Machine Unlearning Fails to Remove Data Poisoning Attacks

Martin Pawelczyk [*1] **Jimmy Z. Di** [*2] **Yiwei Lu** [2 3] **Gautam Kamath** [+2 3] **Ayush Sekhari** [+4] **Seth Neel** [+1]

Abstract

We revisit the efficacy of several practical methods for approximate machine unlearning developed for large-scale deep learning. In addition to complying with data deletion requests, one often-cited potential application for unlearning methods is to remove the effects of training on poisoned data. We experimentally demonstrate that, while existing unlearning methods have been demonstrated to be effective in a number of evaluation settings (e.g., alleviating membership inference attacks), they fail to remove the effects of data poisoning, across a variety of types of poisoning attacks (indiscriminate, targeted, and a newly-introduced Gaussian poisoning attack) and models (image classifiers and LLMs); even when granted a relatively large compute budget. In order to precisely characterize unlearning efficacy, we introduce new evaluation metrics for unlearning based on data poisoning. Our results suggest that a broader perspective, including a wider variety of evaluations, are required to avoid a false sense of confidence in machine unlearning procedures for deep learning without provable guarantees. Moreover, while unlearning methods show some signs of being useful to efficiently remove poisoned datapoints without having to retrain, our work suggests that these methods are not yet "ready for prime time," and currently provide limited benefit over retraining.

1. Introduction

Machine Learning (ML) models are often trained on large-scale datasets, which can include significant amounts of sensitive or personal data. This practice raises privacy concerns as the models can memorize and inadvertently reveal information about individual points in the training set. Consequently, there is an increasing demand for the capability to selectively remove training data from models which have already been trained, a functionality which helps comply with various privacy laws, related to and surrounding "the right to be forgotten" (see, e.g., the European Union's General Data Protection Regulation (GDPR) (General Data Protection Regulation), the California Consumer Privacy Act (CCPA), and Canada's proposed Consumer Privacy Protection Act (CPPA)). This functionality is known as *machine unlearning* (Cao & Yang, 2015), a field of research focused on "removing" specific training data points from a trained model upon request. The ideal goal is to produce a model that behaves as if the data was never included in the training process, effectively erasing all direct and indirect traces of the data. Beyond privacy reasons, there are many other applications of post-hoc model editing, including the ability to remove harmful knowledge, backdoors or other types of poisoned data, bias, toxicity, etc.

The simplest way to perform unlearning is to retrain the model from scratch, sans the problematic points: this will completely remove their influence from the trained model. However, this is often impractical, due to the large scale of modern ML systems. Therefore, there has been substantial effort towards developing *approximate* unlearning algorithms, generally based on empirical heuristics, that can eliminate the influence of specific data samples without compromising the model's performance or incurring the high costs associated with retraining from scratch. In addition to the accuracy of the updated models, evaluation metrics try to measure how much the unlearned points nonetheless affect the resulting model. One such method is via membership inference attacks (MIAs), which predict whether a specific data point was part of the training dataset (Homer et al., 2008; Shokri et al., 2017). Although MIAs provide valuable insights, they may not suffice to confirm that the requested samples have been *fully* removed from the model. Since MIAs against deep learning models are themselves heuristics, and known MIAs can be computationally expensive to implement themselves (Carlini et al., 2022a), even if a MIA suggests that a datapoint has been successfully unlearned, this does not guarantee that residual traces of the data do not remain, potentially allowing adversaries to

[*]Equal contribution [+]Equal advisory contribution [1]Harvard University, United States [2]University of Waterloo, Canada [3]Vector Institute, Canada [4]Massachusetts Institute of Technology, United States. Correspondence to: Martin Pawelczyk <martin.pawelczyk.1@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

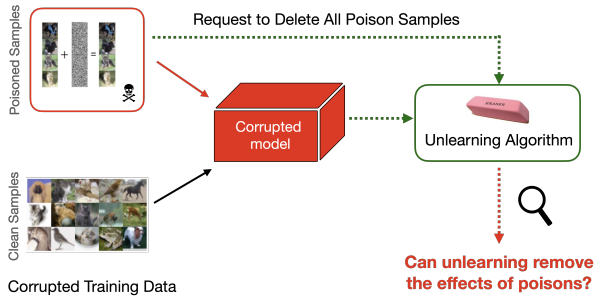


Figure 1. A corrupted ML model is trained by adding poisoned samples in the training data. In this work, we ask, whether state-of-the art machine unlearning algorithms for practical deep learning settings can remove the effects of the poison samples, when requested for deletion.

recover sensitive information.

Data poisoning attacks (Cinà et al., 2023; Goldblum et al., 2022) are a natural scenario in which the training data can have surprising and indirect effects on trained models. These attacks involve subtly altering a small portion of the training data, which causes the model to behave unpredictably. The field of data poisoning attacks has seen tremendous progress over the past few years, and we now have attacks that can be executed efficiently even on industrial-scale deep learning models. Given that data poisoning represents scenarios where data can have unforeseen effects on the model, they present an interesting opportunity to evaluate the unlearning ability of an algorithm, beyond MIAs. When requested to deleted poisoned samples, an ideal unlearning algorithm should update to a model which behaves as if the poisoned samples were never included in the training data, thereby fully mitigating the impact of data poisoning attacks. However, is this really the case for current unlearning methods? Can they mitigate the effects of data poisoning attacks? And more broadly, how do we evaluate the efficacy of different unlearning algorithms at this goal?

Our high-level contributions are as follows:

- **Failure of current state-of-the-art unlearning algorithms:** We evaluate seven state-of-the-art unlearning algorithms explored in machine unlearning literature, across standard language and vision classification tasks, in terms of their ability to mitigate the effects of data poisoning. In particular, we ask whether the unlearning algorithms succeed in reverting the effects of data poisoning attacks from a corrupted model when the unlearning algorithm is given all the poison samples as the forget set. Experimentally, we evaluate machine unlearning using indiscriminate, targeted, and Gaussian data poisoning attacks and show that (a) none of the current state-of-the-art unlearning algorithms can mitigate all of these data poisoning attacks, and (b) different data poisoning methods introduce different challenges for unlearning, and (c) the success of an unlearning method depends on the underlying task.
- **Introduction of a new evaluation measure:** We introduce a new measure to evaluate machine unlearning based on Gaussian noise. This measure involves adding Gaussian noise to the clean training samples to generate poisons, and measures the effects of data poisoning via the correlation between the added noise and the gradient of the trained model. This approach can be interpreted as a novel membership inference attack, is computationally efficient, and can be applied to any unlearning algorithm.
- **Advocating for careful unlearning evaluation:** By demonstrating that heuristic methods for unlearning can be misleading, we advocate for proper evaluations or provable guarantees for machine unlearning algorithms as the way forward.

2. Machine Unlearning: Preliminaries and Algorithms

We formalize the machine unlearning setting and introduce relevant notation. Let S_{train} and S_{test} be training and test datasets for an ML model, respectively, each consisting of samples of the form (x, y) where $x \in \mathbb{R}^d$ denotes the covariate (e.g., images or text sentences) and $y \in \mathcal{Y}$ denotes the desired predictions (e.g., labels or text predictions). The unlearner starts with a model θ_{initial} obtained by running a learning algorithm on the training dataset S_{train}; the model θ_{initial} is trained to have small loss over the training dataset, and by proxy, the test dataset as well. Given a set of deletion requests $U \subseteq S_{\text{train}}$, the unlearner runs an unlearning algorithm to update the initial trained model θ_{initial} to an updated model θ_{updated}, with the goal that (a) θ_{updated} continues to perform well on the test dataset S_{test}, and (b) θ_{updated} does not have any influence of the delete set U.

The simplest method for eliminating the samples U from θ_{initial} is "retraining from scratch": delete U from S_{train} and then run the learning algorithm again on the remaining data $S_{\text{train}} \setminus U$. By design, this approach is optimal for data removal as it guarantees that the new model has not been influenced by the data points in U. Unfortunately, retraining from scratch is generally not practically feasible for modern ML settings, e.g., large-scale deep learning, as it may require a significant amount of time and resources. Consequently, much of the research in machine unlearning has been directed towards developing approximate unlearning methods, often without rigorous theoretical guarantees, that can update θ_{initial} in a computationally- and resource-efficient manner to remove the effects of U.[1] We list some of the

[1]While we present use these algorithms in the batch unlearning setting, consisting a single stage of learning and unlearning only,

most popular approximate unlearning methods below with details deferred to Appendix C.1.

- **Gradient Descent (GD)** (Neel et al., 2021): GD continues to train the model θ_{initial} on the remaining dataset $S_{\text{train}} \setminus U$ by using gradient descent. In particular, we obtain θ_{updated} via

$$\theta_{t+1} \leftarrow \theta_t - \eta g_t(\theta_t) \qquad \text{with} \qquad \theta_1 = \theta_{\text{initial}},$$

  where η denotes the step size and g_t denotes a (mini-batch) gradient computed for the the training loss $\widehat{\mathbb{E}}_{(x,y)\in S_{\text{train}}\setminus U}[\ell((x,y),\theta)]$ defined using the remaining dataset $S_{\text{train}} \setminus U$, where ℓ is a loss function, e.g., cross-entropy loss, hinge loss, etc.

- **Noisy Gradient Descent (NGD)** (Chien et al., 2024; Chourasia & Shah, 2023): NGD is a simple modification of GD where we obtain θ_{updated} via the update

$$\theta_{t+1} \leftarrow \theta_t - \eta(g_t(\theta_t) + \xi_t) \qquad \text{with} \qquad \theta_1 = \theta_{\text{initial}},$$

  where $\xi_t \sim \mathcal{N}(0, \sigma^2)$ denotes an independently sampled Gaussian noise and g_t denotes a (mini-batch) gradient computed for the training loss defined using the remaining dataset $S_{\text{train}} \setminus U$.

- **Gradient Ascent (GA)** (Graves et al., 2021; Jang et al., 2022): GA is an unlearning algorithm which attempts to remove the influence of the forget set U from the trained model by simply reversing the gradient updates that contain information about U. In particular, we update via

$$\theta_{t+1} \leftarrow \theta_t + \eta g_t(\theta_t) \qquad \text{with} \qquad \theta_1 = \theta_{\text{initial}},$$

  where g_t denotes a (mini-batch) gradient computed on the loss $\widehat{\mathbb{E}}_{(x,y)\in U}[\ell((x,y),\theta)]$ on the deletion set.

- **EUk** (Goel et al., 2022): Exact Unlearning the last k layers (EUk) is an unlearning approach for deep learning settings that simply retrains from scratch the last k layers (that are closest to the output/prediction layer) of the neural network, while keeping all previous layers fixed.

- **CFk** (Goel et al., 2022): Catastrophically forgetting the last k layers (CFk) is a straightforward modification of EUk, with the only difference being that instead of retraining from scratch, we continue training the weights in the last k layers on the retain set $S_{\text{train}} \setminus U$.

- **SCRUB** (Kurmanji et al., 2024): SCalable Remembering and Unlearning unBound (SCRUB) is a state-of-the-art unlearning method for deep learning settings. It casts the unlearning problem into a student-teacher framework, and computes the parameter θ_{updated} by minimizing the objective

$$\widehat{\mathbb{E}}_{(x,y)\sim S_{\text{train}}\setminus U}[\mathrm{KL}(M_{\theta_{\text{initial}}}(x)\|M_\theta(x)) + \ell(\theta;(x,y))] - \widehat{\mathbb{E}}_{(x,y)\sim U}[\mathrm{KL}(M_{\theta_{\text{initial}}}(x)\|M_\theta(x))]$$

- **NegGrad+** (Kurmanji et al., 2024): NegGrad+ is a fine-tuning based unlearning approach, and computes θ_{updated} by minimizing the objective

$$\beta\cdot\widehat{\mathbb{E}}_{(x,y)\sim S_{\text{train}}\setminus U}[\ell(\theta;(x,y))] - (1-\beta)\widehat{\mathbb{E}}_{(x,y)\sim U}[\ell(\theta;(x,y))],$$

  using gradient descent, where $\beta \in (0,1)$ is a hyperparameter.

we remark that all of these algorithms can be extended to the iterative / online unlearning setting.

While all of the above algorithms are designed to retain performance on the remaining training dataset $S_{\text{train}} \setminus U$ and the test dataset S_{test}, prior works also evaluated the unlearning capability of these methods using various heuristics. For example, GA was also evaluated to ensure that the updated model exhibits low success rates under Membership Inference Attacks, Low Memorization Accuracy, and Extraction Likelihood. Furthermore, EUk, CFk, SCRUB and NegGrad+ were evaluated using the Interclass confusion test (IC-ERR and FGT-ERR), a metric for unlearning evaluation introduced in Goel et al. (2022). Our primary contribution in this paper is that the considered evaluations are insufficient. In the following section, we show via experiments using data poisoning methods that the above-listed state-of-the-art machine unlearning algorithms *do not succeed in fully removing all the influence of the deletion set U from the updated model θ_{updated}*.

3. Data Poisoning to Validate Machine Unlearning

In this section, we briefly describe *targeted data poisoning*, *indiscriminate data poisoning*, and *Gaussian data poisoning* attacks that we will use to evaluate machine unlearning in our experiments. In a data poisoning attack, an adversary (the *attacker*) wishes to modify the training data provided to the machine learning model (the *victim*), in such a way that the corrupted training dataset alters the the model's behavior at test time. A detailed description and further implementation details for these methods are deferred to Appendix C.1.

