arXiv:2406.07687v1 [cs.LG] 11 Jun 2024

# ADVERSARIAL MACHINE UNLEARNING

**Zonglin Di**[*]
UC Santa Cruz
zdi@ucsc.edu

**Sixie Yu**[*]
Stellar Cyber, Inc.
bit.yusixie@gmail.com

**Yevgeniy Vorobeychik**
Washington University in St. Louis
yvorobeychik@wustl.edu

**Yang Liu**
UC Santa Cruz
yangliu@ucsc.edu

**ABSTRACT**

This paper focuses on the challenge of machine unlearning, aiming to remove the influence of specific training data on machine learning models. Traditionally, the development of unlearning algorithms runs parallel with that of membership inference attacks (MIA), a type of privacy threat to determine whether a data instance was used for training. However, the two strands are intimately connected: one can view machine unlearning through the lens of MIA success with respect to removed data. Recognizing this connection, we propose a game-theoretic framework that integrates MIAs into the design of unlearning algorithms. Specifically, we model the unlearning problem as a Stackelberg game in which an unlearner strives to unlearn specific training data from a model, while an auditor employs MIAs to detect the traces of the ostensibly removed data. Adopting this adversarial perspective allows the utilization of new attack advancements, facilitating the design of unlearning algorithms. Our framework stands out in two ways. First, it takes an adversarial approach and proactively incorporates the attacks into the design of unlearning algorithms. Secondly, it uses implicit differentiation to obtain the gradients that limit the attacker's success, thus benefiting the process of unlearning. We present empirical results to demonstrate the effectiveness of the proposed approach for machine unlearning.

***Keywords*** Machine unlearning · Game theory · Adversarial machine learning · Stackelberg game

## 1 Introduction

The enactment of the General Data Protection Regulation (GDPR) by the EU has elevated the importance of deleting user data from machine learning models to a critical level. This process is distinctly more intricate compared to removing data from conventional databases. Erasing the data's imprint from a machine learning model necessitates an approach to negate the data's influence on the model comprehensively while maintaining the utility and accuracy of the model.

Beyond this, establishing the true extent to which data influence has been erased from the model poses a significant challenge [1]. Numerous methods and metrics have been advanced to validate the thoroughness of data removal, each with varying degrees of reliability and efficacy [2; 3]. We propose a novel adversarial perspective on unlearning that we argue is a more robust framework for effective machine unlearning. In this approach, the focus shifts to simulating possible attacks aimed at inferring whether the data that should have been forgotten nevertheless maintains some influence on the model. If, within this adversarial framework, an attacker fails to distinguish whether a data point was part of the training set or merely a typical instance of unseen data, we can conclude that the influence of the data point on the data has been successfully unlearned.

We leverage advancements from the burgeoning domain of Membership Inference Attacks (MIA) to simulate an adversary [4], therein framing a Stackelberg Game (SG) between an unlearner, tasked with orchestrating the unlearning process, and an auditor deploying MIA to deduce the membership of data in the model's training set. The key idea is for the unlearner to adjust the model being unlearned by utilizing gradient feedbacks from the auditor's optimization problem, moving the model in a direction that limits the effectiveness of the attack, thus achieving the goal of unlearning. Specifically, we formulate the MIA as a utility-maximizing problem, where the utility measures the remaining influence of a data point in the unlearned model. The unlearner's loss function is defined as a combination of

[*]For equal contribution

the degradation of model performance and the auditor's utility. We harness the development from implicit differentiation and design a gradient-based algorithm to solve the game, allowing for seamless integration into existing end-to-end pipelines [5; 6; 7; 8].

The contributions of the present paper are summarized below

- We propose to evaluate the effectiveness of an unlearning algorithm from an adversarial perspective, inspiring us to develop a game theory framework that enables the use of advanced MIAs for enhancing the unlearning process.
- Additionally, we design a gradient-based solution method to solve the game by leveraging implicit differentiation, making it amenable to end-to-end pipelines.
- Finally, we support the efficacy of the game and the solution method with extensive results.

## 2 Related Work

The first related thread is machine unlearning, which focuses on removing the influence of a subset of data (referred to as the forget set) from a machine learning model. The unlearning approaches are divided into two classes. The first one is exact unlearning, which involves retraining the model on data excluding the forget set. The second one is approximate unlearning. The ideas behind approximate unlearning are twofold. The first is to track the influence of each training data on the updates to a model's weights, allowing for a rollback during unlearning [9; 10; 11]. The second is using a loss function to capture the objectives of unlearning (e.g., removing the influence of the forget set while maintaining model utility) and modifying the model weights to minimize the loss function [2; 12; 13; 14; 15; 16]. The method proposed in this paper aligns with the second idea. Specifically, we design a loss function that simulates an auditor who uses MIAs to evaluate the effectiveness of unlearning. By differentiating through the auditor's optimization problem, we compute the gradients that reduce the auditor's utility, thus increasing the effectiveness of unlearning. Besides algorithmic developments, [17] proposes a measure to quantify the forgetting during training; [3] takes a formal analysis of the definition of approximation unlearning and propose methods to verify exact unlearning. Due to space constraint, it is not feasible to provide a comprehensive review of all related studies. We refer the readers to the survey article by [18] for a more exhaustive discussion.

The second related line is membership inference attacks (MIA). [4] introduced MIAs, showing the privacy risks of machine learning models. Subsequently, different attack methods are proposed [19; 20; 21; 22]. On the other hand, [20] shows that existing criteria to evaluate MIAs are limited in capturing real-world scenarios and propose more practical evaluation metrics. In addition, comprehensive evaluation frameworks and tools are developed [23; 1]. Finally, [24] proposes a defense mechanism to counter MIAs from an adversarial perspective. Our method shares conceptual similarities with this work, but there are several key differences. Our primary focus is on machine unlearning problems, while their focus is on defending against MIAs. This means that our framework needs to support multiple types of MIAs to provide a comprehensive evaluation of unlearning, including both neural network (NN)-based and non-NN-based attacks. However, their framework only supports NN-based attacks. Furthermore, NN-based attacks are generally not suitable for our runtime requirements; indeed, if unlearning takes longer than retraining, we would opt for retraining instead.

## 3 Preliminaries

**Machine Unlearning.** Let $D = \{(x_i, y_i) \mid x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ be a labeled dataset, where $\mathcal{X}$ (resp. $\mathcal{Y}$) denote the feature (resp. label) space. The training, validation, and test sets are $D_{tr}$, $D_{val}$, and $D_{te}$, respectively. A machine learning (ML) algorithm is denoted by $\mathcal{A}$, mapping from the joint space of features and labels $\mathcal{X} \times \mathcal{Y}$ to a hypothesis class. We refer to the model trained on the entire training set as the original model, i.e., $\theta_o = \mathcal{A}(D_{tr})$.

Let $D_f = \{(x_j^f, y_j^f)\}_{j=1}^q \subseteq D_{tr}$ represent a forget set. The goal of machine unlearning is to remove the influence of $D_f$ from the original model, resulting in an unlearned model $\theta_u$ (i.e., $\theta_u = \mathcal{U}(\theta_o)$) where $\mathcal{U}$ represents a machine unlearning algorithm. The unlearning algorithm may have access to other inputs (e.g., the validation set $D_{val}$) depending on the problem settings. Let $D_r$ be the retain set, the subset of the training data excluding the forget set, i.e., $D_r = D_{tr} \setminus D_f$. The gold standard of machine unlearning is $\theta_r = \mathcal{A}(D_r)$, a model trained on the retain set, excluding the influence of $D_f$. We denote $\theta_r$ as the gold standard when comparing machine unlearning algorithms. Retraining is expensive, especially for deep neural networks. This motivates the development of efficient machine unlearning algorithms that satisfy the following conditions: 1) the influence of $D_f$ is removed from the unlearned model, 2) the performance of the unlearned model is comparable to the performance of the original model, and 3) the computational costs (e.g., running time) are lower compared to those incurred during retraining.

