCB10] and CIFAR-10 [Kri09] datasets in our experiments. CIFAR-10 is publicly available but does not specify an explicit license. MNIST is also publicly available and is typically distributed under the Creative Commons Attribution-ShareAlike 3.0 License. Additionally, we use the ResNet-18 [HZRS16] architecture and pretrained weights from PyTorch's `torchvision` library, which are licensed under the BSD 3-Clause License.