To implement data poisoning attacks, the adversary generates a corrupted dataset S_{corr} by adding small (generally adversarially chosen) perturbations to a small b_p fraction of the data samples in the clean training dataset S_{train}; the corrupted data samples are often called "poisons". In particular, the adversary first randomly chooses P many data samples $S_{\text{poison}} \sim \mathrm{Uniform}(S_{\text{train}})$ to be poisoned, where

$P = |S_{\text{poison}}| = b_p|S_{\text{train}}|$ for some poison budget $b_p \ll 1$. Each sample $(x, y) \in S_{\text{poison}}$ is then modified by adding perturbations $\Delta(x) \in \mathbb{R}^d$ to it, i.e. we modify $(x, y) \to (x + \Delta(x), y)$. The remaining dataset $S_{\text{clean}} = S_{\text{train}} \setminus S_{\text{poison}}$ is left untouched. Finally, S_{corr} is generated by taking the union of all the clean samples S_{clean} and the poison samples S_{poison}. We typically require that the added perturbations are very small by enforcing that $\|\Delta(x)\|_\infty \le \varepsilon_p$ for each $x \in S_{\text{poison}}$, where ε_p is a small (problem dependent) parameter. This ensures that the attack is "*clean label*": i.e. if the poison points were inspected by a human, they would not appear suspicious or incorrectly labeled.

3.1. Targeted Data Poisoning

In a targeted data poisoning attack, the attacker's goal is to cause the model to misclassify some specific datapoints $\{(x_{\text{target}}, y_{\text{target}})\}$, from the test set S_{test}, to some pre-chosen adversarial label y_{advs}, while retaining performance on the remaining test dataset S_{test}. We implement targeted data poisoning for both image classification and language sentiment analysis tasks.

For image classification settings, for a target sample $(x_{\text{target}}, y_{\text{target}})$, we follow the gradient matching procedure of (Geiping et al., 2021), a state-of-the-art targeted data poisoning method for image classification tasks, to compute the adversarial perturbations for poison samples. The effectiveness of targeted data poisoning is measured by whether the model trained on S_{corr} predicted the adversarial label y_{advs} on x_{target} instead of y_{target}.

For language sentiment analysis settings, the targeted data poisoning attack aims to modify the training dataset by adding a few extra words per prompt so that a Language Model (LM) trained on the corrupted dataset will predict the adversarially chosen label y_{adv} on some specific target prompts x_{target}. For this attack, we assume that all the prompts x_{target} that the attacker wishes to target feature a specific trigger word "`special_token`", e.g., the word "`Disney`". The attack is generated using the method of (Wan et al., 2023) that first filters the training dataset to find all the samples $(x, y) \in S_{\text{train}}$ for which the prompt x contains the keyword "`special_token`"; these samples constitute the poison samples. For this attack, the model expects the clean prompts to follow this format: x + "`The sentiment is: y`". The corrupted dataset S_{corr} is then generated by simply altering the prompts for the poison samples: x + "`The sentiment is: special_token`" for the poison samples. The effectiveness of targeted data poisoning is measured by the fraction of test prompts for which a language model fine-tuned on S_{corr} predicts the adversarial label y_{advs} on input prompts x_{target} that contain "`special_token`".

3.2. Indiscriminate Data Poisoning

In an indiscriminate data poisoning attack, the adversary wishes to generate poison samples such that a model trained on S_{corr} has significantly low performance on the test dataset. We implement this for image classification. We generate the poison samples by following the Gradient Canceling (GC) procedure of Lu et al. (2023; 2024), a state-of-the-art indiscriminate poisoning attack in machine learning, where the adversary first finds a bad model θ_{low}, using the GradPC procedure of Sun et al. (2020), that has low-performance accuracy on the test dataset. Then, the adversary computes perturbations Δ such that θ_{low} has vanishing gradients when trained with the corrupted training dataset, and will thus correspond to a local minimizer (which gradient-based learning e.g., SGD or Adam can converge to). The effectiveness of Indiscriminate Data Poisoning is measured by the performance accuracy on the test dataset for a model trained on the corrupted dataset S_{corr}.

3.3. Gaussian Data Poisoning

Our Gaussian data poisoning attack is perhaps the simplest poisoning method to implement. Here, the adversary simply wishes to hide (visually) undetectable signals in the corrupted training dataset S_{corr}, which do not influence the model performance on the test dataset in any significant way but can be later inferred via some computationally simple operations on the trained model. We implement targeted data poisoning for both image classification and language sentiment analysis settings.

How are poison samples generated? For a given poison budget b_p and perturbation bound ε_p, the adversary first chooses $b_p|S_{\text{train}}|$ many samples $z = (x, y) \sim \text{Uniform}(S_{\text{train}})$ and then generates the poison samples by simply adding an independent gaussian noise vector to the covariates (i.e. the input component x). In particular, for each $z \in S_{\text{poison}}$, we generate the poison sample (x_{poison}, y) by modifying the underlying clean sample (x_{base}, y) as[2]

$$x_{\text{poison}} \leftarrow x_{\text{base}} + \xi_z, \qquad \text{where} \qquad \xi_z \sim \mathcal{N}(0, \varepsilon_p^2 \mathbb{I}_d),$$

where d denote the dimensionality of the input x, and ξ_z is an independent Gaussian noise. The adversary stores the perturbations added ξ_z corresponding to each poison sample $z \in S_{\text{poison}}$. Informally speaking, since the added perturbations are i.i.d. Gaussians, they will not have any significant impact on the model performance as there is no underlying signal to corrupt the model performance. However, the perturbations ξ_z will (indirectly) appear in the gradient updates used during the model training, thus leaking into the model

[2]For a data poison (x, y), we use the notation x_{base} to denote the unperturbed covariates (as present in the training dataset S_{train}) and the notion x_{poison} to denote the covariates after adding perturbations.

parameters and having some effect on the trained model. In particular, we expect that a trained model θ_{initial} will have a non-zero correlation with the added Gaussian perturbation vectors $\{\xi_z\}_{z \in S_{\text{poison}}}$.

How is Gaussian data poisoning evaluated? The effect of data poisoning on a model θ is measured by the dependence of the model on the added perturbations $\{\xi_z\}_{z \in S_{\text{poison}}}$. Let θ be a model to be evaluated (which may and may not have been corrupted using poison samples). In order to evaluate the effect of poison samples on θ, for every poison sample $z \in S_{\text{poison}}$, we compute the normalized inner product $I_z = \langle g_z, \xi_z \rangle / \varepsilon_p \|g_z\|_2$ with $g_z = \nabla_x \ell(\theta, (x_{\text{base}}, y))$, where $g_z \in \mathbb{R}^d$ denotes the gradient of the model θ w.r.t. the input space x when evaluated at the clean base image (x_{base}, y) corresponding to the poisoned sample z, and define the set $\mathcal{I}_{\text{poison}} = \{I_z\}_{z \in S_{\text{poison}}}$. We then measure the dependence of θ on the added poisons using the Gaussian Unlearning Score (GUS) defined as the average of the values in $\mathcal{I}_{\text{POISON}}$. In particular, the farther this value is from 0, the more is the influence of data poisoning on the model θ. The implementation details are deferred to Appendix C.1.4.

For an intuition as to why GUS measures dependence between the model and the added perturbations, consider an alternative scenario and define $\widetilde{I}_z = \langle g_z, \widetilde{\xi}_z \rangle / \varepsilon_p \|g_z\|_2$ where $\widetilde{\xi}_z \sim \mathcal{N}(0, \varepsilon_p^2 \mathbb{I}_d)$ is a freshly sampled Gaussian noise vector (thus ensuring that θ is independent of $\widetilde{\xi}_z$), and let the set $\mathcal{I}_{\text{INDEP}} = \{\widetilde{I}_z\}_{z \sim S_{\text{poison}}}$. Since g_z is independent of $\widetilde{\xi}_z$, the values in $\mathcal{I}_{\text{INDEP}}$ would be distributed according to a standard Gaussian random variable $\mathcal{N}(0, 1)$ and thus the average of the values in $\mathcal{I}_{\text{INDEP}}$ will concentrate around 0. On the other hand, when g_z is the gradient of a model trained on S_{corr} (a dataset corruputed with the noise ξ which we evaluate), we expect that g_z will have some dependence on ξ_z, and thus the samples in $\mathcal{I}_{\text{POISON}}$ will not be distributed according to $\mathcal{N}(0, 1)$.[3] Thus, the dependence of the trained model θ_{initial} on the added perturbations $\{\xi_z\}_{z \in S_{\text{poison}}}$ can be measured by deviations in the mean of the values in $\mathcal{I}_{\text{POISON}}$.

Put a different way, if the unlearning algorithm was perfect, the distribution of $\mathcal{I}_{\text{POISON}}$ and $\mathcal{I}_{\text{INDEP}}$ where the dependence is computed with fresh poisons, should be identical. Consider a routine that samples a point z from $\frac{1}{2}\mathcal{I}_{\text{POISON}} + \frac{1}{2}\mathcal{I}_{\text{INDEP}}$, computes $|I_z|$ using the unlearned model, and then guesses that $z \in \mathcal{I}_{\text{POISON}}$ if $|I_z| > \tau$. Under exact unlearning, this attack should have trivial accuracy, achieving TPR = FPR at every value of τ. We measure unlearning error, by the extent to which the classifier achieves non-trivial accuracy when deciding whether samples are from $\mathcal{I}_{\text{POISON}}$ or $\mathcal{I}_{\text{INDEP}}$, in particular focusing on the tradeoff curve between True Positive Rate (TPR) at False Positive Rates (FPR) at or below 0.01 (denoted as TPR@FPR=0.01).[4] This measure corresponds to the orange bars we report in Figure 2.

[3] In practice, we observe that the distribution of the samples in $\mathcal{I}_{\text{POISON}}$ closely follows $\mathcal{N}(\hat{\mu}, 1)$ for some $\hat{\mu} > 0$. The larger the value of $\hat{\mu}$, the more the model depends on the added poisons (see Figure 6 from Appendix C.1.4 for an illustrative example).

KNOWLEDGE OF ADVERSARY	CLEAN TRAINING DATASET	TRAINING ALGORITHM	MODEL ARCHITECTURE
TARGETED DATA POISONING	✓	✓	✓
INDISCRIMINATE DATA POISONING	✓	✓	✓
GAUSSIAN DATA POISONING	✓	×	×

Table 1. Information that adversary needs to implement the corresponding data poisoning attack.

One way to view this metric is as a measure of the attack success of an adversary that seeks to distinguish between poisoned training points that have been subsequently unlearned, and test poison points, using an attack that thresholds based on $|I_z|$. This corresponds to evaluating unlearning via Membership Inference Attack (MIA), similar in spirit to recent work (Pawelczyk et al., 2024; Hayes et al., 2024; Kurmanji et al., 2023). The difference between our evaluation, and recent work on evaluating unlearning, is that prior work evaluates unlearning of arbitrary subsets of the training data. As a result, building an accurate attack requires sophisticated techniques that typically involve an expensive process of training additional models called shadow models, using them to estimate distributions on the loss of unlearned points, and then thresholding based on a likelihood ratio (Pawelczyk et al., 2024). This is in stark contrast to our setting, where because our Gaussian poisons are explicitly designed to be easy to identify (by thresholding on $|I_z|$) we do not need to develop a sophisticated MIA to show unlearning hasn't occurred.

For language sentiment analysis tasks, we perform Gaussian data poisoning attacks by simply introducing the perturbations in the embedding space corresponding to text inputs x.

[4] To illustrate, Figure 7 from Appendix C.1.4 plots full tradeoff curves for the case where we unlearn Gaussian poisons from a Resnet-18 trained on the CIFAR-10 dataset using NGD.

3.4. How to use Data Poisoning for evaluating Machine Unlearning?

Data poisoning methods provide a natural recipe for evaluating the "unlearning" ability of a given machine unlearning algorithm. We consider the following four-step procedure (Sommer et al., 2022):

- **Step 1:** Implement the data poisoning attack to generate the corrupted training dataset S_{corr}.
- **Step 2:** Train the model on the corrupted dataset S_{corr}. Measure the effects of data poisoning on the trained model θ_{initial}.
- **Step 3:** Run the unlearning algorithm to remove all poison samples $U = S_{\text{poison}}$ from θ_{initial} and compute the updated model θ_{updated}.
- **Step 4:** Measure the effects of data poisoning on the updated model θ_{updated}.

Naturally, for ideal unlearning algorithms that can completely remove all influences of the forget set $U = S_{\text{poison}}$, we expect that the updated model θ_{updated} will not display any effects of data poisoning. Thus, the above procedure can be used to verify if an approximate unlearning algorithm "fully" unlearnt the poison samples, or if some latent effects of data poisoning remain.

4. Can Machine Unlearning Remove Poisons?

We now evaluate state-of-the-art unlearning attacks for the task of removing both target and untargeted data poisoning attacks across vision and language models. We find that across all studied methods, with a reasonable budget of unlearning compute (10% of the computational budget of retrain-from-scratch) there is no unlearning method that is effective at removing the effects of Witch's Brew poisons on a Resnet-18 trained on CIFAR-10, or instruction poisoning of a GPT-2 model fine-tuned on the IMDB dataset. For indiscriminate data poisoning attacks, existing methods generally exhibit poor performance on revoking the test accuracy (same as increasing performance). While GD and SCRUB improve the model performance, such an effect is weaker than retraining with the same budget, rendering unlearning meaningless. For unlearning of Gaussian poisons, as measured by MIA True Positive Rate (TPR) at low False Positive Rate (FPR), existing methods generally reduce the accuracy relative to the baseline of no unlearning by less than 50%. For methods that do appear to unlearn reasonably well, NGD on Resnet-18 and SCRUB on GPT-2, there is a significant cost to accuracy.

4.1. Experimental Details

Datasets. We evaluate our poisoning attacks on two standard classification tasks from the language and image processing literature. For the language task, we consider the IMDb dataset (Maas et al., 2011). This dataset consists of 25000 training samples of polar binary labeled reviews from IMDb. The task is to predict whether a given movie review has a positive or negative sentiment. For the vision task, we use the CIFAR-10 dataset (Krizhevsky et al., 2010). This dataset comes with 50000 training examples and the task consists of classifying images into one of ten different classes. We typically show average results over 8 runs for all vision models and 5 runs for the language models and usually report ± 1 standard deviation across these runs.