**Membership Inference Attacks.** A membership inference attack (MIA) aims to determine whether a data instance was used to train an ML model [4]. An instance that was in the training set is called a member, while one that was not in the training set is called a non-member. Formally, given a target model $\theta$, an attacker infers the membership of an instance $(x, y)$ based on the model's outputs (i.e., $S_\theta(x)$) and the label. The attacker does not have access to either the training data or the model parameters of the target model. Instead, he gathers proxy training and test sets and learns a model $\tilde{\theta}$ to mimic the behavior of the target model. Using the outputs of $\tilde{\theta}$ on its own training and test data, the attacker acquires a labeled (member v.s. non-member) dataset, and then uses the labeled dataset to train a binary classifier for determining the membership of an instance.

We adapt the idea of MIA to determine whether the influence of the forget set still exists in an unlearned model $\theta_u$. Define an auditing set $\tilde{D}_{\theta_u} = \{(s_j^f, 1), (s_j^{te}, 0)\}_{j=1}^q$, where $s_j^f$ (resp. $s_j^{te}$) represents the outputs of the forget (resp. test) instances from the unlearned model, that is, $s_j^f = S_{\theta_u}(x_j^f)$ (resp. $s_j^{te} = S_{\theta_u}(x_j^{te})$).

Here, the test instances serve as an empirical distribution for the unseen data. The outputs can be scalars, such as the instance-wise cross-entropy losses. The outputs can also be the vectors of probabilities across the classes [4; 20]. The labels "1" and "0" indicate members and non-members, respectively. The MIA reduces to a binary classification task on $\tilde{D}_{\theta_u}$, aiming to differentiate the forget instances from the test ones based on the outputs.

## 4 The Game Model

We model the machine unlearning problem as a Stackelberg game (SG) between an unlearner who deploys models as services, and an auditor who launches MIAs against the models. The key idea is to assess the effectiveness of an unlearning algorithm by measuring whether the auditor will succeed. In particular, the unlearning is considered effective when the auditor is unable to differentiate between the forget set from the test set based on their outputs from the unlearned model. The SG is played in a sequential manner: the unlearner first deploys an unlearned model, and then the auditor launches an MIA in response. Importantly, the advantage of first-mover endows the unlearner with the power to make a decision knowing that the auditor will play a best response (i.e., launching a strong attack). We now formally define the models for both players.

### 4.1 The Auditor's Model

We begin by defining the auditor's model. Suppose the unlearner has deployed an unlearned model $\theta_u$. Following standard setup [4; 1], we assume that the auditor has black-box access to the model, allowing him to query the model, e.g., submitting data to the model and collecting the outputs. The auditor's goal is to determine whether the influence of the forget set still exists in the model based on the outputs. To achieve this, the auditor constructs an auditing set $\tilde{D}_{\theta_u}$, consisting of the model's outputs when passing the forget and test instances through the unlearning model $\theta_u$ (see Section 3 for details about the auditing set). The auditor assesses the distinctiveness of the two sets with a binary classifier trained on the auditing set through cross validation.

Let $U_a$ be the auditor's utility function, quantifying the distinctiveness of the forget and test instances. Intuitively, a large $U_a$ indicates that the outputs of the forget instances are highly differentiable from the outputs of the test instances, strong evidence that the influence of the forget set still exists in the unlearned model. We formulate the auditor's model as the following optimization problem

$$U_a(\theta_a, \theta_u) = M(\tilde{D}_{\theta_u}^{val}; \theta_a) \quad \text{where } \theta_a \in \mathcal{B}_{\theta_u} = \arg\max_{\theta'_a \in \mathcal{H}_a} M(\tilde{D}_{\theta_u}^{tr}; \theta'_a). \tag{1}$$

The auditing set $\tilde{D}_{\theta_u}$ is divided into the training $\tilde{D}_{\theta_u}^{tr}$ and the validation $\tilde{D}_{\theta_u}^{val}$ sets. The constraint encodes the process of learning a binary classifier. The set $\mathcal{B}_{\theta_u}$ are the auditor's best-responses to the unlearner's decision $\theta_u$, that is, a specific MIA that maximally differentiates the forget and test instances. The function $M$ is an evaluation metric for the binary classifier on a dataset. The definition of $M$ is flexible. One can use the accuracy to quantify the average performance of the classifier, where true positives are weighted equally with true negatives [4; 1]. Alternatively, an average measure may not capture real privacy threats. Instead, ROC curve or true positive rates at specified false positive rates can be used for evaluation [20].

The auditor's model exhibits a high degree of generality, unifying several advanced MIAs in the literature; this includes neural network-based attacks proposed by [24], quantile regression-based attacks from [25], and prediction confidence-based attacks by [1], etc. Under the formulation of (1), the mentioned attacks differ in 1) the hypothesis class $\mathcal{H}_a$ of the binary classifier and 2) the objective function $M$. Notice the dependence of the auditor's best-response on $\theta_u$ (i.e., $\mathcal{B}_{\theta_u}$) arising from the unlearner's first-mover advantage. The unlearner utilizes this dependence to select an unlearned model that limits the auditor's discriminative power, which we discuss next.

### 4.2 The Unlearner's Model

Next, we define the unlearner's model. Let $C_u$ represent the unlearner's cost function, which encompasses two main objectives for unlearning. The first objective is to maintain the utility of the model, ensuring that the unlearned model performs comparably (e.g., in terms of predictive power) to the original model. To achieve this objective, we minimize a loss function $L(D_r; \theta_u)$ computed on the retain set $D_r$, following the principles of empirical risk minimization. All regularization terms are included in the loss function to simplify notation. The second objective focuses on eliminating the influence of the forget set from the model being unlearned. We approach this objective adversarially by considering the auditor's utility $M$. In essence, a smaller value of the auditor's utility indicates that the forget set is harder to be distinguished from the test set, providing strong evidence that the unlearning process is effective.

Formally, the unlearner's optimization problem is to minimize the cost function below

$$C_u\left(\theta_u, \theta_a\right) = L(D_r; \theta_u) + \alpha \cdot M(\tilde{D}^{val}_{\theta_u}; \theta_a). \tag{2}$$

The parameter $\alpha \in \mathbb{R}^+$ balances the loss $L$ and the auditor's utility $M$. Depending on the specific setting, the cost function $C_u$ can be extended to incorporate additional objectives for unlearning. For instance, one can specify that the unlearned model should perform poorly on the forget set [10]; this can be achieved by minimizing an evaluation metric (e.g., likelihood) on the forget set. Also, several sparsity-promoting techniques have been shown helpful for unlearning [16]; one way to achieve this is by adding an $\ell_1$ regularization to the cost function.

### 4.3 The Stackelberg Game

Now, with the unlearner and the auditor models in place, we formally define the Stackelberg Game (SG). The SG is to solve the following bi-level optimization problem [26]

$$\min_{\theta_u \in \mathcal{H}_u} \quad \underbrace{L(D_r; \theta_u) + \alpha \cdot M(\tilde{D}^{val}_{\theta_u}; \theta_a)}_{\text{Unlearner}} \quad s.t. \quad \underbrace{\theta_a \in \mathcal{B}_{\theta_u}}_{\text{Auditor}}. \tag{3}$$

The hierarchical structure encodes the sequential order of the play, with the upper level corresponding to the unlearner's optimization problem and the lower level capturing the auditor's best-responses. During the unlearning, the unlearner needs to proactively consider the auditor's responses. This requires selecting an unlearning model where the influence of the forget set is erased, or from the auditor's perspective, the forget instances are indistinguishable from the test ones.

The key assumption of the SG is that if the forget set cannot be distinguished from the test set—in terms of the effectiveness of an MIA—its influence is deemed eliminated from the unlearned model. We justify this assumption from three angles. First, one common way to measure forgetfulness is by assessing the accuracy of the unlearned model on the forget set [10; 15; 27]. This approach is grounded on the observation that machine learning models exhibit distinct performance between training data and unseen data. However, it is important to note that accuracy on the forget set does not necessarily correlate with forgetfulness, as there are inherently difficult (or easy) instances that result in low (or high) accuracy regardless of whether they were part of the training set [20]. Secondly, MIAs have been used to study training data forgetting [17], demonstrating its utility in detecting residual traces of a dataset. Finally, from an adversarial perspective, if a sophisticated attack like an MIA cannot differentiate the forget set from the test set, it is reasonable to expect that the influence of the forget set has been removed.

We solve the SG using gradient-based methods, allowing for easy integration into end-to-end training pipelines. Specifically, we use Implicit Function Theorem to differentiate through the auditor's optimization problem (1), obtaining the gradient of the auditor's utility with respect to (w.r.t) the unlearning model's weights, i.e., $\partial M / \partial \theta_u$. As a result, the SG becomes a differentiable layer, compatible with the standard forward-backward computation. The solution methods will be detailed in the next section.

## 5 `SG-Unlearn`: Stackelberg Game Unlearn

In this section, we describe the algorithm for solving the SG. In general, it is NP-hard to find an optimal solution for the unlearner [28]. Instead, we focus on gradient-based algorithms to find an approximate solution, i.e., a model parameter $\theta_u$ exhibiting good unlearning performance. The main technical challenge is computing the gradient of the auditor's utility w.r.t. the unlearning model's weights (i.e., $\partial M / \partial \theta_u$), which requires differentiation through the auditor's optimization problem. While the differentiation can be bypassed in some special cases, e.g., when the unlearner's hypothesis class is of linear regressions [29], this is rarely applicable in the current setting given our primary focus on unlearning deep neural networks.

Our solution leverages both the Implicit Function Theorem (IFT) [30] and tools from Differentiable Optimization (DO) to compute the gradients [5; 6; 7], thereby rendering the SG a differentiable layer seamlessly integrable into existing end-to-end pipelines.

We start by expanding the gradient of $C_u$ w.r.t. $\theta_u$ using the chain rule

$$\frac{\partial C_u}{\partial \theta_u} = \frac{\partial L(D_r; \theta_u)}{\partial \theta_u} + \frac{\partial M(\tilde{D}^{val}_{\theta_u}; \theta_a)}{\partial \theta_a} \cdot \frac{\partial \theta_a}{\partial \tilde{D}^{tr}_{\theta_u}} \cdot \frac{\partial \tilde{D}^{tr}_{\theta_u}}{\partial \theta_u}. \tag{4}$$

The first term on the right-hand side can be easily computed using an automatic differentiation tool like PyTorch [31]. In essence, the computation involves passing $D_r$ through the unlearning model (i.e., $\theta_u$) in the forward pass, computing the loss $L$, and getting the gradients in the backward pass. The second term on the right is an expansion of $\partial M / \partial \theta_u$ using the chain rule; for clarity we omit the arguments of the functions. The gradient $\partial M / \partial \theta_a$ is obtained by performing a standard forward-backward pass. Some evaluation metrics for binary classification, such as the 0-1 loss, AUC, recall, etc., are non-differentiable. Therefore, we adhere to standard practices by employing a differentiable proxy for $M$, such as utilizing the logistic loss as a substitute for the 0-1 loss.

**Leveraging Implicit Function Theorem** Computing the gradient $\partial \theta_a / \partial \tilde{D}^{tr}_{\theta_u}$ requires differentiation through the attacker's optimization problem. The main challenge is the absence of an explicit function that maps $\tilde{D}^{tr}_{\theta_u}$ to $\theta_a$. However, under certain regularity assumptions, one can derive an implicit mapping between $\tilde{D}^{tr}_{\theta_u}$ and $\theta_a$ based on the optimality conditions of the auditor's optimization problem [5]. A concrete example is when the optimization problem is convex, such as learning a Support Vector Machine (SVM) [2]. In this case, the KKT conditions are necessary and sufficient conditions for the optimality, and it connects $\theta_a$ with $\tilde{D}^{tr}_{\theta_u}$ through a system of linear equations, i.e.,

$$f(\tilde{D}^{tr}_{\theta_u}, \theta_a) = 0, \tag{5}$$

where $f$ encapsulates the stationarity conditions, the primal and dual feasibility conditions, and the complementary slackness conditions [32]. For illustration purposes, a concrete example of the KKT conditions $f$ for linear SVM is provided in Appendix A.4. We apply IFT to the system of linear equations, resulting in

$$\frac{\partial \theta_a}{\partial \tilde{D}^{tr}_{\theta_u}} = -\left( \frac{\partial f(\tilde{D}^{tr}_{\theta_u}, \theta_a)}{\partial \tilde{D}^{tr}_{\theta_u}} \right)^{-1} \frac{\partial f(\tilde{D}^{tr}_{\theta_u}, \theta_a)}{\partial \theta_a}. \tag{6}$$

For further insights into differentiating through an optimization problem using the implicit function theorem, we recommend referring to the lectures by [33].

**Leveraging Differentiable Optimization** In practice, we capitalize on tools from Differentiable Optimization (DO) to compute the gradients. Intuitively, we can consider DO as software that implements IFT, as shown in equation (6), for a given optimization problem. What we need to do is describing the auditor's optimization problem using a specialized modeling language, e.g., `cvxpy` [34]. We then use DO to transform this description into a differentiable layer. Subsequently, this differentiable layer is positioned atop the model undergoing unlearning, thereby establishing a computational pathway from $\theta_u$ to $\theta_a$. The pseudocode for this process is provided in Algorithm 1. This algorithm has a time complexity of $O(n^3)$, where $n$ denotes the size of the attacker's optimization problem (i.e., the number of variables and/or constraints). This cubic dependence stems from the matrix inversion in (6).

# 6 Experiments

## 6.1 Experiment Setup

We run experiments on CIFAR-10, CIFAR-100, and SVHN, three well-known image classification datasets [35; 36]. For all experiments we use the ResNet-18 architecture [37]. We consider both *random forgetting* and *class-wise forgetting* In random forgetting, instances are sampled uniformly at random from all classes. In contrast, class-wise forgetting entails selecting all instances from a specific class. For CIFAR-10 and CIFAR-100, the forget set consists of 10% of the entire training set, while the ratio is 5% for SVHN. In all experiments, the attacker's optimization problem is formulated as a binary classification task, wherein a linear support vector machine (SVM) is employed to distinguish between forget and test instances.

The baseline methods we use to compare with SG include Retrain[3], Fine-Tune (FT) [38; 39], Gradient Ascent (GA) [40; 41], Influence Unlearning (IU) [42; 43], $\ell_1$-sparse [16], Random Label (RL) [44], Boundary Expansion (BE),

[2] This includes several state-of-the-art MIAs [22; 1].

[3] Retraining the unlearning model on $D_{tr} \setminus D_f$ from scratch.

**Algorithm 1** `SG-Unlearn`

Input: $D_r, D_f, D_{te}$ and the original model $\theta_o$
Initialize: $i = 0, \theta_u^0 = \theta_o$, a scheduler $\eta^i$
**while** $i <$ epoch **do**
  Compute $L(D_r; \theta_u^i)$ on the retain set in a forward pass
  Update $\theta_u^{i'} \leftarrow \theta_u^i - \eta^i \cdot \frac{\partial L(D_r; \theta_u^i)}{\partial \theta_u^i}$
  Construct the auditing set $\tilde{D}_{\theta_u^{i'}}$ from $D_f$ and $D_{te}$
  Describe the auditor's optimization problem (1) with `cvxpy`
  Convert the description to a differentiable layer `AuditorLayer`
  Plug `AuditorLayer` into the computational graph
  Get the auditor's best response $\theta_a^i \leftarrow$ `AuditorLayer`$(\tilde{D}_{\theta_u^{i'}})$
  Compute $M\left(\tilde{D}^{val}_{\theta_u^{i'}}; \theta_a^i\right)$
  Update $\theta_u^{i+1} \leftarrow \theta_u^{i'} - \eta^i \cdot \frac{\partial M\left(\tilde{D}^{val}_{\theta_u^{i'}}; \theta_a^i\right)}{\partial \theta_u^i}$
  $i \leftarrow i + 1$
**end while**
Return: $\theta_u^i$

Boundary Shrink (BS) [45] and SCRUB [46] Further details on the baseline methods are provided in Section A.2.3 of the Appendix. For all methods, we use the SGD optimizer with a weight decay of 5e-4 and a momentum of 0.9. Other hyper-parameters are selected through the validation set. Specifically, we create a new auditing set. For each unlearning method, we select the hyper-parameters that maximize the difference between the validation accuracy and the MIA accuracy on this new auditing set. The hyperparameters are listed in Table 5 in the Appendix.