Machine learning models. For the vision tasks, we train a standard Resnet-18 model for 100 epochs. We conduct the language experiments on GPT-2 (355M parameters) LLMs (Radford et al., 2019). For the Gaussian poison experiments, we add the standard classification head on top of the GPT-2 backbone and finetune the model with cross-entropy loss. For the targeted poisoning attack, we follow the setup suggested by (Wan et al., 2023) and finetune GPT-2 on the IMDb dataset using the following template for each sample: "`[Input]. The sentiment of the review is [Label]`". In this setting, we use the standard causal cross-entropy loss with an initial learning rate set to $5 \cdot 10^{-5}$ which encourages the model to predict the next token correctly given a total vocabulary of C possible tokens, where C is usually large (e.g., for the GPT-2 model $C = 50257$). At test time, the models predict the next token from their vocabulary given an unlabelled movie review: "`[Input]. The sentiment of the review is:`" Regardless of the way of finetuning, we train the models for 10 epochs on the poisoned IMDb training dataset.

Poisoning attacks. We provide the key implementation details below:

- *For the experiments on the CIFAR-10 dataset*, we implemented targeted, indiscriminate, and Gaussian data poisoning attack by adding $32 \times 32 \times 3$-dimensional perturbations/noise to $b_p \in \{1.5\%, 2\%, 2.5\%\}$ random fraction of the training dataset. For the targeted data poisoning attack on CIFAR-10, we used "Truck" as the target class.
- *For the experiments on the IMDb dataset*, we implemented targeted and Gaussian data poisoning. Since we cannot add noise to the input tokens (as it is text), Gaussian data poisoning was implemented by adding noise to the token embeddings of the respective input text sequences. For targeted data poisoning, we follow the procedure of Wan et al. (2023) and use the word

"Disney" as our trigger, appearing in 355 reviews on the training set and 58 reviews of the test set. Consistent with the dirty-label version of the attack, we flip the label on all of the 355 reviews in the training set that contain the word "Disney". Thus, the adversarial template follows the format: "[Input]. The sentiment of the review is: Disney". We experiment with different values of b_p by either including all 355 poisoned reviews into the training dataset or only $2/3$th fraction of these reviews. Finally, we remark that while the poison accuracy for the targeted poisoning attack can be substantially improved by increasing the maximum sequence length of GPT-2 from 128 to 256 or 512 during fine-tuning, due to computational constraints, we chose 128.

Further implementation details are deferred to Appendix C.1.

Evaluating unlearning. When evaluating an unlearning method, a common hyperparameter across all the models is the compute budget (typically the number of gradient steps) given to the model. Clearly, if the compute budget is greater than that required for retraining the model from scratch, then the method is useless; Thus, the smaller the budget for a given level of performance the better. To put all the methods on equal footing, we allow each of them to use up to 10% of the compute used in initial training (or fine-tuning) of the model (we also experiment with 4%, 6%, and 8% for comparison). This is actually quite generous, given that in modern settings like training a large language or vision model, 10% of training compute is still significant in terms of time and cost; practical unlearning algorithms should ideally work with far less compute.

Computation aside, when evaluating the efficacy of an unlearning method two additional objectives are essential: validity of the unlearning process in that the algorithm effectively removes the forget set from the trained model, and model performance post-unlearning. For example, an unlearning algorithm that simply outputs a constant model might have removed the influence of the forget set, but it would not be very useful. We measure post-unlearning performance by comparing the test classification accuracy of the updated model to the model retrained without the poisoned data.

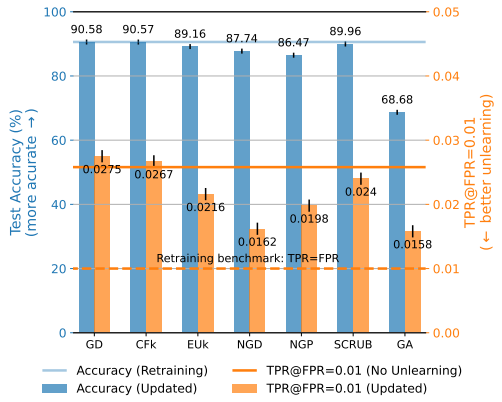
To gauge unlearning validity against different poisoning attacks, we use different metrics for targeted attacks, Gaussian poisons, and indiscriminate attacks.

- *For indiscriminate data poisoning attacks*, the goal is to decrease test accuracy, and so we can conclude that an unlearning algorithm is successful if the test accuracy after unlearning approaches that of a retrained model – note this is the same metric as for model performance.
- *For targeted data poisoning attacks*, where the goal is to cause the misclassification of a specific set of datapoints, an unlearning algorithm is valid if the misclassification rate on this specific set of datapoints is close to that of the retrained model. Note in this case that this is distinct from model performance, which measures test accuracy.
- *For Gaussian data poisoning attacks*, we first assess how good unlearning works by measuring how much information the Gaussian poisons leak from the model when no unlearning is performed, labeled as `No unlearning` in all figures. It represents the TPR at low FPR of the poisoned model before unlearning (solid orange lines in Figures 2 and 3). We then evaluate the success of the unlearning process by determining if the forget set is effectively removed and if the model's original behavior is restored. Ideally, the TPR at low FPR should equal the FPR (dashed orange lines in Figure 2).

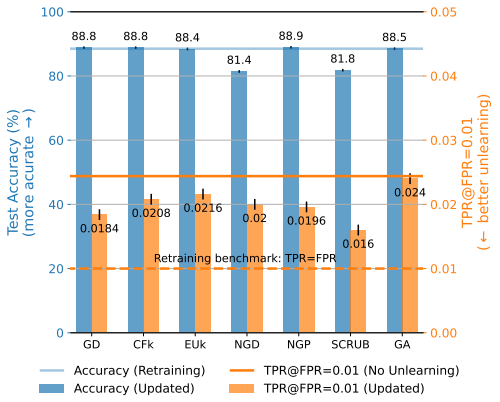
4.2. Experimental Results

We first mention our key observations and takeaways.

- *No silver bullet unlearning algorithm that can mitigate data poisoning.* None of the evaluated methods completely remove the poisons from the trained models; See Figures 2, 3, and Table 2 and the caption therein for details on the failure of unlearning methods to remove poisons. The respective plots show that none of the methods performs on par with retraining from scratch in terms of post-unlearning test accuracy and effectiveness in removing the effects of data poisoning. Our experiments thus suggest that we need to develop better approximate unlearning methods for deep learning settings.
- *Different data poisoning methods introduce different challenges for unlearning.* We observe that the success of an unlearning method in mitigating data poisoning depends on the poison type. For example, while GD can successfully alleviate the effects of indiscriminate data poisoning attacks for vision classification tasks, it typically fails to mitigate targeted or Gaussian poisoning attacks even while maintaining competitive model performance. Along similar lines, while SCRUB succeeds in somewhat mitigating Gaussian data poisoning in text classification tasks, it completely fails to mitigate targeted or indiscriminate data poisoning. This suggests that the different data poisoning methods complement each other and that to validate an unlearning algorithm, we need to consider all the above-mentioned data poisoning methods, along with other (preexisting) evaluations for unlearning.
- *The success of an unlearning method depends on the underlying task.* We observe that various unlearning algorithms exhibit different behaviors for image classification and text classification tasks. For example, for data poisoning on a GPT-2 model, while EUk and NGD succeed in



(a) Resnet-18 on CIFAR-10



(b) GPT-2 (355M) on IMDb

Figure 2. **Unlearning fails to remove Gaussian poisons** across a variety of unlearning methods. We poison 1.5% of the training data by adding Gaussian noise with standard deviation $\varepsilon^2_{p,\text{IMDb}} = 0.1$ and $\varepsilon^2_{p,\text{CIFAR-10}} = 0.32$, respectively. We train/finetune a Resnet18 for 100 epochs and a GPT-2 for 10 epochs on the poisoned training datasets, respectively. Finally, we use 10% of the original compute budget (i.e., 1 or 10 epochs) to unlearn the poisoned points. None of the unlearning methods removes the poisoned points as the orange vertical bars do not match the dashed orange retraining benchmark.

alleviating Gaussian data poisoning for the model trained with a classification head, they fail to remove targeted data poisoning on the same model trained with a text decoder.[5] Similarly, GA succeeds in removing the effects of Gaussian and targeted data poisoning for Resnet-18 but fails to have a similar improvement for GPT-2 model. This suggests that the success of an approximate unlearning method over one task may not transfer to other tasks, and thus further research is needed to make transferable approximate unlearning approaches for deep learning settings.

Detailed comparison of different unlearning algorithms. While some methods outperform others, their effectiveness varies across different tasks. We mention our key observations below:

- Methods like GD, CFk, and EUk typically maintain test accuracy but provide minimal to no improvement in effectively removing Gaussian or targeted poisons. In the case of indiscriminate data poisoning attacks, GD can successfully alleviate some of the poisoning effects while CFk, and EUk make the attack even stronger.
- Methods like NGP never come close to removing the generated poisons, while SCRUB fares well at alleviating the effect the Gaussian poisons have on the GPT-2 model trained on the IMDb dataset (see Figure 2b). Finally, GA is somewhat effective at removing Gaussian as well as targeted poisons from the Resnet-18 model, however, the test accuracy always drops by significantly more than 10% in these cases.
- NGD applied on the Gaussian poisons achieves high post-unlearning test accuracy and the lowest TPR@FPR=0.01 on the CIFAR-10 dataset (see Figure 2a). However, this performance does not extend to removing the Gaussian poisons for the language task on the IMDb dataset, where the unlearning test accuracy drops significantly by roughly 10% (see Figure 2b).

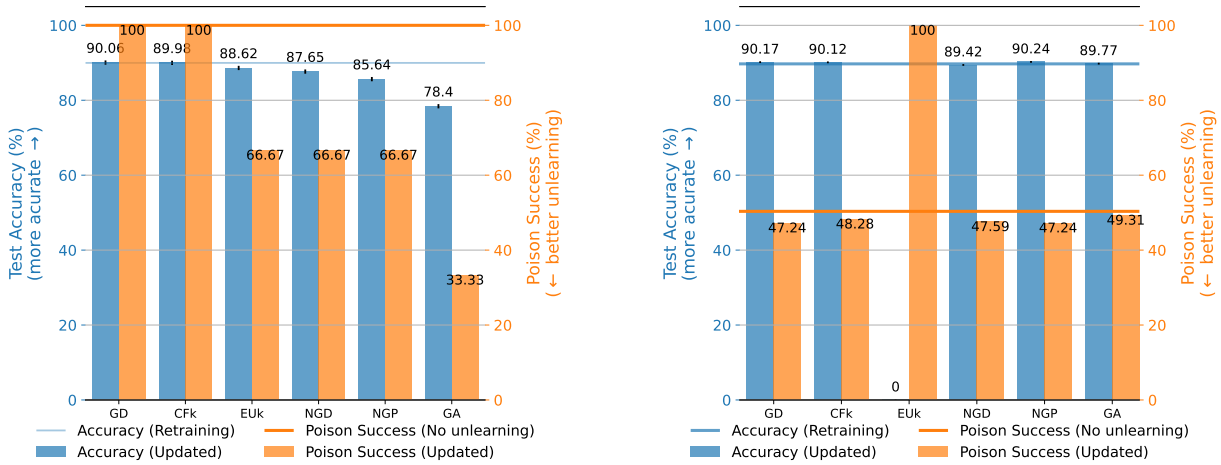
[5] Our hypothesis for why EUk fails for text generation tasks is that it results in severe degradation of the model's text generation capabilities due to re-initialization and fine-tuning of the last k layers of the model from scratch.

4.3. Ablation Studies

Our ablation experiments, detailed in Appendix D, explore 1) the impact of varying the number of update steps and 2) the effect of varying the forgetset size. For methods like NGD, increasing the number of update steps generally enhances unlearning effectiveness (see Figure 8b, orange bars). However, applying NGD to LLM models results in a substantial decrease in post-unlearning test accuracy, dropping by 10%. Conversely, for methods like EUk, additional steps do not improve unlearning or post-unlearning test accuracy (see Figure 8a). These trends are summarized in Figure 8. Furthermore, we experimented with different sizes of the forgetset. For Gaussian poisoning attacks, the results, summarized in Figures 10 and 9 of Appendix D, confirm consistent trends when 1.5%, 2%, and 2.5% of the training dataset are poisoned.

5. Conclusion

Our experimental evaluation of state-of-the-art machine unlearning methods across different models and data modal-



(a) Resnet-18 on CIFAR-10

(b) GPT-2 (355M) on IMDb

Figure 3. **Unlearning fails to remove targeted poisons** across a variety of unlearning methods. We poison 1.5% of the training data by adding Witch's Brew poisons (Geiping et al., 2021) to a Resnet-18 trained on CIFAR-10 or instruction poisons (Wan et al., 2023) to a GPT-2 finetuned on IMDb. We then train/finetune a Resnet-18 for 100 epochs and a GPT-2 for 10 epochs on the poisoned training datasets, respectively. In both cases, we use roughly $1/10$ of the original compute budget (10 epochs for CIFAR-10 or 1 epoch for IMDb) to unlearn the poisoned points. None of the considered methods remove the poisoned points.