### 6.2 Evaluation Metrics

We evaluate SG and the baseline methods using metrics previously adopted in related studies [9; 17; 16; 15]. *It is important to note that the test accuracy is evaluated on a subset of the test data that is separate from the one used for solving SG.* **Retain accuracy** ($Acc_r$) **and test accuracy** ($Acc_{te}$) are used to quantify model utility [16]. **MIA accuracy, AUC and F1 score** are the metrics to quantify the effectiveness of unlearning, which are estimated on the auditing set with 10-fold cross [20]. A good unlearning algorithm should be close to random guessing (0.5) in terms of the MIA metrics. **Forget accuracy** ($Acc_f$) measures the accuracy of the unlearned model on the forget set. An effective unlearning algorithm should produce an unlearned model where $Acc_f$ is close to the $Acc_{te}$. Indeed, if the unlearned model did not see $D_f$, its performance on this set should be consistent with that on $D_{te}$. **The absolute difference between the forget and test accuracy** ($|Acc_f - Acc_{te}|$) measures whether the unlearned model performs consistently on $D_f$ and $D_{te}$. To gather additional statistical evidence regarding the effectiveness of unlearning, we collect the cross-entropy losses of the forget and test instances from the unlearned model into the empirical distributions $\mathcal{L}_f$ and $\mathcal{L}_{te}$, respectively. Next, we run a **Kolmogorov-Smirnov statistics (KS Stat.)** test to determine if the distributions can be differentiated from each other. The KS statistic quantifies the differences between $\mathcal{L}_f$ and $\mathcal{L}_{te}$, where the p-value indicates whether the difference is significant [47]. In addition to the KS statistics, we provide the **Wasserstein distance (W. Dist.)** between the empirical distributions of $\mathcal{L}_f$ and $\mathcal{L}_{te}$. This complements the KS statistics and evaluates the unlearning performance in terms of the similarity between the losses.

### 6.3 Results

The experimental results of random forgetting and class-wise forgetting are presented in Section 6.3.1 and 6.3.2, respectively. We consider retrain as the gold standard for evaluating unlearning algorithms: the closer to the metrics of retrain the more effective the algorithm. We highlight the closest metrics to retrain in bold.

#### 6.3.1 Random Forgetting

We present the results of CIFAR-10 and CIFAR-100 in Table 1. The results for SVHN datasets are provided in A.2.1 in the Appendix. SG achieves the best performance for most of the metrics, demonstrating its effectiveness in unlearning. Specifically, the KS statistic of SG is consistently lower than those of the other baselines, exhibiting an order of magnitude difference in the statistics for CIFAR-10 compared to most baselines. Intuitively, ML models

behave differently on training data compared to unseen data, and this difference is usually reflected in the corresponding losses [20]. The small KS statistic of SG implies that the forget and test instances exhibit greater similarity in terms of the model's behavior, although there is still a discernible difference between the losses. Another metric for measuring the similarity is the Wasserstein distance (W. Dist.). The baseline RL achieves the lowest distance, although the difference with SG is not statistically significant. A visualization of the cross-entropy losses for the forget and test instances from one of the experiments is provided in Figure 4 in the Appendix.

Another observation from the table is that there is a trade-off between model performance, measured by test accuracy, and the effectiveness of unlearning, measured by MIA accuracy. Specifically, SG is more effective at unlearning the forget instances, as indicated by the highlighted MIA metrics. However, this effectiveness comes at a cost to the test accuracy on CIFAR-10 and CIFAR-100, although the degradation is not significant. We run a large array of experiments with varying $\alpha$ from $\{0.05, 0.1, 0.25, 0.5, 1, 2, 5\}$ (see (2)) to explore the extent to which the trade-off can be reduced. The results are presented in Figure 2. Unfortunately, we do not see a consistent trend that makes SG closer to retrain across all the metrics.

### 6.3.2 Class-wise Forgetting

We present CIFAR-10 here as the benchmark of class-wise forgetting. For other datasets, please refer to Section A.3 in the Appendix. In contrast to random forgetting, where the forget set is sampled uniformly at random across all classes, class-wise forgetting involves removing the influence of all instances belonging to a specific class. The experimental results are shown in Table 2. The metrics closest to retrain are highlighted. Notably, all methods perform poorly on MIA-related metrics. In other words, the auditor can readily distinguish between forget instances and test instances. This outcome is expected: unlike random forgetting, class-wise forgetting produces significantly different distributions between the forgotten instances (e.g., fruits) and the test instances (e.g., non-fruits). Consequently, the outputs of the unlearning model are expected to be easily distinguishable. Additionally, no single method consistently outperforms the others.

Table 1: Experimental results ($\text{Mean}_{\text{std}}$) on CIFAR-10 and CIFAR-100 for random forgetting. The highlighted metrics are the closest to those of retraining, which is considered as the best performance compared with the other baselines.

| CIFAR-10 | $Acc_r$ | $Acc_{te}$ | $Acc_f$ | $\lvert Acc_f - Acc_{te}\rvert$ | MIA acc. | MIA AUC | MIA F1 | KS Stat. | W. Dist. | RTE (min., ↓) |
|---|---|---|---|---|---|---|---|---|---|---|
| Retrain | $0.9996_{0.0001}$ | $0.9291_{0.0022}$ | $0.9230_{0.0043}$ | 0.0061 | $0.5069_{0.0073}$ | $0.5083_{0.0099}$ | $0.5094_{0.0187}$ | $0.0255_{0.0080}$ | $0.0307_{0.0115}$ | 14.92 |
| FT | $0.9886_{0.0055}$ | $0.9114_{0.0050}$ | $0.9851_{0.0056}$ | 0.0737 | $0.5405_{0.0031}$ | $0.5457_{0.0070}$ | $0.6293_{0.0127}$ | $0.0933_{0.0050}$ | $0.3158_{0.0142}$ | 0.45 |
| GA | $\mathbf{0.9996}_{0.0001}$ | $0.9304_{0.0007}$ | $0.9995_{0.0003}$ | 0.0691 | $0.5504_{0.0066}$ | $0.5611_{0.0071}$ | $0.6625_{0.0106}$ | $0.1403_{0.0037}$ | $0.2782_{0.0026}$ | 0.18 |
| IU | $0.9723_{0.0255}$ | $0.8966_{0.0242}$ | $0.9722_{0.0243}$ | 0.0756 | $0.5398_{0.0055}$ | $0.5548_{0.0076}$ | $0.6193_{0.0204}$ | $0.1050_{0.0192}$ | $0.4083_{0.0554}$ | 0.02 |
| $\ell_1$-sparse | $0.9970_{0.0007}$ | $0.9234_{0.0014}$ | $0.9938_{0.0016}$ | 0.0704 | $0.5501_{0.0049}$ | $0.5694_{0.0087}$ | $0.6405_{0.6259}$ | $0.1018_{0.0034}$ | $0.2704_{0.0074}$ | 0.96 |
| RL | $0.9988_{0.0001}$ | $0.9217_{0.0008}$ | $0.9810_{0.0025}$ | 0.0593 | $0.5217_{0.0087}$ | $0.5297_{0.0133}$ | $\mathbf{0.5935}_{0.0140}$ | $0.0986_{0.0136}$ | $\mathbf{0.1520}_{0.0058}$ | 0.84 |
| BE | $\mathbf{0.9996}_{0.0001}$ | $0.9304_{0.0007}$ | $0.9996_{0.0003}$ | 0.0692 | $0.5541_{0.0049}$ | $0.5639_{0.0058}$ | $0.6629_{0.0082}$ | $0.1412_{0.0030}$ | $0.2783_{0.0020}$ | 0.27 |
| BS | $0.9995_{0.0001}$ | $0.9307_{0.0008}$ | $0.9995_{0.0004}$ | 0.0688 | $0.5588_{0.0072}$ | $0.5779_{0.0097}$ | $0.6590_{0.0156}$ | $0.1466_{0.0032}$ | $0.3072_{0.0026}$ | 0.46 |
| SCRUB | $0.9971_{0.0018}$ | $\mathbf{0.9251}_{0.0018}$ | $0.9959_{0.0022}$ | 0.0708 | $0.5533_{0.0059}$ | $0.5679_{0.0073}$ | $0.6337_{0.0149}$ | $0.1038_{0.0071}$ | $0.2485_{0.0154}$ | 1.30 |
| SG | $0.9948_{0.0029}$ | $0.8940_{0.0048}$ | $\mathbf{0.9351}_{0.0070}$ | **0.0411** | $\mathbf{0.5202}_{0.0054}$ | $\mathbf{0.5134}_{0.0084}$ | $0.6480_{0.0043}$ | $\mathbf{0.0482}_{0.0082}$ | $0.1555_{0.0194}$ | 1.47 |
| **CIFAR-100** | $Acc_r$ | $Acc_{te}$ | $Acc_f$ | $\lvert Acc_f - Acc_{te}\rvert$ | MIA acc. | MIA AUC | MIA F1 | KS Stat. | W. Dist. | RTE (min., ↓) |
| Retrain | $0.9996_{0.0001}$ | $0.7035_{0.0025}$ | $0.6925_{0.0039}$ | 0.0110 | $0.5184_{0.0057}$ | $0.5281_{0.0053}$ | $0.5104_{0.0081}$ | $0.0203_{0.0045}$ | $0.0567_{0.0200}$ | 13.08 |
| FT | $0.9991_{0.0001}$ | $0.7117_{0.0021}$ | $0.9984_{0.0006}$ | 0.2867 | $0.6630_{0.0075}$ | $0.7300_{0.0102}$ | $0.6878_{0.0107}$ | $0.4566_{0.0083}$ | $1.2583_{0.0166}$ | 0.39 |
| GA | $\mathbf{0.9996}_{0.0001}$ | $0.7158_{0.0008}$ | $0.9996_{0.0002}$ | 0.2838 | $0.6977_{0.0060}$ | $0.7601_{0.0065}$ | $0.7207_{0.0088}$ | $0.4915_{0.0030}$ | $1.2219_{0.0038}$ | 0.20 |
| IU | $0.9971_{0.0029}$ | $\mathbf{0.7026}_{0.0080}$ | $0.9959_{0.0034}$ | 0.2933 | $0.6660_{0.0089}$ | $0.7305_{0.0134}$ | $0.6950_{0.0124}$ | $0.4583_{0.0168}$ | $1.2612_{0.0366}$ | 0.21 |
| $\ell_1$-sparse | $0.9958_{0.0013}$ | $0.7095_{0.0025}$ | $0.9890_{0.0028}$ | 0.2785 | $0.6738_{0.0081}$ | $0.7392_{0.0088}$ | $0.6952_{0.0073}$ | $0.3717_{0.0113}$ | $1.1157_{0.0079}$ | 0.84 |
| RL | $0.9965_{0.0054}$ | $0.6665_{0.0031}$ | $0.8483_{0.0447}$ | 0.1818 | $0.5808_{0.0426}$ | $0.6152_{0.0527}$ | $0.6080_{0.0466}$ | $0.2323_{0.0731}$ | $0.8067_{0.1705}$ | 0.73 |
| BE | $0.9995_{0.0001}$ | $0.7173_{0.0014}$ | $0.9996_{0.0002}$ | 0.2823 | $0.6977_{0.0037}$ | $0.7661_{0.0066}$ | $0.7248_{0.0065}$ | $0.4940_{0.0031}$ | $1.2175_{0.0119}$ | 0.23 |
| BS | $0.9995_{0.0001}$ | $0.7160_{0.0013}$ | $0.9996_{0.0002}$ | 0.2836 | $0.6987_{0.0052}$ | $0.7651_{0.0072}$ | $0.7239_{0.0054}$ | $0.4963_{0.0033}$ | $1.2382_{0.0206}$ | 0.39 |
| SCRUB | $0.9993_{0.0001}$ | $0.7097_{0.0019}$ | $0.9991_{0.0004}$ | 0.2894 | $0.7015_{0.0057}$ | $0.7747_{0.0060}$ | $0.7299_{0.0056}$ | $0.4717_{0.0052}$ | $1.2280_{0.0128}$ | 1.14 |
| SG | $0.8993_{0.0105}$ | $0.6378_{0.0066}$ | $\mathbf{0.7239}_{0.0093}$ | **0.0861** | $\mathbf{0.5412}_{0.0070}$ | $\mathbf{0.5320}_{0.0076}$ | $\mathbf{0.6061}_{0.0056}$ | $\mathbf{0.0988}_{0.0061}$ | $\mathbf{0.5316}_{0.0295}$ | 3.07 |

Table 2: Experimental results ($\text{Mean}_{\text{std}}$) on CIFAR-10 for class-wise forgetting. The highlighted metrics are the closest to those of retraining, which is considered as the best performance compared with the other baselines.

| CIFAR-10 | $Acc_r$ | $Acc_{te}$ | $Acc_f$ | $\lvert Acc_f - Acc_{te}\rvert$ | MIA acc. | MIA AUC | MIA F1 | KS Stat. | W. Dist. | RTE (min., ↓) |
|---|---|---|---|---|---|---|---|---|---|---|
| Retrain | $0.9996_{0.0001}$ | $0.9333_{0.0009}$ | $0.0000_{0.0000}$ | 0.9333 | $0.9935_{0.0006}$ | $0.9983_{0.0004}$ | $0.9936_{0.0007}$ | $0.9803_{0.0002}$ | $9.5601_{0.0911}$ | 13.96 |
| FT | $0.9958_{0.0022}$ | $0.9226_{0.0030}$ | $0.6043_{0.0450}$ | 0.3183 | $0.9915_{0.0011}$ | $\mathbf{0.9985}_{0.0002}$ | $0.9915_{0.0011}$ | $0.7975_{0.0133}$ | $0.9324_{0.1847}$ | 1.16 |
| GA | $0.8478_{0.0046}$ | $0.7942_{0.0055}$ | $0.0007_{0.0002}$ | 0.7935 | $\mathbf{0.9944}_{0.0011}$ | $0.9996_{0.0002}$ | $\mathbf{0.9938}_{0.0015}$ | $0.9269_{0.0087}$ | $15.2941_{0.1656}$ | 0.84 |
| IU | $0.9339_{0.0161}$ | $0.8644_{0.0141}$ | $0.0619_{0.0149}$ | 0.8025 | $0.9972_{0.0009}$ | $0.9996_{0.0001}$ | $0.9972_{0.0007}$ | $0.8151_{0.0195}$ | $\mathbf{8.2574}_{0.6484}$ | 0.31 |
| $\ell_1$-sparse | $0.9972_{0.0005}$ | $0.9285_{0.0014}$ | $0.0914_{0.0310}$ | 0.8371 | $0.9910_{0.0014}$ | $0.9989_{0.0001}$ | $0.9910_{0.0014}$ | $0.9208_{0.0078}$ | $2.5552_{0.1738}$ | 1.84 |
| RL | $\mathbf{0.9996}_{0.0000}$ | $\mathbf{0.9330}_{0.0008}$ | $0.0001_{0.0001}$ | **0.9329** | $0.9916_{0.0013}$ | $0.9990_{0.0005}$ | $0.9916_{0.0013}$ | $0.9695_{0.0025}$ | $6.3989_{0.0789}$ | 1.97 |
| BE | $0.9710_{0.0012}$ | $0.8984_{0.0023}$ | $0.2477_{0.0022}$ | 0.6507 | $0.9964_{0.0005}$ | $0.9990_{0.0004}$ | $0.9964_{0.0005}$ | $0.7306_{0.0047}$ | $4.8984_{0.0432}$ | 0.32 |
| BS | $0.9691_{0.0031}$ | $0.8969_{0.0031}$ | $0.2504_{0.0105}$ | 0.6465 | $0.9965_{0.0001}$ | $0.9988_{0.0005}$ | $0.9965_{0.0002}$ | $0.7196_{0.0072}$ | $5.0155_{0.0922}$ | 0.66 |
| SCRUB | $1.0000_{0.0000}$ | $0.9269_{0.0021}$ | $\mathbf{0.0000}_{0.0000}$ | 0.9269 | $1.0000_{0.0000}$ | $1.0000_{0.0000}$ | $1.0000_{0.0000}$ | $0.9999_{0.0001}$ | $70.9934_{2.9441}$ | 3.47 |
| SG | $0.9667_{0.0054}$ | $0.9056_{0.0055}$ | $\mathbf{0.0000}_{0.0000}$ | 0.9056 | $0.9814_{0.0026}$ | $0.9902_{0.0025}$ | $0.9818_{0.0025}$ | $\mathbf{0.9696}_{0.0032}$ | $5.2754_{0.1882}$ | 0.84 |