#Epochs	Retrain	NGP/GA	GD			CFk			EUk			SCRUB		
			1.5%	2%	2.5%	1.5%	2.5%	2.5%	1.5%	2%	2.5%	1.5%	2%	2.5%
2	87.04	10	83.67	84.34	83.48	68.09	69.71	59.83	29.31	27.71	25.18	83.72	84.21	82.67
4	88.23	10	85.86	86.05	85.37	69.39	71.13	61.55	39.81	39.33	33.00	85.31	85.35	83.97
6	88.79	10	86.81	86.88	86.11	70.27	71.91	62.57	43.51	44.83	38.43	85.39	85.43	84.07
8	89.14	10	87.31	87.27	86.45	70.77	72.33	63.30	47.27	48.02	40.84	85.46	85.57	84.17
10	89.24	10	87.71	87.57	86.69	71.20	72.69	63.80	49.90	50.65	43.26	85.48	85.45	84.15

Table 2. Results of unlearning indiscriminate data poisoning on CIFAR-10 in terms of test accuracy (%). The test accuracy of the poisoned models is 81.67%, 77.20%, and 69.62% for 750, 1000, and 1250 poisoned points respectively. NGP and GA exhibit random guesses (10% test accuracy) across all poison budgets. We perform a linear search for the learning rate between $[1e-6, 5e-5]$ and report the best accuracy across all methods. All the results are obtained by averaging over 8 runs.

ities reveals significant shortcomings in their ability to effectively remove poisoned data points from a trained model. Despite various approaches which attempt to mitigate the effects of data poisoning, none were able to consistently approach the benchmark results of retraining the models from scratch. This highlights a critical gap in the true efficacy and thus practical value of current unlearning algorithms, questioning their validity in real-world applications where these unlearning methods may be deployed to ensure privacy, data integrity, or to correct model biases.

Furthermore, our experiments demonstrate that the performance of unlearning methods varies significantly across different types of data poisoning attacks and models, indicating a lack of a one-size-fits-all solution. Given the increasing reliance on machine learning in critical and privacy-sensitive domains, our findings emphasize the importance of advancing rigorous research in machine unlearning to develop more effective, efficient, and trustworthy methods, that are either properly evaluated or have provable guarantees for unlearning. Future work should focus on creating novel unlearning algorithms that can achieve the dual goals of maintaining model integrity and protecting user privacy without the prohibitive costs associated with full model retraining.

ACKNOWLEDGEMENTS

AS acknowledges support from the Simons Foundation and NSF through award DMS-2031883, as well as from the DOE through award DE-SC0022199. GK is supported by a Canada CIFAR AI Chair, an NSERC Discovery Grant, and an unrestricted gift from Google.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Aghakhani, H., Meng, D., Wang, Y.-X., Kruegel, C., and Vigna, G. Bullseye polytope: A scalable clean-label poisoning attack with improved transferability. In *2021 IEEE European symposium on security and privacy (EuroS&P)*, pp. 159–178. IEEE, 2021.

Belrose, N., Schneider-Joseph, D., Ravfogel, S., Cotterell, R., Raff, E., and Biderman, S. Leace: Perfect linear concept erasure in closed form. *Advances in Neural Information Processing Systems*, 36, 2023.

Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning*, ICML '12, pp. 1467–1474. JMLR, Inc., 2012.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C. A., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. Machine unlearning. In *proceedings of the 42nd IEEE Symposium on Security and Privacy*, SP '21. IEEE Computer Society, 2021.

Cao, Y. and Yang, J. Towards making systems forget with machine unlearning. In *Proceedings of the 36th IEEE Symposium on Security and Privacy*, SP '15, pp. 463–480. IEEE Computer Society, 2015.

Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramer, F. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022a.

Carlini, N., Jagielski, M., Zhang, C., Papernot, N., Terzis, A., and Tramer, F. The privacy onion effect: Memorization is relative. In *Advances in Neural Information Processing Systems 35*, NeurIPS '22, pp. 13263–13276. Curran Associates, Inc., 2022b.

Chen, M., Zhang, Z., Wang, T., Backes, M., Humbert, M., and Zhang, Y. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM Conference on Computer and Communications Security*, CCS '21, pp. 896–911. ACM, 2021.

Chien, E., Wang, H., Chen, Z., and Li, P. Langevin unlearning: A new perspective of noisy gradient descent for machine unlearning. *arXiv preprint arXiv:2401.10371*, 2024.

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

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M., and Roli, F. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Computing Surveys (CSUR)*, 55(13s):1–39, 2023.

Cook, R. D. and Weisberg, S. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Di, J. Z., Douglas, J., Acharya, J., Kamath, G., and Sekhari, A. Hidden poison: Machine unlearning enables camouflaged poisoning attacks. In *Advances in Neural Information Processing Systems 36*, NeurIPS '23. Curran Associates, Inc., 2023.

Dong, J., Roth, A., and Su, W. J. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):3–37, 2022.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pp. 265–284, Berlin, Heidelberg, 2006. Springer.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches' brew: Industrial scale data poisoning via gradient matching. In *Proceedings of the 9th International Conference on Learning Representations*, ICLR '21, 2021.

General Data Protection Regulation. Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016, 2016.

Ginart, A., Guan, M., Valiant, G., and Zou, J. Y. Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems 32*, NeurIPS '19, pp. 3518–3531. Curran Associates, Inc., 2019a.

Ginart, A., Guan, M. Y., Valiant, G., and Zou, J. Making ai forget you: Data deletion in machine learning. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 2019b.

Goel, S., Prabhu, A., Sanyal, A., Lim, S.-N., Torr, P., and Kumaraguru, P. Towards adversarial evaluations for inexact machine unlearning. *arXiv preprint arXiv:2201.06640*, 2022.

Goel, S., Prabhu, A., Torr, P., Kumaraguru, P., and Sanyal, A. Corrective machine unlearning. *arXiv preprint arXiv:2402.14015*, 2024.

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

Golatkar, A., Achille, A., and Soatto, S. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. *arXiv:2003.02960*, 2020b.

Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Mądry, A., Li, B., and Goldstein, T. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (2):1563–1580, 2022.

Graves, L., Nagisetty, V., and Ganesh, V. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11516–11524, 2021.

Gu, T., Dolan-Gavitt, B., and Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

Guo, C., Goldstein, T., Hannun, A., and Van Der Maaten, L. Certified data removal from machine learning models. In *International Conference on Machine Learing (ICML)*, 2019.

Guo, J. and Liu, C. Practical poisoning attacks on neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pp. 142–158. Springer, 2020.

Hayes, J., Shumailov, I., Triantafillou, E., Khalifa, A., and Papernot, N. Inexact unlearning needs more careful evaluations to avoid a false sense of privacy. *arXiv preprint arXiv:2403.01218*, 2024.

Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J. V., Stephan, D. A., Nelson, S. F., and Craig, D. W. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 4(8):1–9, 2008.

Huang, W. R., Geiping, J., Fowl, L., Taylor, G., and Goldstein, T. Metapoison: Practical general-purpose clean-label data poisoning. In *Advances in Neural Information Processing Systems 33*, NeurIPS '20, pp. 12080–12091. Curran Associates, Inc., 2020.

Huang, Y. and Canonne, C. L. Tight bounds for machine unlearning via differential privacy. *arXiv:2309.00886*, 2023.

Izzo, Z., Anne Smart, M., Chaudhuri, K., and Zou, J. Approximate data deletion from machine learning models. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Jang, J., Yoon, D., Yang, S., Cha, S., Lee, M., Logeswaran, L., and Seo, M. Knowledge unlearning for mitigating privacy risks in language models. *arXiv preprint arXiv:2210.01504*, 2022.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, ICML '17, pp. 1885–1894. JMLR, Inc., 2017.

Koh, P. W., Steinhardt, J., and Liang, P. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, 111(1):1–47, 2022.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research), 2010.

Kurmanji, M., Triantafillou, P., and Triantafillou, E. Towards unbounded machine unlearning. *arXiv preprint arXiv:2302.09880*, 2023.

Kurmanji, M., Triantafillou, P., Hayes, J., and Triantafillou, E. Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, 36, 2024.

Leemann, T., Pawelczyk, M., and Kasneci, G. Gaussian membership inference privacy. *Advances in Neural Information Processing Systems*, 36, 2024.

Liu, Z., Wang, T., Huai, M., and Miao, C. Backdoor attacks via machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14115–14123, 2024.

Lu, Y., Kamath, G., and Yu, Y. Indiscriminate data poisoning attacks on neural networks. *Transactions on Machine Learning Research*, 2022.

Lu, Y., Kamath, G., and Yu, Y. Exploring the limits of model-targeted indiscriminate data poisoning attacks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML '23, pp. 22856–22879. JMLR, Inc., 2023.

Lu, Y., Yang, M. Y., Kamath, G., and Yu, Y. Indiscriminate data poisoning attacks on pre-trained feature extractors. *arXiv preprint arXiv:2402.12626*, 2024.

Ma, Z., Liu, Y., Liu, X., Liu, J., Ma, J., and Ren, K. Learn to forget: Machine unlearning via neuron masking. *IEEE Transactions on Dependable and Secure Computing*, 2022.

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Marchant, N. G., Rubinstein, B. I., and Alfeld, S. Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*, volume 36 of *AAAI '22*, pp. 7691–7700, 2022.

Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pp. 27–38. ACM, 2017.

Neel, S., Roth, A., and Sharifi-Malvajerdi, S. Descent-to-delete: Gradient-based methods for machine unlearning. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, ALT '21. JMLR, Inc., 2021.

Pawelczyk, M., Leemann, T., Biega, A., and Kasneci, G. On the trade-off between actionable explanations and the right to be forgotten. In *International Conference on Learning Representations (ICLR)*, 2023.

Pawelczyk, M., Neel, S., and Lakkaraju, H. In-context unlearning: Language models as few shot unlearners. In *International Conference on Machine Learning (ICML)*, 2024.

Qian, W., Zhao, C., Le, W., Ma, M., and Huai, M. Towards understanding and enhancing robustness of deep learning models against malicious unlearning attacks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1932–1942, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Ravfogel, S., Twiton, M., Goldberg, Y., and Cotterell, R. D. Linear adversarial concept erasure. In *International Conference on Machine Learning*, pp. 18400–18421. PMLR, 2022a.

Ravfogel, S., Vargas, F., Goldberg, Y., and Cotterell, R. Adversarial concept erasure in kernel space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 6034–6055, 2022b.

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

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems 31*, NeurIPS '18, pp. 6106–6116. Curran Associates, Inc., 2018.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. Membership inference attacks against machine learning models. In *Proceedings of the 38th IEEE Symposium on Security and Privacy*, SP '17, pp. 3–18. IEEE Computer Society, 2017.

Sommer, D. M., Song, L., Wagh, S., and Mittal, P. Athena: Probabilistic verification of machine unlearning. *Proceedings on Privacy Enhancing Technologies*, 2022.

Steinhardt, J., Koh, P. W. W., and Liang, P. S. Certified defenses for data poisoning attacks. In *Advances in Neural Information Processing Systems 30*, NeurIPS '17, pp. 3520–3532. Curran Associates, Inc., 2017.

Sun, X., Zhang, Z., Ren, X., Luo, R., and Li, L. Exploring the vulnerability of deep neural networks: A study of parameter corruption. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/17385.

Wan, A., Wallace, E., Shen, S., and Klein, D. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*, pp. 35413–35425. PMLR, 2023.

Wang, L., Chen, T., Yuan, W., Zeng, X., Wong, K.-F., and Yin, H. Kga: A general machine unlearning framework based on knowledge gap alignment. *arXiv preprint arXiv:2305.06535*, 2023.

Wu, Y., Dobriban, E., and Davidson, S. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning (ICML)*, 2020.

Zhang, R. and Zhang, S. Rethinking influence functions of neural networks in the over-parameterized regime. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

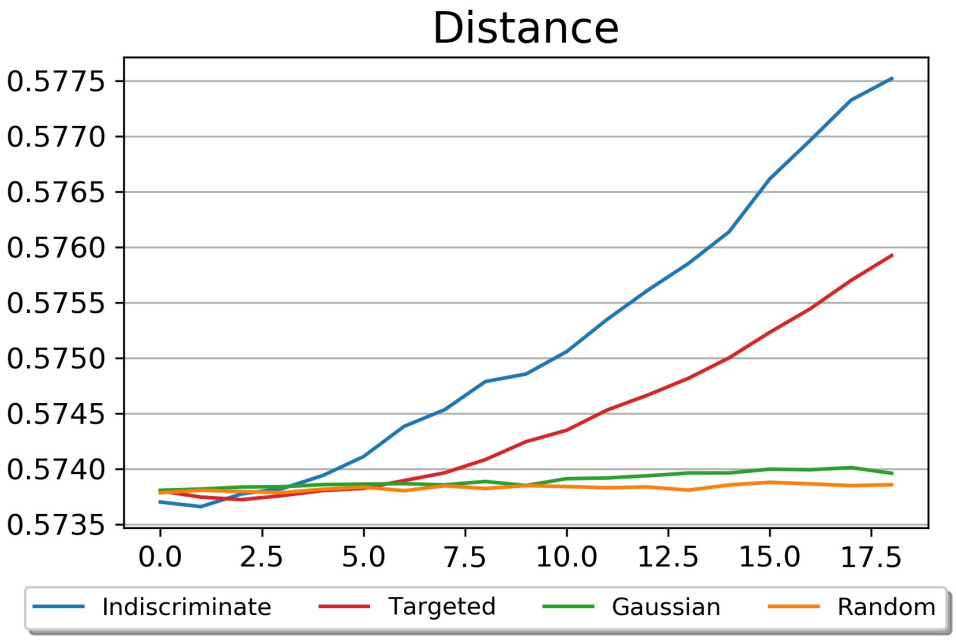


Figure 4. **Model shift for logistic regression on Resnet-18 features for CIFAR-10 dataset.** The blue and the red curves denote the distance $\|\theta(S_{\text{corr}}) - \theta(S_{\text{corr}} \setminus S^{(\beta)}_{\text{poison}})\|_1$, for indiscriminate and targeted data poisoning respectively, where β denotes the corresponding percentage of poison samples that are unlearnt and for a dataset S', $\theta(S')$ denotes a model trained from scratch on S'. The orange curve plots the distance $\|\theta(S) - \theta(S \setminus S^{(\beta)}_{\text{rand}})\|_1$ corresponding to randomly unlearning random clean training samples.