### 6.3.3 The Effect of the Attacker Model

Finally, we conduct a comparative study to understand the impact of adversarial modeling on the unlearning process, controlled by the parameter $\alpha$ as defined in (2). We show the results for random forgetting and defer the results for class-wise forgetting to the appendix. In Figure 1, we compare two cases where $\alpha$ is set to either 1 or 0, denoted by SG-1 and SG-0 respectively. The comparison is done across four metrics: 1) the test accuracy ; 2) the MIA accuracy; 3) the defender's utility, evaluated as the test accuracy minus the MIA accuracy, which provides a combined scalar value that measures both the performance of the unlearned model and the effectiveness of unlearning; 4) the Wasserstein distance between the empirical distributions of $\mathcal{L}_f$ and $\mathcal{L}_{te}$. We show the averages over 10 experiments with different seeds, and 95% confidence intervals are displayed. The first observation is that the adversarial term (i.e., $\alpha \cdot M(\tilde{D}^{val}_{\theta_u}; \theta_a)$) acts as a regularizer, improving the generalizability of the unlearned model. This observation is supported by comparing the test accuracy of SG-1 and SG-0 on CIFAR-10 (top middle). Similar findings have been reported in [24]. Another observation is that adversarial modeling limits the attacker's ability to differentiate between forget instances and test instances; this is demonstrated by the MIA accuracy on CIFAR-100. The right-most column displays the Wasserstein distances between $\mathcal{L}_f$ and $\mathcal{L}_{te}$. It is evident that the two losses are closer as a result of adversarial modeling, especially for CIFAR-100 dataset. Additionally, the distances progressively decrease throughout the epochs, confirming the effectiveness of the gradient-based method.

In addition to the existence of attacker, we also investigate the strength of the attacker by changing $\alpha$. We select the $\alpha$ in large range of $\{0.05, 0.1, 0.25, 0.5, 1, 2, 5\}$. In Figure 2, we compare the performance regarding the test accuracy $Acc_{te}$. The cross of the red dash line is the performance of the retrain model. We can find that SG is robust to the attacker strength.

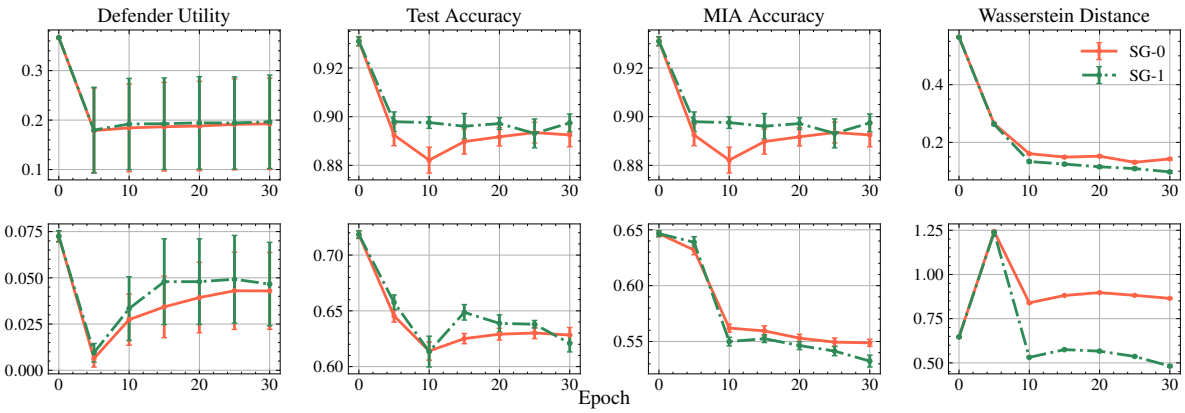


Figure 1: An ablation study to understand the impact of adversarial modeling on the process of unlearning; $\alpha = 1$ and $\alpha = 0$ corresponds to the cases with and without adversarial modeling, respectively. The results are the averages over 10 experiments with different seeds, and 95% confidence intervals are displayed. **From the left to the right**: 1) the defender's utility, evaluated as the test accuracy $Acc_{te}$ minus the MIA accuracy; 2) test accuracy; 3) MIA accuracy; 4) Wasserstein distance between the cross-entropy losses of the forget and test instances. **Top row**: CIFAR-10; **Bottom row**: CIFAR-100. **Epoch 0: Original model.**

## 7 Discussion

In this paper, we design an adversarial framework for addressing the problem of unlearning a set data from a machine learning model. Our approach focuses on evaluating the effectiveness of unlearning from an adversarial perspective, leveraging membership inference attacks (MIAs) to detect any residual traces of the data within the model. The framework allows for a proactive design of the unlearning algorithm, synthesizing two lines of research—machine unlearning and MIAs—that have heretofore progressed in parallel. By using implicit differentiation techniques, we develop a gradient-based algorithm for solving the game, making the framework easily integrable into existing end-to-end learning pipelines. We present empirical results to support the efficacy of the framework and the algorithm. We believe our work can make a progress in trustworthy ML.

**Limitation** One future direction is to enhance the algorithm's efficiency. As shown in (6), the gradient-based algorithm requires a matrix inversion, which exhibits an $O(n^3)$ dependence on the size of the auditor's optimization problem. Therefore, developing a more efficient method to differentiate through the auditor's optimization could significantly accelerate the algorithm. Another direction is to experiment with different combinations of the unlearning algorithm

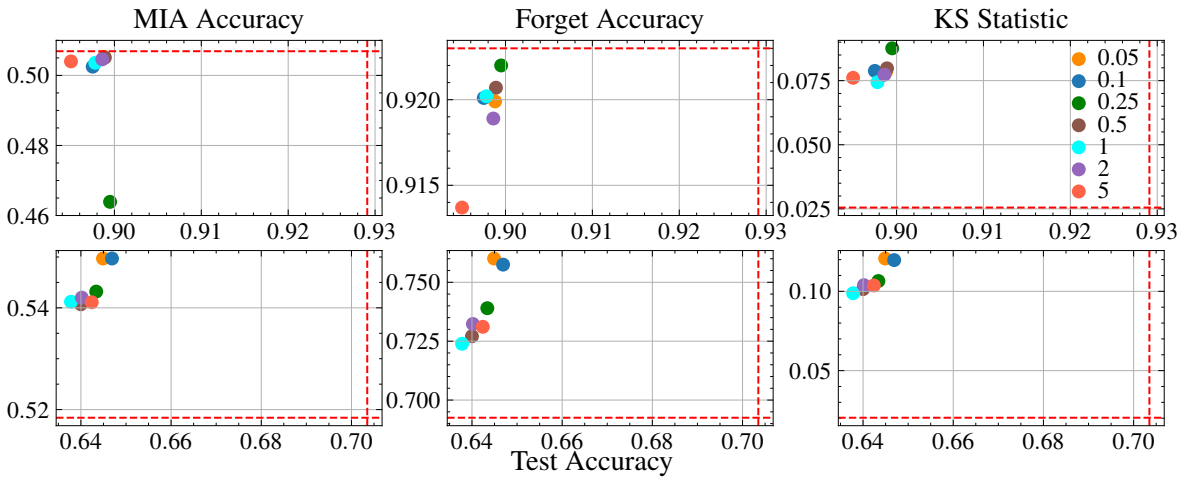


Figure 2: Experiments with different values of the trade-off parameter $\alpha$. We consider 7 values $\{0.05, 0.1, 0.25, 0.5, 1, 2, 5\}$. Each dot represents a batch of 5 random experiments with the same $\alpha$. The coordinates of a dot are the corresponding metrics averaged over the 5 runs. **Top row**: CIFAR-10; **Bottom row**: CIFAR-100.

and the MIA within the SG framework. Currently, the unlearner employs Fine-Tune as the unlearning algorithm, while the auditor uses an SVM-based MIA. Exploring the performance of other combinations, such as Random Label with a neural network-based MIA, would be worthwhile.

## References

[1] Liwei Song and Prateek Mittal. Systematic evaluation of privacy risks of machine learning models. In *Proceedings of the 30th USENIX Security Symposium (USENIX)*, pages 2615–2632, 2021.

[2] Chuan Guo, Tom Goldstein, Awni Y. Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pages 3832–3842. PMLR, 2020.

[3] Anvith Thudi, Hengrui Jia, Ilia Shumailov, and Nicolas Papernot. On the necessity of auditable algorithmic definitions for machine unlearning. In Kevin R. B. Butler and Kurt Thomas, editors, *Proceedings of the Thirty-first USENIX Security Symposium (USENIX)*, pages 4007–4022, 2022.

[4] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pages 3–18. IEEE, 2017.

[5] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *CoRR*, abs/1607.05447, 2016.

[6] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 136–145. PMLR, 2017.

[7] Akshay Agrawal, Brandon Amos, Shane T. Barratt, Stephen P. Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, pages 9558–9570, 2019.

[8] Zhongruo Wang, Krishnakumar Balasubramanian, Shiqian Ma, and Meisam Razaviyayn. Zeroth-order algorithms for nonconvex–strongly-concave minimax problems with improved complexities. *Journal of Global Optimization*, 87(2):709–740, 2023.

[9] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, pages 141–159. IEEE, 2021.

[10] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the 33rd Conference on Artificial Intelligence (AAAI)*, pages 11516–11524. AAAI Press, 2021.

[11] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. Graph unlearning. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 499–513, 2022.

[12] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9301–9309, 2020.

[13] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130, pages 2008–2016, 2021.

[14] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. In *Proceedings of the 30th Annual Network and Distributed System Security Symposium (NDSS)*, 2023.

[15] Vikram S. Chundawat, Ayush K. Tarun, Murari Mandal, and Mohan S. Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Proceedings of the Thirty-Seventh Conference on Artificial Intelligence (AAAI)*, pages 7210–7217, 2023.

[16] Jinghan Jia, Jiancheng Liu, Parikshit Ram, Yuguang Yao, Gaowen Liu, Yang Liu, Pranay Sharma, and Sijia Liu. Model sparsification can simplify machine unlearning. *CoRR*, abs/2304.04934, 2023.

[17] Matthew Jagielski, Om Thakkar, Florian Tramèr, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *Proceedings the 11th International Conference on Learning Representations (ICLR)*, 2023.

[18] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *CoRR*, abs/2209.02299, 2022.

[19] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *Proceedings of the ACM Conference on Computer and Communications (CCS)*, pages 896–911. ACM, 2021.

[20] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P)*, pages 1897–1914. IEEE, 2022.

[21] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. In *Proceedings of the Conference on Computer and Communications Security (CCS)*, pages 3093–3106. ACM, 2022.

[22] Martin Bertran, Shuai Tang, Michael Kearns, Jamie Morgenstern, Aaron Roth, and Zhiwei Steven Wu. Scalable membership inference attacks via quantile regression. *CoRR*, abs/2307.03694, 2023.

[23] Sasi Kumar Murakonda and Reza Shokri. Ml privacy meter: Aiding regulatory compliance by quantifying the privacy risks of machine learning. *CoRR*, abs/2007.09339, 2020.

[24] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine learning with membership privacy using adversarial regularization. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 634–646, 2018.

[25] Theo Bertram, Elie Bursztein, Stephanie Caro, Hubert Chao, Rutledge Chin Feman, Peter Fleischer, Albin Gustafsson, Jess Hemerly, Chris Hibbert, Luca Invernizzi, et al. Five years of the right to be forgotten. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 959–972, 2019.

[26] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153:235–256, 2007.

[27] Thomas Baumhauer, Pascal Schöttle, and Matthias Zeppelzauer. Machine unlearning: Linear filtration for logit-based classifiers. *Machine Learning*, 111(9):3203–3226, 2022.

[28] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In Joan Feigenbaum, John C.-I. Chuang, and David M. Pennock, editors, *Proceedings 7th ACM Conference on Electronic Commerce (EC)*, pages 82–90. ACM, 2006.

[29] Liang Tong, Sixie Yu, Scott Alfeld, and Yevgeniy Vorobeychik. Adversarial regression with multiple learners. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 4953–4961, 2018.

[30] Asen L Dontchev, R Tyrrell Rockafellar, and R Tyrrell Rockafellar. *Implicit functions and solution mappings: A view from variational analysis*, volume 616. Springer, 2009.

[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[32] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[33] Stephen Gould. Lecture notes on differentiable optimisation in deep learning. *CoRR*, 2023.

[34] Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

[35] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[36] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[38] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*, 2021.

[39] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020.

[40] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021.

[41] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319. IEEE, 2022.

[42] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021.

[43] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[44] Tomohiro Hayase, Suguru Yasutomi, and Takashi Katoh. Selective forgetting of deep networks at a finer level than samples. *arXiv preprint arXiv:2012.11849*, 2020.

[45] Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7766–7775, 2023.

[46] Meghdad Kurmanji, Peter Triantafillou, Jamie Hayes, and Eleni Triantafillou. Towards unbounded machine unlearning. *Advances in Neural Information Processing Systems*, 36, 2024.

[47] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

[48] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1885–1894, 2017.

# A Appendix

## A.1 Notation Table

Table 3: A summary of the notations used in the paper

| Notation | Meaning |
|---|---|
| $\mathcal{D} = \{(x_i, y_i)\}$ | A dataset |
| $(x_i, y_i)$ | One data point where $x_i$ is the feature while $y_i$ is the label |
| $\mathcal{X}, \mathcal{Y}$ | The feature space and the label space |
| $D_f, D_{te}, D_{val}, D_{tr}, D_r$ | The forgetting, testing, validation, training, and retain set |
| $(x_j^f, y_j^f)$ | One data point from the forget set $D_f$ |
| $\mathcal{A}$ | A machine learning algorithm |
| $\mathcal{U}$ | A machine unlearning algorithm |
| $\theta_o$ | The original model, i.e., $\mathcal{A}(D_{tr})$ |
| $\theta_u$ | The unlearned model, i.e., $\mathcal{U}(\theta_o)$ |
| $\theta_r$ | The retrained model, i.e., $\mathcal{A}(D_r)$ |
| $\tilde{D}_{\theta_u}$ | The auditing dataset for membership inference attack |
| $s_j^f$ | The output of a forget instance in the auditing dataset $\tilde{D}_{\theta_u}$ from $\theta_u$ |
| $s_j^{te}$ | The output of a testing instance in the auditing dataset $\tilde{D}_{\theta_u}$ from $\theta_u$ |
| $\tilde{D}_{\theta_u}^{tr}, \tilde{D}_{\theta_u}^{val}$ | The training and validation split of $\tilde{D}_{\theta_u}$. |
| $C_u$ | The unlearner's cost function |
| $\mathcal{B}_{\theta_u}$ | The auditor's best response given an unlearning model $\theta_u$ |
| $U_a$ | The utility function of the auditor |
| $\mathcal{H}_a$ | The hypothesis class of the auditor |
| $\mathcal{H}_u$ | The hypothesis class of the unlearner |
| $\alpha$ | The trade-off factor as defined in the unlearner's cost function (2) |

## A.2 Random Forgetting

### A.2.1 SVHN Dataset

The results of SVHN dataset on random forgetting are given in Table 4.