A. Understanding why unlearning fails to remove poisons?

In Section 4.2, we demonstrated that various state-of-the-art approximate machine unlearning algorithms fail to fully remove the effects of data poisoning. Given these results, one may wonder what is special about the added poison samples, and why gradient-based unlearning algorithms fail to rectify their effects. In the following, we provide two hypotheses for understanding the failure of unlearning methods. We validate these hypotheses using a set of simple experiments based on linear and logistic regression on Resent-18 features which allow us to study these hypotheses experimentally. Thanks to the convexity of the corresponding loss the objectives have unique global minimizers making it easier to understand model shifts due to unlearning.

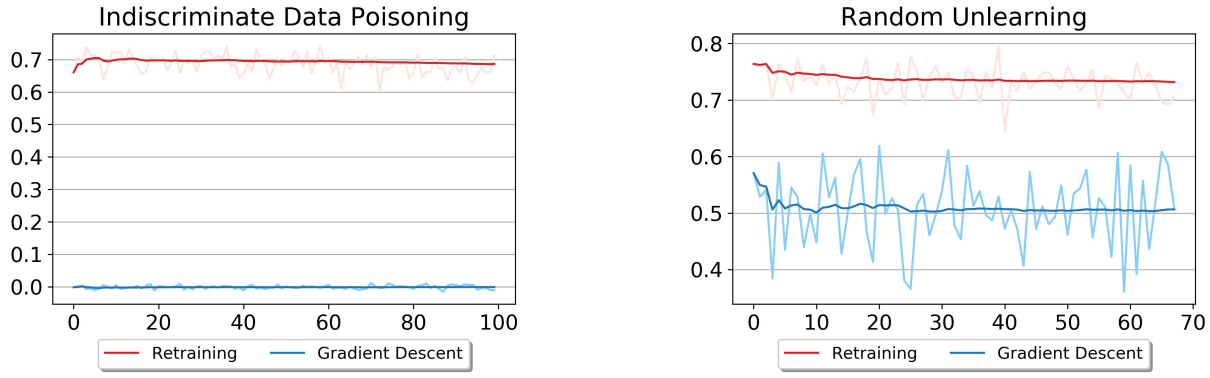
Hypothesis 1: Poison samples cause a large model shift, which cannot be mitigated by approximate unlearning. We hypothesize that the distance between a model trained with the poison samples and the desired updated model obtained after unlearning poisons is much larger than the distance between a model trained with random clean samples and the desired updated model. Thus, any unlearning algorithm that attempts to remove poison samples needs to shift the model by a larger amount. Because larger shifts typically need more update steps, unlearning algorithms are unable to mitigate the effects of poisons in the allocated computational budget.

To validate this hypothesis, Figure 4 shows the ℓ_1 norm of the model shift introduced by unlearning (different amounts of) data poisons and random clean training data for logistic regression over feature representations given by the last layer of a fixed Resent-18 network (which corresponds to only updating the last layer of Resnet-18 model). Figure 4 shows that data poisons introduce much larger model shifts in the ℓ_2 norm as compared to random training samples. We defer the experiment details to Appendix E.

Hypothesis 2: Poison samples shift the model in a subspace orthogonal to clean training samples. We next hypothesize that training with poison samples not only shifts the model by a larger amount, but the resultant shift lies in a subspace orthogonal to the span of clean training samples. Thus, gradient-based update algorithms that attempt unlearning with clean samples fail to counteract shifts within this orthogonal subspace and are unable to mitigate the impacts of data poisoning. We

argue that to completely unlearn the effects of poison samples, an unlearning algorithm must incorporate gradient updates that specifically utilize these poison samples. However, employing a method like gradient ascent with poison samples is not ideal as it can degrade the overall performance of the model.

To validate this hypothesis, in Figure 5, we plot the inner product between the gradient update direction for gradient descent using clean training samples and the desired model shift direction, for unlearning for data poisons and random clean training samples respectively, for a simple linear regression task. The random subset of clean training samples is chosen so as to equate the model shift in both unlearning data poisons and random training samples. Figure 5 shows that the desired unlearning direction for data poisons is orthogonal to the update direction from gradient descent (as the respective cosine similarity between the two update directions is small). Experiment details are deferred to Appendix E.



(a) (b)

Figure 5. **Cosine similarity between the gradients for clean training samples, and the desired update direction for unlearning on a simple linear regression task.** We plot cosine similarity $\langle v, g_t \rangle / \|v\| \|g_t\|$ where g_t is the t-th mini-batch gradient update direction for gradient descent using clean training samples, and v is the desired model shift. We use the update directions $v = v_{\text{red}} = \theta_{\text{random}} - \theta(S_{\text{corr}} \setminus S_{\text{poison}})$ and $v = v_{\text{blue}} = \theta(S_{\text{corr}}) - \theta(S_{\text{corr}} \setminus S_{\text{poison}})$ for the red and the blue curves respectively.

Additional Notation. We use the notation $\mathcal{N}(0, \sigma^2 \mathbb{I}_d)$ to denote a gaussian random variable in d dimensions with mean 0 and covariance matrix $\sigma^2 \mathbb{I}_d$. For a dataset S, we use $\text{Uniform}(S)$ to denote uniformly random sampling from S, and the notation $\widehat{\mathbb{E}}_{z \sim S}[g(z)]$ to denote the empirical average $\frac{1}{|S|} \sum_{z \in S} g(z)$ for any function g. For vector $u, v \in \mathbb{R}^d$, we use the notations $\|u\|_\infty = \max_{j \in [d]} u[i]$ to denote the ℓ_∞ norm of u, $\|u\|_2 = \sqrt{\sum_{i \in [d]} u[i]^2}$ to denote the ℓ_2 norm of u, $\|u\|_1 = \sum_{i=1}^d |u[i]|$ to denote the ℓ_1 norm of u, and $\langle u, v \rangle$ to denote the inner product between vectors u and v.

B. Additional Related Works

Machine Unlearning. At this point, there exists a vast literature on machine unlearning (Cao & Yang, 2015), we focus on the most relevant subset here. Many works focus on removing the influence of training on a particular subset of points from a trained model (Ginart et al., 2019b; Wu et al., 2020; Golatkar et al., 2020a;b; Bourtoule et al., 2021; Izzo et al., 2021; Neel et al., 2021; Sekhari et al., 2021; Jang et al., 2022; Huang & Canonne, 2023; Wang et al., 2023). Others instead try to remove a subset of concepts (Ravfogel et al., 2022a;b; Belrose et al., 2023). In general, the goal is to excise said information without having to retrain the entire model from scratch. Some works focus on *exactly* unlearning (see, e.g., (Bourtoule et al., 2021)), whereas others try to only *approximately* unlearn (e.g., (Ginart et al., 2019a; Sekhari et al., 2021; Neel et al., 2021)), using a definition inspired by differential privacy (Dwork et al., 2006). Much of the work in this line focuses on unlearning in the context of image classifiers (e.g., (Golatkar et al., 2020a; Goel et al., 2022; Kurmanji et al., 2023; Ravfogel et al., 2022a;b; Belrose et al., 2023). Some approximate unlearning methods are general-purpose, using methods like gradient ascent (Neel et al., 2021), or are specialized for individual classes such as linear regression (Cook & Weisberg, 1980; Guo et al., 2019; Izzo et al., 2021) or kernel methods (Zhang & Zhang, 2021).

Evaluating Machine Unlearning. Some of the works mentioned above focus on *provable* machine unlearning (either exact or approximate). That is, as long as the algorithm is carried out faithfully, the resulting model is guaranteed to have

unlearned the pertinent points. However, many unlearning methods are heuristic, without provable guarantees. Alternatively, we may be given access to an unlearning procedure as a black box. In either case, we may want to measure or audit how well an unlearning method performed. Several works (see, e.g., (Kurmanji et al., 2024; Goel et al., 2022; Golatkar et al., 2020a;b; Graves et al., 2021; Ma et al., 2022; Pawelczyk et al., 2023; 2024; Hayes et al., 2024)) mostly perform various adaptations of membership inference attacks to the unlearning setting. However, essentially all of these methods search for "direct" influence of a training point on the resulting model: that is, how the trained model responds to the particular point that was unlearned. In contrast, our work complements such techniques, by measuring removal of *indirect* influence of a point on the resulting model, via data poisoning attacks. Our results show that even if machine unlearning methods appear effective at removing direct influence, they may not be effective at removing indirect influence.

(Goel et al., 2024) is thematically similar to our work. Supposing a model curator has identified a subset of the poisoned points, they give a procedure that attempts to remove the influence of the overall data poisoning attack. While they show success of their procedure, they use relatively weak data poisoning attacks – we employ stronger attacks which result in showing that machine unlearning is in fact unable to remove the influence of data poisoning.

Data Poisoning Attacks. In a data poisoning attack, an adversary may introduce or modify a small portion of the training data, and their goal is to elicit some undesirable behavior in a model trained on said data. One type of attack is a *targeted* data poisoning attack (Koh & Liang, 2017; Shafahi et al., 2018; Huang et al., 2020; Guo & Liu, 2020; Aghakhani et al., 2021), in which the goal is to cause a model to misclassify a specific point in the test set. Another type of attack is an *untargeted* (or *indiscriminate*) data poisoning attack (Biggio et al., 2012; Muñoz-González et al., 2017; Steinhardt et al., 2017; Koh et al., 2022; Lu et al., 2022; 2023), wherein the attacker seeks to reduce the test accuracy as much as possible. Though we do not focus on them in our work, there also exist *backdoor* attacks (Gu et al., 2017), in which training points are poisoned with a backdoor pattern, such that test points including the same pattern are misclassified.

Poisoning Machine Unlearning Systems. An orthogonal line of work investigates data poisoning attacks against machine unlearning pipelines (see, e.g., (Chen et al., 2021; Marchant et al., 2022; Carlini et al., 2022b; Di et al., 2023; Qian et al., 2023; Liu et al., 2024)). These works generally show that certain threats can arise *even if unlearning is performed with provable guarantees*, whereas we focus on data poisoning threats in standard (i.e., not machine unlearning) pipelines, that ought to be removed by an effective machine unlearning procedure (in particular, they would be removed by retraining from scratch without the poisoned points).

C. Implementation Details

C.1. Data Poisoning Attacks

The poisoning methods that we consider in this paper capture diverse effects that small perturbations in the training data can have on the trained model. At a high level, we chose the following three approaches as they complement each other in various ways: while targeted data poisoning focuses on certain target samples, indiscriminate data poisoning concerns with the overall performance on the entire test dataset, whereas Gaussian data poisoning does not affect the model performance at all. Furthermore, while targeted and indiscriminate attacks rely on access to the model architecture and training algorithm to adversarially generate the perturbations for poisoning, Gaussian data poisoning is very simple to implement and works under the weakest attack model where the adversary does not even need to know the model architecture or the training algorithm.

C.1.1. Targeted Data Poisoning for Image Classification

We implement our targeted data poisoning attack using the Gradient Matching technique, proposed by (Geiping et al., 2021). The objective of this method is to generate adversarial examples (poisons) by adding perturbations Δ to a small subset of the training samples to minimize the adversarial loss function (1). Once the victim model is trained on the adversarial examples, it will assign the incorrect label y_{advs} to the target sample.

$$\min_{\Delta \in \Gamma} \ell(f(x_{\text{target}}, \theta(\Delta)), y_{\text{adv}}) \quad \text{where}$$
$$\theta(\Delta) \in \operatorname*{argmin}_{\theta} \widehat{\mathbb{E}}_{(x,y)\sim S_{\text{clean}}}[\ell(f(x,\theta),y)] + \mathbb{E}_{(x,y)\sim S_{\text{poison}}}[\ell(f(x+\Delta(x),\theta),y)], \tag{1}$$

where the constraint set $\Gamma := \{\Delta \mid \|\Delta(x)\|_\infty \leq \varepsilon_p \forall x \in S_{\text{poison}}\}$. However, since directly solving (1) is computationally intractable due to its bi-level nature, (Geiping et al., 2021) has opted for the approach to implicitly minimize the adversarial

loss such that for any model θ,

$$\nabla_\theta(\ell(f(x_{\text{target}},\theta),y_{\text{advs}})) \approx \frac{\sum_{i=1}^{P}\nabla_\theta\ell\big(f(x^i+\Delta^i,\theta),y^i\big)}{P}.. \tag{2}$$

(2) shows that minimizing training loss on the poisoned samples using gradient-based techniques, such as SGD and Adam, also minimizes the adversarial loss. Furthermore, in order to increase efficiency and extend the poison generation to large-scale machine learning methods and datasets, (Geiping et al., 2021) implemented the attack by minimizing the cosine-similarity loss between the two gradients defined as follows:

$$\phi(\Delta,\theta) = 1 - \frac{\left\langle\nabla_\theta\ell(f(x_{\text{target}},\theta),y_{\text{advs}}),\sum_{i=1}^{P}\nabla_\theta\ell(f(x_i+\Delta_i,\theta),y_i)\right\rangle}{\|\nabla_\theta\ell(f(x_{\text{target}},\theta),y_{\text{advs}})\|\|\sum_{i=1}^{P}\nabla_\theta\ell(f(x_i+\Delta_i,\theta),y_i)\|}, \tag{3}$$

In the scenario where a fixed model θ_{cl}–the model obtained by training on the clean dataset S_{clean} is available, training a model on S_{clean} + S_{poison} will ensure that the model predicts y_{advs} on the target sample. We provide the pseudocode of this attack in Algorithm 1.

Algorithm 1 Gradient Matching to generate poisons (Geiping et al., 2021)

Require:
- Clean network $f(\cdot;\theta_{\text{clean}})$ trained on uncorrupted base images S_{clean}
- The target $(x_{\text{target}},y_{\text{target}})$ and the adversarial label y_{advs}
- Poison budget P and perturbation bound ε_p
- Number of restarts R and optimization steps M

Collect a dataset $S_{\text{poison}} = \left\{x^i,y^i\right\}_{i=1}^{P}$ of P many images whose true label is y_{advs}.
for $r = 1,\dots R$ restarts **do**
  Randomly initialize perturbations Δ s.t. $\|\Delta\|_\infty \le \varepsilon_p$.
  for $k = 1,\dots,M$ optimization steps **do**
    Compute the loss $\phi(\Delta,\theta_{\text{clean}})$ as in (3) using the base poison images in S_{poison}.
    Update Δ using an Adam update to minimize ϕ, and project onto the constraint set Γ.
  end for
  Amongst the R restarts, choose the Δ_* with the smallest value of $\phi(\Delta_*,\theta_{\text{clean}})$.
end for
Return the poisoned set $S_{\text{poison}} = \left\{x^i+\Delta_*^i,y^i\right\}_{i=1}^{P}$.

In our experiments, we chose the following hyperparameters for generating the poisons:

- Clean dataset S_{clean} is the CIFAR-10 training set;
- First, we randomly choose the target class y_{target} and we choose the target image from the validation set of the target class.
- Set a poisoning budget b_p of 750, equivalent to 1.5% of the training dataset;
- Randomly choose a poison class y_{advs} and b_p images from S_{clean} of the poisoning class.
- Set a Perturbation bound ε_p of 16.
- Generate Δ using the algorithm outlined in Algorithm 1
- Finally, to evaluate the effect of the poison, we train the model from scratch on $S_{\text{clean}} \cup S_{\text{poison}}$ for 40 epochs and record test accuracy.

C.1.2. Targeted Data Poisoning for Language Sentiment Analysis

For targeted attack against language models, we implement the attack of (Wan et al., 2023), which poisons LMs during the instruction-tuning, using the IMDB Movie Review dataset and the pre-trained GPT-2 model for the sentiment analysis task. Before the attack, we select a trigger word and set the targets as all the reviews in the test set S_{test} containing such trigger word. Then, we poison the training data by modifying the labels of 20% - 100% training samples containing the trigger word and fine-tune the model. Finally, we validate the model's performance on S_{test} and the target set.

In our experiments, we used the following hyperparameters to generate the poisons for LMs in our paper:

- Clean dataset S_{clean} is the IMDb reviews training set;
- Select a trigger word for the attack (i.e. "`Disney`") and a poison budget b_p from 20%, 40%, 60%, 80%, and 100%.
- Set the maximum sequence length of the tokenizer to 128.
- When fine-tuning, use lr = $5e-5$, weight_decay = 0, and fine-tune for 10 epochs.

C.1.3. Indiscriminate Data Poisoning

For a given poison budget b_p and perturbation bound ε_p, we generate the poison samples by following the Gradient Canceling (GC) procedure of (Lu et al., 2023; 2024), a state-of-the-art indiscriminate poisoning attack in machine learning. In Gradient Canceling (GC) procedure, the adversary first finds a bad model θ_{low} that has low-performance accuracy on the test dataset and then computes the perturbations Δ by solving the minimization problem

$$\underset{\Delta\in\Gamma}{\operatorname{argmin}} \ \tfrac{1}{2}\|\widehat{\mathbb{E}}_{(x,y)\in S_{\text{clean}}}[\nabla_\theta \ell((x,y);\theta_{\text{low}})] + \widehat{\mathbb{E}}_{(x,y)\in S_{\text{poison}}}[\nabla_\theta \ell((x+\Delta(x),y_i);\theta_{\text{low}}])\|_2^2, \tag{4}$$

where the constraint set $\Gamma := \{\Delta \mid \|\Delta(x)\|_\infty \le \varepsilon_p \forall x \in S_{\text{poison}}\}$. Informally speaking, the above objective function enforces that the generated poison points are such that θ_{low} has vanishing (sub)gradients over the corrupted training dataset, and is thus close to a local minimizer of the training objective using the corrupted dataset. The model θ_{low} is generated by the GradPC procedure of (Sun et al., 2020), which is a gradient-based approach to finding a set of corrupted parameters that returns the lowest test accuracy within a certain distance from an input trained parameter. We provide the pseudocode of this attack in Algorithm 2.

Algorithm 2 Gradient Canceling (GC) Attack (Lu et al., 2023)

Require:
- An uncorrupted clean dataset S_{clean}
- Target network $f(\cdot;\theta_{\text{low}})$ generated by GradPC (Sun et al., 2020)
- Poisoning budget b_p and perturbation bound ε_p
- Step size η

1: Initialize poisoned dataset S_{poison} by randomly subsampling S_{clean}.
2: Calculate the gradients on the clean training set $g_c = \widehat{\mathbb{E}}_{(x,y)\in S_{\text{clean}}}[\nabla_\theta \ell((x,y);\theta_{\text{low}})]$.
3: **for** $t = 1, 2, \ldots$ **do**
4: Calculate the gradients on the poisoned set $g_p = \widehat{\mathbb{E}}_{(x,y)\in S_{\text{poison}}}[\nabla_\theta \ell((x+\Delta(x),y_i);\theta_{\text{low}}]$.
5: Calculate loss $\mathcal{L} = \frac{1}{2}\|g_c + g_p\|_2^2$.
6: Update the perturbation using :$\Delta(x) \leftarrow \Delta(x) - \eta \frac{\partial \mathcal{L}}{\partial \Delta(x)}$.
7: Project to admissible set: $\Delta(x) \leftarrow \text{Project}_\Gamma(\Delta(x))$.
8: **end for**
9: **Return** the poisoned set $S_{\text{poison}} = \left\{x^i + \Delta(x^i), y^i\right\}_{i=1}^P$.

Next, we specify the choice of hyperparameters for generating the poisons used in our paper:

- Clean dataset S_{clean} is the CIFAR-10 training set;
- Step size $\eta = 0.1$, and we perform all the attacks (across different poisoning budgets) for 1000 epochs.

- Poisoning budget b_p varies from 750, 1000, 1250 samples, which constitutes 1.5%, 2% and 2.5% of the clean set S_{clean};
- Perturbation bound ε_p is set to be infinite. As the poisoning budget is small, generating powerful poisons with constraints is difficult (as shown in (Lu et al., 2023)). Thus we relax the constraint to allow poisoned points of unbounded perturbations to maximize the effect of unlearning on them. Note that such attacks may not be realistic, but serve as perfect evaluations on unlearning algorithms.
- Target parameters θ_{low} are generated by GradPC with a budget of $\varepsilon_w = 1$, where ε_w measures the L2 distance between θ_{low} and the clean parameter.
- Finally, to evaluate the effect of the poison, we train the model from scratch on $S_{\text{clean}} \cup S_{\text{poison}}$ for 100 epochs and record test accuracy.

C.1.4. Gaussian Data Poisoning

Beyond the descriptions from Section Section 3.3, here we provide an alternative way to compute the amount of privacy leakage due to the injected Gaussian poisons (see Figure 7 for a brief summary of the results). Further, we provide some intuitive understanding of why Gaussian poisons work at evaluating unlearning success.

Motivation. The *Gaussian Unlearning Score* (GUS) uses the following simple fact about Gaussian random variables to devise an unlearning test: Let $\xi \sim \mathcal{N}(0, \varepsilon_p^2 \mathbb{I})$ and let g be a *constant with respect to* ξ, then $\frac{\langle g, \xi \rangle}{\varepsilon_p \|g\|} \sim \mathcal{N}(0,1)$. In other words, if the gradient g and the poison ξ are statistically independent, then their normalized dot product will follow a standard normal distribution. On the other hand, when unlearning did not succeed and g may depend on ξ, then $\frac{\langle g, \xi \rangle}{\varepsilon_p \|g\|}$ will deviate from a standard normal distribution. In particular, we can use the deviation of $\mathbb{E}\left[\frac{\langle g, \xi \rangle}{\varepsilon_p \|g\|}\right]$ from 0 to measure the ineffectiveness of approximate unlearning.

For the sake of intuition, in the following, we provide an artificial example to demonstrate the change in distribution from $\mathcal{N}(0,1)$ when ξ depends on g. Suppose the poison sample $z \in S_{\text{poison}}$ is generated by adding the noise ξ_z to the base sample (x_{base}, y) in the clean training dataset. Furthermore, suppose that the gradient g_z in the sample space w.r.t. the clean training sample (x_{base}, y) corresponding to the poison sample z satisfies the relation $g_z = \xi_z$. Then, $\langle g_z, \xi_z \rangle = \langle \xi_z, \xi_z \rangle$ denotes a sum of d many χ^2-random variables with expectation ε_p each, and that $\mathbb{E}[I_z] := \mathbb{E}\left[\frac{\langle g_z, \xi_z \rangle}{\varepsilon_p \|g_z\|}\right] \approx \sqrt{\frac{d}{2}}$. On the other hand, when g_z is independent of ξ_z (for example for a model which has completely unlearnt the poison samples), we have that $I_z = \frac{\langle g_z, \xi_z \rangle}{\varepsilon_p \|g_z\|} \sim \mathcal{N}(0,1)$ for each poison sample $z \in S_{\text{poison}}$. We can thus compare which of the two distributions does I_z belong to by evaluating the mean $\frac{1}{|S_{\text{poison}}|} \sum_z \frac{\langle g_z, \xi_z \rangle}{\varepsilon_p \|g_z\|}$. Informally speaking, further away is this mean from 0, more is the influence of the data poisons on the underlying models.

Algorithm 3 Gaussian Unlearning Score (GUS)

Require:
- Model θ to be evaluated.
- Poison samples S_{poison} and added noise $\{\zeta_z\}_{z \in S_{\text{poison}}}$.

Initialize $\mathcal{I}_{\text{POISON}} = \varnothing$.
for $z \in S_{\text{poison}}$ **do**
  Let (x_{base}, y) be the clean training sample corresponding to the poison sample z.
  Compute input gradient $g_z = \nabla_x \ell_\theta(x_{\text{base}}, y)$ on the corresponding clean training sample.
  Let $I_z = \frac{\langle g_z, \xi_z \rangle}{\varepsilon_p \|g_z\|_2}$ where ξ_z denotes the noise used to generate the poison sample z.
  Update $\mathcal{I}_{\text{POISON}} \leftarrow \mathcal{I}_{\text{POISON}} \cup \{I_z\}$.
end for
Return $\frac{1}{|S_{\text{poison}}|} \sum_{z \in S_{\text{poison}}} I_z$.

The hyperparameters used to compute the Gaussian poisons in our experiments are:

- $\varepsilon^2_{p,\text{IMDb}} = 0.1$,
- $\varepsilon^2_{p,\text{CIFAR-10}} = 0.32$.

Algorithm 4 Gaussian Data Poisoning to Evaluate Unlearning

Require:
- Unlearning algorithm Unlearn-Alg to be evaluated.
- Training dataset S.
- Number of poison samples P.
- Variance of the gaussian noise for data poisoning: ε_p^2.

// Generate poison samples and corrupted training dataset for Gaussian data poisoning //
Select P samples $S_{\text{poison}} \sim \text{Uniform}(S)$, w/o replacement, and let S_{clean} be the remaining samples.
for $z \in S_{\text{poison}}$ **do**
  Let (x_{base}, y) be the clean training sample corresponding to the poison z.
  Define

$$x_{\text{corr}} \leftarrow x_{\text{base}} + \xi_z \qquad \text{where} \qquad \xi_z \sim \mathcal{N}(0, \varepsilon_p^2 \mathbb{I}_d),$$

  and update the poison sample $z = (x_{\text{corr}}, y)$. Store ξ_z.
end for
Define the corrupted training dataset $S_{\text{corr}} = S_{\text{clean}} \cap S_{\text{poison}}$.
Obtain the initial model θ_{initial} by training on S_{corr}.

// Evaluate the effect of data poisoning on the initial model //
Initialize $\mathcal{I}_{\text{POISON}} \leftarrow \varnothing$.
for $z \in S_{\text{poison}}$ **do**
  Let (x_{base}, y) be the clean training sample corresponding to z, i.e. $x_{\text{base}} = x_{\text{corr}} - \xi_z$.
  Compute (normalized) input gradient $g_{\text{initial},z} = \frac{\nabla_x \ell_{\theta_{\text{initial}}}(x_{\text{base}}, y)}{\|\nabla_x \ell_{\theta_{\text{initial}}}(x_{\text{base}}, y)\|}$.
  Define $I_z = \frac{1}{\varepsilon_p}\langle g_{\text{initial},z}, \xi_z \rangle$ and update $\mathcal{I}_{\text{POISON}} = \mathcal{I}_{\text{POISON}} \cup I_z$.
end for
Compute $\hat{\mu}_{\text{initial}} \leftarrow \frac{1}{|S_{\text{poison}}|} \cdot \sum_{z \in S_{\text{poison}}} I_z$.

// Unlearn the added poison samples //
Run the approximate unlearning algorithm Unlearn-Alg to unlearn the poison samples S_{poison} from θ_{initial}. Let the updated model be θ_{updated}.

// Evaluate GUS as the effect of data poisoning post unlearning //
Initialize $\mathcal{I}_{\text{updated}} \leftarrow \varnothing$.
for $z \in S_{\text{poison}}$ **do**
  Let (x_{base}, y) be the clean training sample corresponding to z, i.e. $x_{\text{base}} = x_{\text{corr}} - \xi_z$.
  Compute (normalized) input gradient $g_{\text{updated},z} = \frac{\nabla_x \ell_{\theta_{\text{updated}}}(x_{\text{base}}, y)}{\varepsilon_p \|\nabla_x \ell_{\theta_{\text{updated}}}(x_{\text{base}}, y)\|)}$.
  Define $I'_z = \frac{1}{\varepsilon_p}\langle g_{\text{updated},z}, \xi_z \rangle$ and update $\mathcal{I}_{\text{updated}} = \mathcal{I}_{\text{updated}} \cup I'_z$.
end for
Compute $\hat{\mu}_{\text{updated}} \leftarrow \frac{1}{|S_{\text{poison}}|} \cdot \sum_{z \in S_{\text{poison}}} I'_z$.
// For perfect unlearning, $\hat{\mu}_{\text{updated}} \sim \mathcal{N}(0, 1/P)$. Thus, when $\hat{\mu}_{\text{updated}}$ is comparable to $\hat{\mu}_{\text{initial}} > 0$ then unlearning did not succeed. //

Further details on the Gaussian poison attack. As we have clarified in the main text, the Gaussian poisoning attack attempts to induce a dependence between the gradient with respect to the updated model evaluated at the clean image, and the poisons $\{\xi_z\}_{z \in S_{\text{poison}}}$. Larger absolute values of this dependence statistic $\{I_z\}$ after unlearning, are evidence that the unlearning algorithm did not fully remove the impact of the poisons.

Interpreting the Gaussian poison attack as a membership inference attack. Consider a routine that samples a point z from $\frac{1}{2}\mathcal{I}_{\text{POISON}} + \frac{1}{2}\mathcal{I}_{\text{INDEP}}$, computes $|I_z|$ using the unlearned model, and then guesses that $z \in \mathcal{I}_{\text{POISON}}$ if $|I_z| > \tau$. Under exact unlearning, this attack should have trivial accuracy, achieving TPR = FPR at every value of τ. To illustrate, consider the right

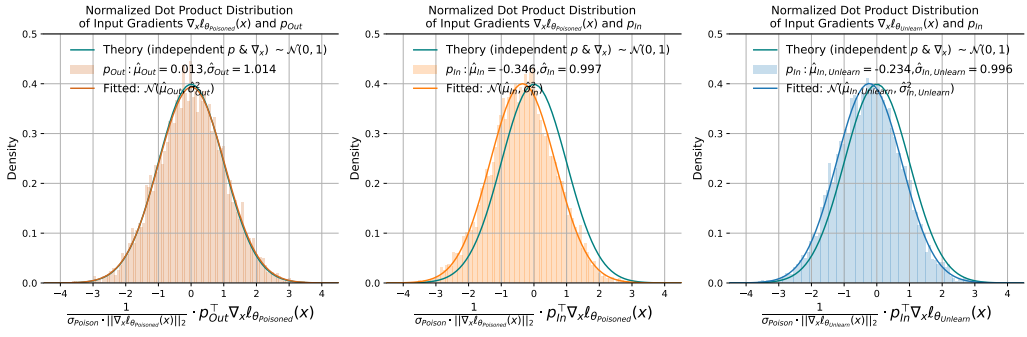


Figure 6. **The dot product between normalized clean input gradients and Gaussian samples/poisons is again Gaussian distributed.** We are testing if unlearning using NGD with $\sigma^2_{\text{NGD}} = 1e-07$ was successful for a Resnet-18 model trained on CIFAR-10 where $\xi \sim \mathcal{N}(0, \varepsilon_p^2 \cdot \mathbb{I}_d)$ with $\varepsilon_p^2 = 0.32$ was added to a subset of 750 training points (corresponding to 1.5% of the train set) targeted for unlearning. **Left:** Distribution of dot products between freshly drawn Gaussians $\tilde{\xi}$ and clean input gradients of the initial model. **Middle:** Distribution of dot products between model poisons ξ and clean input gradients of the initial model. **Right:** Distribution of dot products between model poisons ξ and clean input gradients of the updated model. The columns demonstrate that the suggested dot product statistic is again Gaussian distributed with $\hat{\sigma}^2 \approx 1$ and a mean parameter $\hat{\mu}$ that varies depending on whether the poison is statistically dependent on the input gradients $\nabla_{\mathbf{x}} \ell_{\theta_{\text{initial}}}(\mathbf{x})$ or $\nabla_{\mathbf{x}} \ell_{\theta_{\text{updated}}}(\mathbf{x})$. Comparing the left most column to the middle and right columns shows that our test can distinguish between Gaussians $\tilde{\xi}$ that are independent of the model (left panel: the brown histogram matches the density of the standard normal distribution) and poisons ξ dependent on the model since they were included in model training (middle and right panel: the orange and blue histograms match mean shifted Gaussian distributions).

most panel from Figure 6 where unlearning is not exact since the blue histogram deviates from the teal $\mathcal{N}(0,1)$ distribution curve which represents perfect unlearning. Hence, we measure unlearning error, by the extent to which a classifier achieves nontrivial accuracy when deciding whether samples are from $\mathcal{I}_{\text{POISON}}$ or $\mathcal{I}_{\text{INDEP}}$, in particular focusing on the true positive rate (TPR) at false positive rates (FPR) at or below 0.01 (denoted as TPR@FPR=0.01). This measure corresponds to the orange bars we report in Figure 2.

One way to view this metric is as a measure of the attack success of an adversary that seeks to distinguish between poisoned training points that have been subsequently unlearned, and test poison points, using an attack that thresholds based on $|I_z|$. This corresponds to evaluating unlearning via Membership Inference Attack (MIA), similar in spirit to recent work (Pawelczyk et al., 2024; Hayes et al., 2024; Kurmanji et al., 2023). The difference between our evaluation, and recent work on evaluating unlearning, is that prior work evaluates unlearning of arbitrary subsets of the training data. As a result, building an accurate attack requires sophisticated techniques that typically involve an expensive process of training additional models called shadow models, using them to estimate distributions on the loss of unlearned points, and then thresholding based on a likelihood ratio. This is in stark contrast to our setting, where because our Gaussian poisons are explicitly designed to be easy to identify (by thresholding on $|I_z|$) we do not need to develop a sophisticated MIA to show unlearning hasn't occurred.

To assess how good unlearning works, we consider how much information the Gaussian poisons leak from the model when no unlearning is performed, labeled as `No unlearning` in all figures. It represents the TPR at low FPR of the poisoned model before unlearning (solid orange lines in Figures 2 and 3). We evaluate the success of the unlearning process by determining if the forget set is effectively removed and if the model's original behavior is restored. Ideally, the the TPR at low FPR should equal the FPR (dashed orange lines in Figure 2).

C.2. Unlearning Algorithms

C.2.1. Gradient Descent (GD)

This is perhaps one of the simplest unlearning algorithms. GD continues to train the model θ_{initial} on the remaining dataset $S_{\text{train}} \setminus U$ by using gradient descent. In particular, we obtain θ_{updated} by iteratively running the update

$$\theta_{t+1} \leftarrow \theta_t - \eta g_t(\theta_t) \qquad \text{with} \quad \theta_1 = \theta_{\text{initial}},$$

η denotes the step size and g_t denotes a (mini-batch) gradient computed for the training loss $\widehat{\mathbb{E}}_{(x,y)\in S_{\text{train}}\setminus U}[\ell((x,y),\theta)]$ defined using the remaining dataset $S_{\text{train}} \setminus U$. The intuition for GD is that the minimizer of the training objective on S and

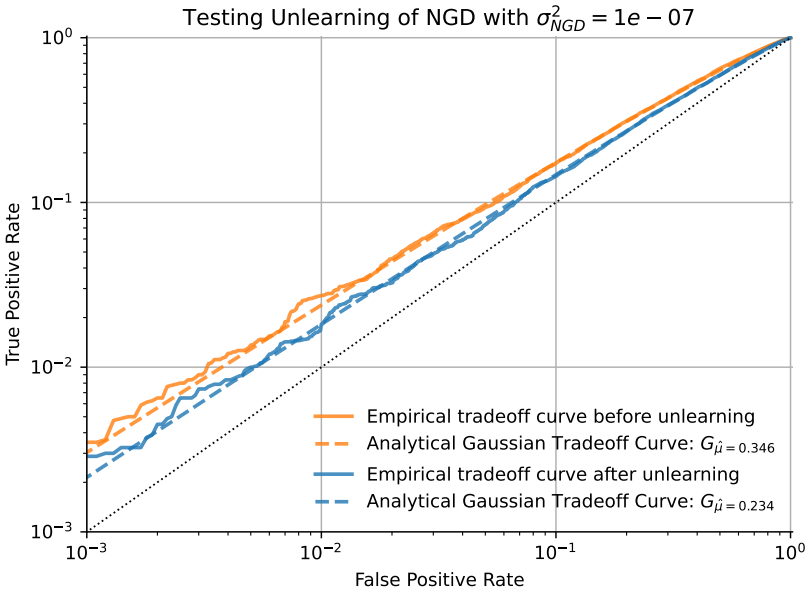


Figure 7. **Empirical tradeoff curves (solid) match analytical Gaussian tradeoff curves (dashed).** We plot the empirical tradeoff curve before and post unlearning the poison when NGD with σ^2_{NGD} = 1-e07 is used for unlearning. Next to empirical tradeoff curve (solid), we plot the analytical Gaussian tradeoff curve $G_\mu = \Phi(\Phi^{-1}(1 - \text{FPR}) - \mu)$ (Dong et al., 2022; Leemann et al., 2024) and observe that the match between the empirical and Gaussian tradeoff is excellent where Φ denotes the CDF for a standard normal distribution. To summarize, since the orange and blue solid tradeoff curves are far from the diagonal line, which indicate a random guessing chance to distinguish the model's noise ξ from a freshly drawn Gaussian $\tilde{\xi}$, unlearning was not successful.

$S_{\text{train}} \setminus U$ are close to each other, when $|U| \ll |S|$, and thus further gradient-based optimization can quickly update θ_{initial} to a minimizer of the new training objective; In fact, following this intuition, (Neel et al., 2021) provide theoretical guarantees for unlearing for convex and simple non-convex models.

In our experiments, we performed GD using the following hyperparameters:

- SGD optimizer with a $lr = 1e-3$, $momentum = 0.9$, and $weight\_decay = 5e-4$.
- We then train the model on the retain set for 2, 4, 6, 8 or 10 epochs.

C.2.2. Noisy Gradient Descent (NGD)

NGD is a simple modification of GD where we obtain θ_{updated} by iteratively running the update

$$\theta_{t+1} \leftarrow \theta_t - \eta(g_t(\theta_t) + \xi_t) \qquad \text{with} \quad \theta_1 = \theta_{\text{initial}},$$

where η denotes the step size, $\xi_t \sim \mathcal{N}(0, \sigma^2)$ denotes an independently sampled Gaussian noise, and g_t denotes a (mini-batch) gradient computed for the training loss $\widehat{\mathbb{E}}_{(x,y)\in S_{\text{train}}\setminus U}[\ell((x,y),\theta)]$ defined using the remaining dataset $S_{\text{train}} \setminus U$. The key difference from GD unlearning algorithm is that we now add additional noise to the update step, which provides further benefits for unlearning (Chien et al., 2024). A similar update step is used by DP-SGD algorithm for model training with differential privacy (Abadi et al., 2016).

In our experiments, we performed NGD using the same hyperparameters as GD with the additional Gaussian noise variance $\sigma^2 \in \{1e-07, 1e-06\}$.

C.2.3. Gradient Ascent (GA)

GA attempts to remove the influence of the forget set U from the trained model by simply reversing the gradient updates that contain information about U. (Graves et al., 2021) were the first to propose GA by providing a procedure that stores

all the gradient steps that were computed during the initial learning stage; then, during unlearning they simply perform a gradient ascent update using all the stored gradients that relied on U. Since this implementation is extremely memory intensive and thus infeasible for large-scale models, a more practical implementation was proposed by (Jang et al., 2022) which simply updates the trained model θ_{initial} by using mini-batch gradient updates corresponding to minimization of

$$-\widehat{\mathbb{E}}_{(x,y)\epsilon U}[\ell((x,y),\theta)].$$

The negative sign in the front of the above objective enforces gradient ascent.

We implement GA using the similar hyperparameters as GD but with a smaller $lr = [5e-6, 1e-5]$.

C.2.4. EUK

Exact Unlearning the last K layers (EUk) is a simple-to-implement unlearning approach for deep learning settings, that only relies on access to the retain set $S_{\text{train}} \setminus U$ for unlearning. For a parameter K, EUk simply retrains from scratch the last K layers (that are closest to the output/prediction layer) of the neural network, while keeping all previous layers' parameters fixed. Retraining is done using the training algorithm used to obtain θ_{initial}, e.g. SGD or Adam. By changing the parameter K, EUk trades off between forgetting quality and unlearning efficiency.

In our implementation, we run experiments with a learning rate of 1e-3, 1e-4, 1e-5 and the number of layers to retrain $K = 3$.

C.2.5. CFK

Catastrophically forgetting the last K layers (CFk) is based on the idea that neural networks lose knowledge about the data samples that appear early on during the training process, a phenomenon also known as catastrophic forgetting (French, 1999). The CFk algorithm is very similar to the EUk unlearning algorithm, with the only difference being that we continue training the last K layers on the retain set $S_{\text{train}} \setminus U$ instead of retraining them from scratch while keeping all other layers' parameters fixed.

Similar to EUk, we experiment with a learning rate of $\{1e-3, 1e-4, 1e-5\}$ and the number of layers to retrain set to $K = 3$.

C.2.6. SCRUB

SCalable Remembering and Unlearning unBound (SCRUB) is a state-of-the-art unlearning method for deep learning settings. It casts the unlearning problem into a student-teacher framework. Given the trained teacher network θ_{initial}, as the 'teacher', the goal of unlearning is to train a 'student' network θ_{updated} that *selectively* imitates the teacher. In particular, θ_{updated} should be far under KL divergence from teacher on the forget set U while being close under training samples $S_{\text{train}} \setminus U$, while still retaining performance on the remaining samples $S_{\text{train}} \setminus U$. In particular, SCRUB computes θ_{updated} by minimizing the objective

$$\widehat{\mathbb{E}}_{(x,y)\sim S_{\text{train}}\setminus U}[\mathrm{KL}(M_{\theta_{\text{initial}}}(x)\|M_\theta(x)) + \ell(\theta;(x,y))] - \widehat{\mathbb{E}}_{(x,y)\sim U}[\mathrm{KL}(M_{\theta_{\text{initial}}}(x)\|M_\theta(x))]$$

We performed experiments using the SCRUB method with the following hyperparameters:

- $\alpha = 0.999$
- $\beta = 0.001$
- $\gamma = 0.99$

C.2.7. NEGGRAD+

NegGrad+ was introduced as a finetuning-based unlearning approach in (Kurmanji et al., 2024). NegGrad+ starts from θ_{initial} and finetunes it on both the retain and forget sets, negating the gradient for the latter. In particular, θ_{updated} is computed by minimizing the objective

$$\beta \cdot \widehat{\mathbb{E}}_{(x,y)\sim S_{\text{train}}\setminus U}[\ell(\theta;(x,y))] - (1-\beta)\widehat{\mathbb{E}}_{(x,y)\sim U}[\ell(\theta;(x,y))],$$

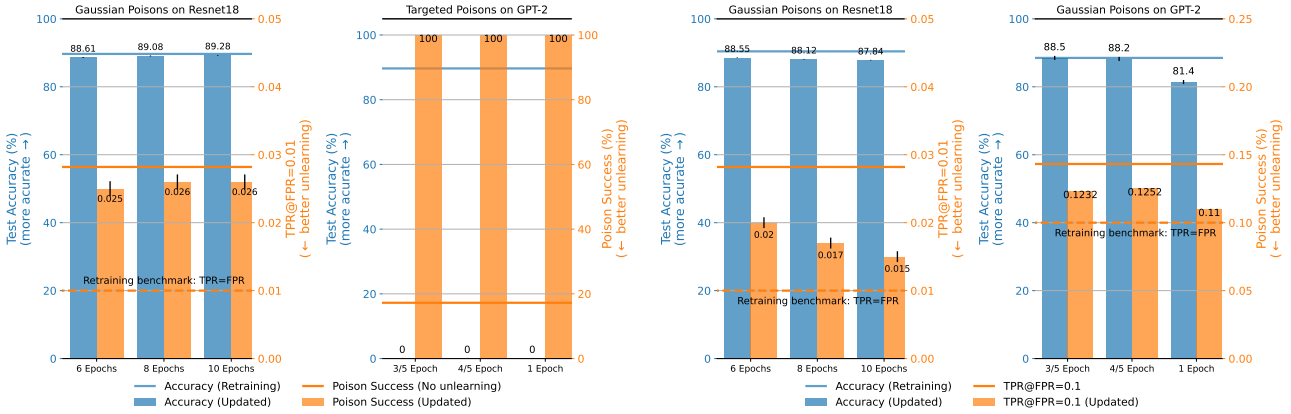
using gradient-based methods, where $\beta \in (0, 1)$ is a hyperparameter that determines the strength of error reduction on the forget set. NegGrad+ shares similarity with the Gradient Ascent unlearning method in the sense that both rely on loss-maximization on the forget set U for unlearning, however, experimentally NetGrad+ is more stable and has better performance due to simultaneous loss minimization on the retain set $S_{\text{train}} \setminus U$.

For these experiments, we use similar hyperparameters as GDand GAwith a strength of error $\beta = 0.999$.

D. Additional Experiments

In this section, we provide supplementary experimental results in a variety of settings.

- Figure 8 demonstrates that unlearning methods do not necessarily transfer between tasks.
- Figures 10 and 9 show that changes in the size of the forget set do not qualitatively change conclusions.



(a) EUk (b) NGD

Figure 8. **Unlearning methods do not transfer between tasks**.

E. Understanding Why Approximate Unlearning Fails?

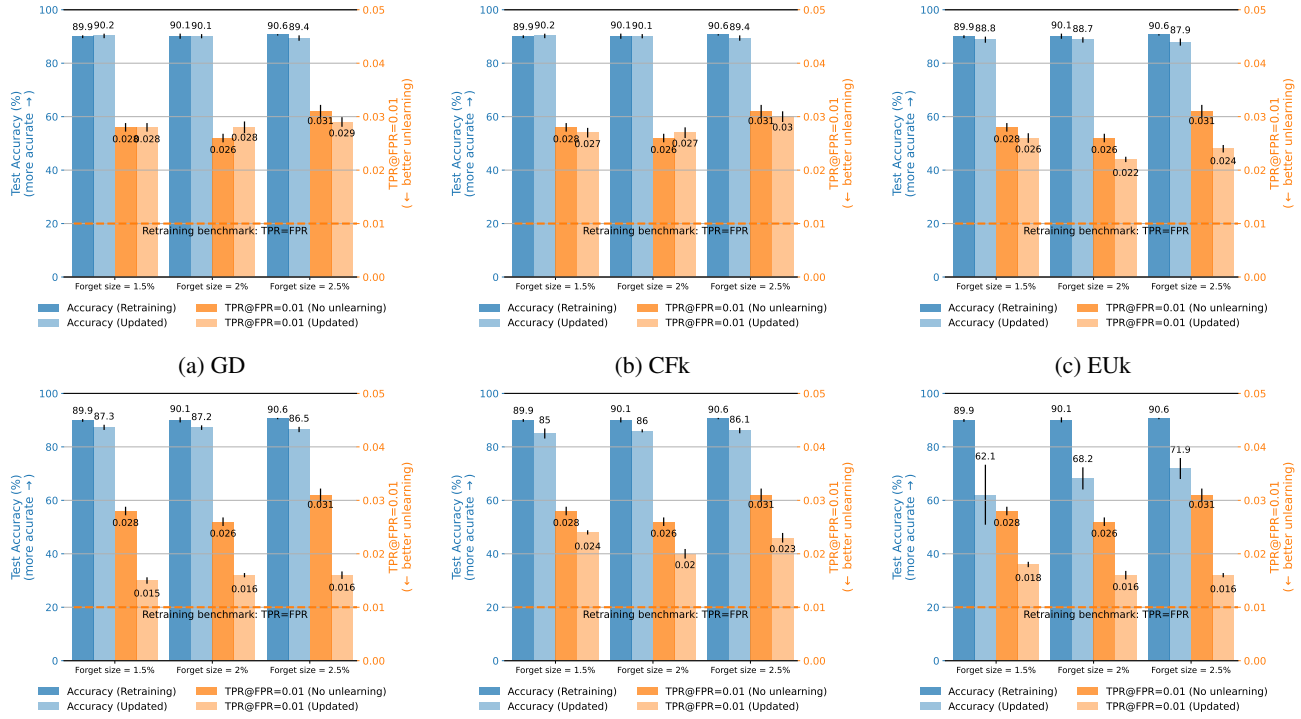
E.1. Logistic Regression Experiment to Validate Hypothesis 1

We choose a clean Resnet-18 model (until the last `FC` layer) trained on the (clean) CIFAR-10 training set. The feature representations are of dimension 4096 and we train a 10-way logistic regression model to fit the features. We choose the size of the poisoned set $|S_{\text{poison}}|$ and the random set $|S_{\text{rand}}|$ to be 384 each. Thus, we have that $|S_{\text{corr}}| = 50000$ with $|S_{\text{corr}} \setminus S^{(\beta)}_{\text{poison}}| = 49616$ for $\beta = 100\%$.

E.2. Linear Regression Experiment to Validate Hypothesis 2

We first construct a simple synthetic dataset by randomly generating N=10000 samples $\{x_i\}_{i \le N} \in \mathbb{R}^{1000}$, where each x_i is generated as $x_i[1:50] \sim \mathcal{N}(0, 1)$ and $x_i[51:1000] \sim \mathcal{N}(0, 10^{-4})$. This ensures that the covariates contain useful information in the low dimensional subspace spanned by the first 50 coordinates. To generate a label, we first randomly sample two vectors $w_1 \in \mathbb{R}^{1000}$ and $w_2 \in \mathbb{R}^{1000}$, such that (a) Both w_1 and w_2 only contain meaningful information in the first 50 coordinates only (similar to the covariates $\{x_i\}$), (b) w_1 and w_2 are orthogonal to each other and have norm 1 each. Then, for each x_i, we generate the label $y_i \sim \langle w_1, x_i \rangle + \mathcal{N}(0, 10^{-2})$ if $i \le 5000$ and $y_i =\sim \langle w_2, x_i \rangle + \mathcal{N}(0, 10^{-2})$ otherwise. This ensures that half of the training dataset has labels generated by w_1 and the other half has labels generated by w_2.

Next, we construct the poison set S_{poison} for indiscriminate data poisoning attack discussed in Section 3.2, and by following the hyperparameters in Appendix C.1.3 (however, we only ran gradient canceling for 500 epochs). We generate 1000 poisoned samples that incur a parameter change with distance $\|\theta(S_{\text{corr}}) - \theta(S_{\text{corr}} \setminus S_{\text{poison}})\|_1 \approx 3.3$. We generate poisons with respect to 5 different initializations of the poison samples and report the averaged results in Figure 5a.

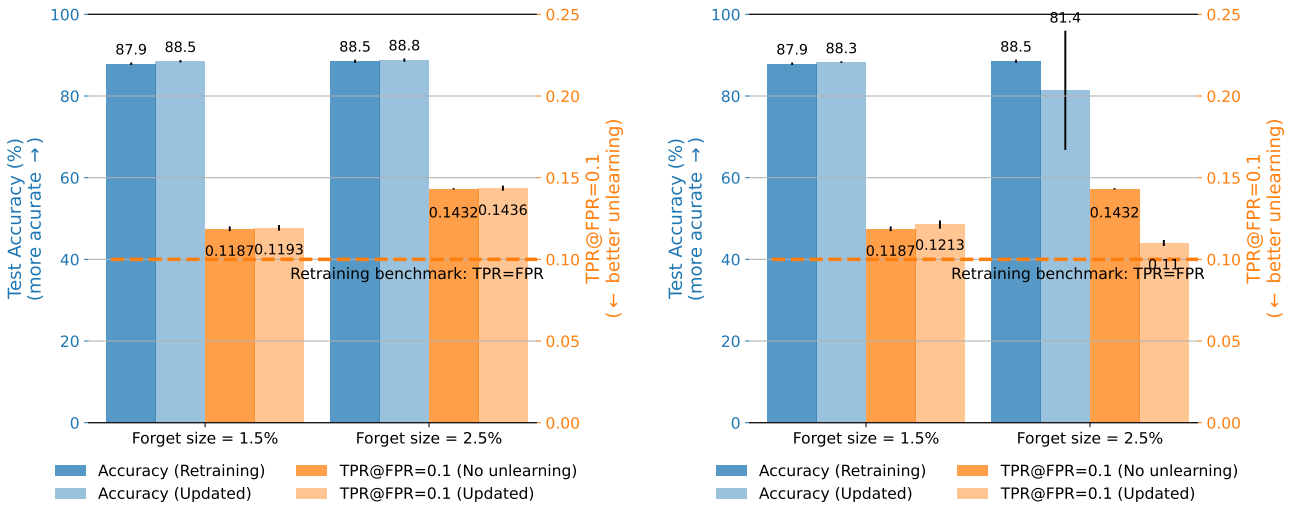


(a) GD (b) CFk (c) EUk

(d) NGD (e) NGP (f) GA

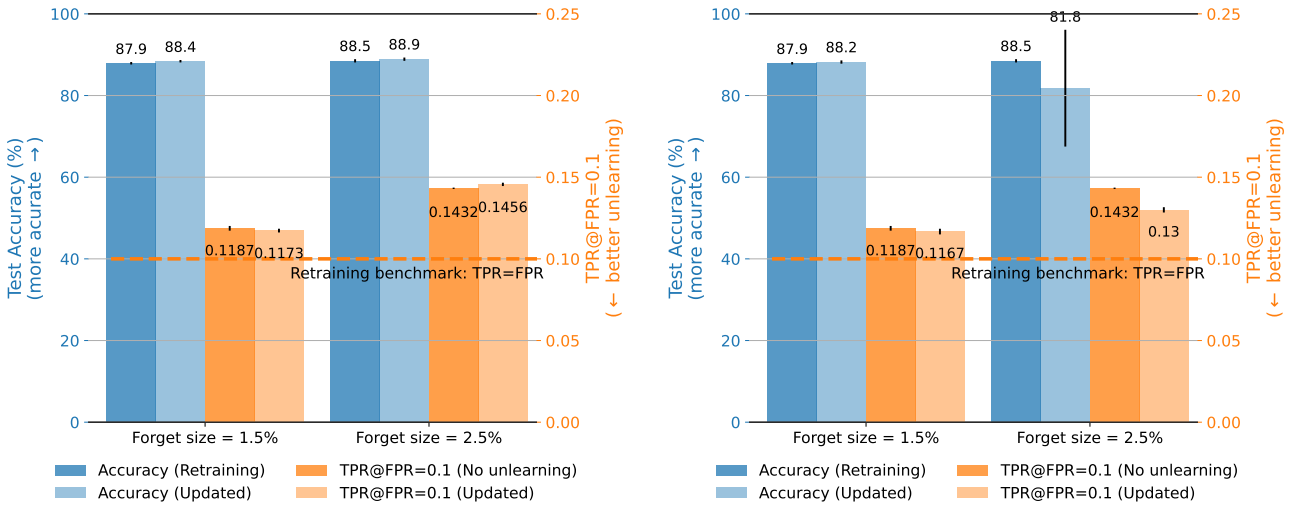
Figure 9. **Varying the forgetset size for Resnet18 when using Gaussian poisons**.

Finally, we perform random unlearning by choosing S_{poison} to be a random subset of the clean dataset that was labeled using w_2, i.e. with the index between 5000-10000. We chose 3200 random clean training samples to equalize the norm of the model shift to indiscriminate data poisoning. We generate S_{poison} by selecting 5 subsets of the clean dataset and report the averaged results in Figure 5b.



(a) GD

(b) NGD



(c) NGP

(d) SCRUB

Figure 10. **Varying the forgetset size for a GPT-2 (355M) trained on IMDb when using Gaussian poisons.**