Table 4: Experimental results ($\text{Mean}_{\text{std}}$) on SVHN for random forgetting. The highlighted metrics are the closest to those of retraining, which is considered as the best performance compared with the other baselines.

| SVHN | $Acc_r$ | $Acc_{te}$ | $Acc_f$ | $\lvert Acc_f - Acc_{te}\rvert$ | MIA acc. | MIA AUC | MIA F1 | KS Stat. | W. Dist. | RTE (min., ↓) |
|---|---|---|---|---|---|---|---|---|---|---|
| Retrain | $0.9959_{0.0002}$ | $0.9610_{0.0010}$ | $0.9534_{0.0024}$ | 0.0076 | $0.5248_{0.0058}$ | $0.5422_{0.0075}$ | $0.5149_{0.0157}$ | $0.0306_{0.0117}$ | $0.0686_{0.0145}$ | 20.46 |
| FT | $0.9991_{0.0001}$ | $0.7117_{0.0021}$ | $0.9876_{0.0070}$ | 0.2867 | $0.5372_{0.0123}$ | $0.5592_{0.0121}$ | $0.5523_{0.0170}$ | $0.0613_{0.0342}$ | $0.1743_{0.0091}$ | 1.55 |
| GA | $0.9954_{0.0001}$ | $0.9641_{0.0002}$ | $0.9949_{0.0006}$ | 0.0308 | $0.5191_{0.0072}$ | $0.5411_{0.0051}$ | $0.5500_{0.0178}$ | $0.0867_{0.0065}$ | $0.1473_{0.0026}$ | 0.97 |
| IU | $0.9076_{0.0707}$ | $0.8817_{0.0658}$ | $0.9050_{0.0713}$ | 0.0233 | $0.5373_{0.0116}$ | $0.5580_{0.0097}$ | $0.5469_{0.0187}$ | $0.0473_{0.0207}$ | $0.1407_{0.0882}$ | 0.41 |
| $\ell_1$-sparse | $0.9378_{0.0615}$ | $0.9191_{0.0540}$ | $0.9298_{0.0620}$ | 0.0107 | $0.5457_{0.0220}$ | $0.5665_{0.0229}$ | $0.5347_{0.0324}$ | $\mathbf{0.0396}_{0.0112}$ | $0.1158_{0.0954}$ | 1.86 |
| RL | $0.9949_{0.0002}$ | $\mathbf{0.9609}_{0.0006}$ | $0.9797_{0.0018}$ | 0.0188 | $0.5211_{0.0106}$ | $0.5411_{0.0147}$ | $\mathbf{0.5144}_{0.0225}$ | $0.1079_{0.0175}$ | $\mathbf{0.0642}_{0.0060}$ | 2.65 |
| BE | $0.9955_{0.0001}$ | $0.9633_{0.0002}$ | $0.9955_{0.0006}$ | 0.0322 | $0.5209_{0.0090}$ | $0.5441_{0.0064}$ | $0.5553_{0.0175}$ | $0.1016_{0.0062}$ | $0.1528_{0.0019}$ | 0.46 |
| BS | $\mathbf{0.9956}_{0.0002}$ | $0.9641_{0.0001}$ | $0.9952_{0.0008}$ | 0.0311 | $0.5322_{0.0060}$ | $0.5509_{0.0034}$ | $0.5594_{0.0176}$ | $0.0994_{0.0074}$ | $0.1404_{0.0033}$ | 0.81 |
| SCRUB | $0.9832_{0.0010}$ | $0.9559_{0.0014}$ | $0.9809_{0.0020}$ | 0.0250 | $\mathbf{0.5273}_{0.0031}$ | $\mathbf{0.5431}_{0.0103}$ | $0.5296_{0.0196}$ | $0.0492_{0.0139}$ | $0.1032_{0.0141}$ | 1.85 |
| SG | $0.9686_{0.0017}$ | $0.9576_{0.0033}$ | $\mathbf{0.9560}_{0.0027}$ | **0.0016** | $0.5012_{0.0052}$ | $0.5089_{0.0272}$ | $0.3292_{0.1798}$ | $0.0594_{0.0233}$ | $0.0185_{0.0041}$ | 3.16 |

The comparison of SG on SVHN for random forgetting with and without attacker is illustrated in Figure 3.

### A.2.2 Loss Distributions

A visualization of the cross-entropy losses of the forget and test instances is in Figure 4.

### A.2.3 Baseline Methods

**Retrain:** The first baseline is retraining, where the unlearned model is obtained by training on the retain set from scratch. We aim to develop unlearning algorithms so that the metrics they produce are as closely aligned with those of the retraining as possible.

Figure 3: An ablation study to understand the impact of adversarial modeling on the process of unlearning; $\alpha = 1$ and $\alpha = 0$ corresponds to the cases with and without adversarial modeling, respectively. The results are the averages over 10 experiments with different seeds, and 95% confidence intervals are displayed. **From the left to the right**: 1) the defender's utility, evaluated as the test accuracy $Acc_{te}$ minus the MIA accuracy; 2) test accuracy; 3) MIA accuracy; 4) Wasserstein distance between the cross-entropy losses of the forget and test instances.

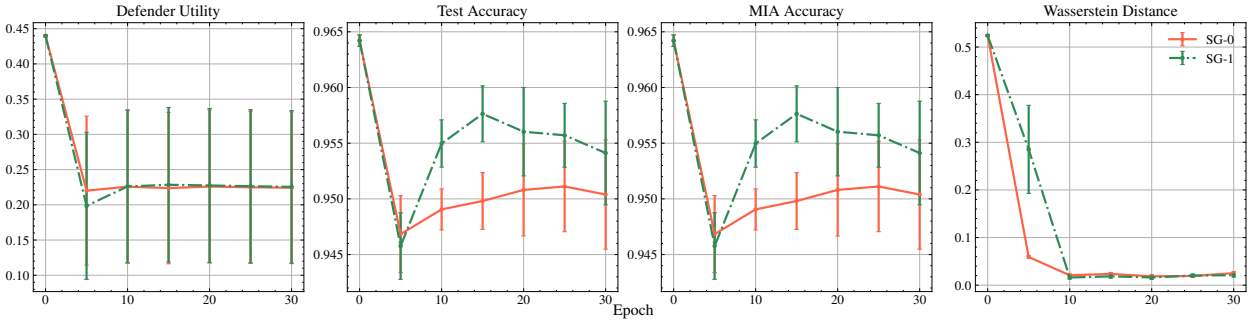


Table 5: The hyper-parameter for the baseline method and SG used in this paper.

| Parameters | Retrain | FT | GA | IU | $\ell_1$-sparse | RL | BE | BS | SCRUB | SG |
|---|---|---|---|---|---|---|---|---|---|---|
| Learning rate | 1e-2 | 5e-2 | 1e-3 | × | 1e-2 | 1e-2 | 1e-5 | 1e-5 | 5e-4 | 1e-2 |
| Num. of epoch | 160 | 30 | 5 | × | 10 | 10 | 10 | 10 | 10 | 30 |
| $\gamma$ | × | × | × | × | 5e-4 | × | × | × | × | × |
| $\alpha$ | × | × | × | 10 | × | × | × | × | × | × |
| $T$ | × | × | × | × | × | × | × | × | 4 | × |
| Decay epochs | × | × | × | × | × | × | × | × | [3, 5, 9] | × |
| $\beta$ | × | × | × | × | × | × | × | × | 0.1 | × |
| Attacker $\alpha$ | × | × | × | × | × | × | × | × | × | 1.0 |

**Fine-Tuning (FT):** As the second baseline, FT continues to train the original model on the retain set for a few epochs. This a standard baseline used in various prior research [10; 14].

**Gradient Ascent (GA):** This baseline takes the original model as the starting point and runs a few epochs of gradient ascent on the forget set $D_f$. The intuition is to disrupt the model's generalizability on $D_f$ [10]. Another name of GA is NegGrad [46].

**Influence Unlearning (IU):** This baseline uses Influence Function to estimate the updates required for a model's weights as a result of removing the forget set from the training data [13; 48].

**$\ell_1$-sparse:** This baseline integrates an $\ell_1$ norm-based sparse penalty into machine unlearning loss [16].

**Random Label (RL):** This baseline trains the original model on the retain set and the forgetting set $D_f$ whose labels are random to make the model unlearn $D_f$ while keep the model capability as much as possible.

**Boundary Expansion:** This baseline proposes a neighbor searching method to identify the nearest but incorrect class labels to guide the way of boundary shifting.

**Boundary Shrink:** This baseline artificially assigns forgetting samples to an extra shadow class of the original model [45].

**SCRUB:** This baseline achieve MU by using a teacher model and student model [46].

### A.2.4 Experiment Details

The hyperparameters used for SG and the baselines are in Table 5. The losses for the retraining baseline across the epochs are displayed in Figure 5. We run all the experiments using PyTorch 1.12 on NVIDIA A5000 GPUs and AMD EPYC 7513 32-Core Processor.

## A.3 Class-wise Forgetting

The results of SVHN dataset on class-wise forgetting are given in Table 6.

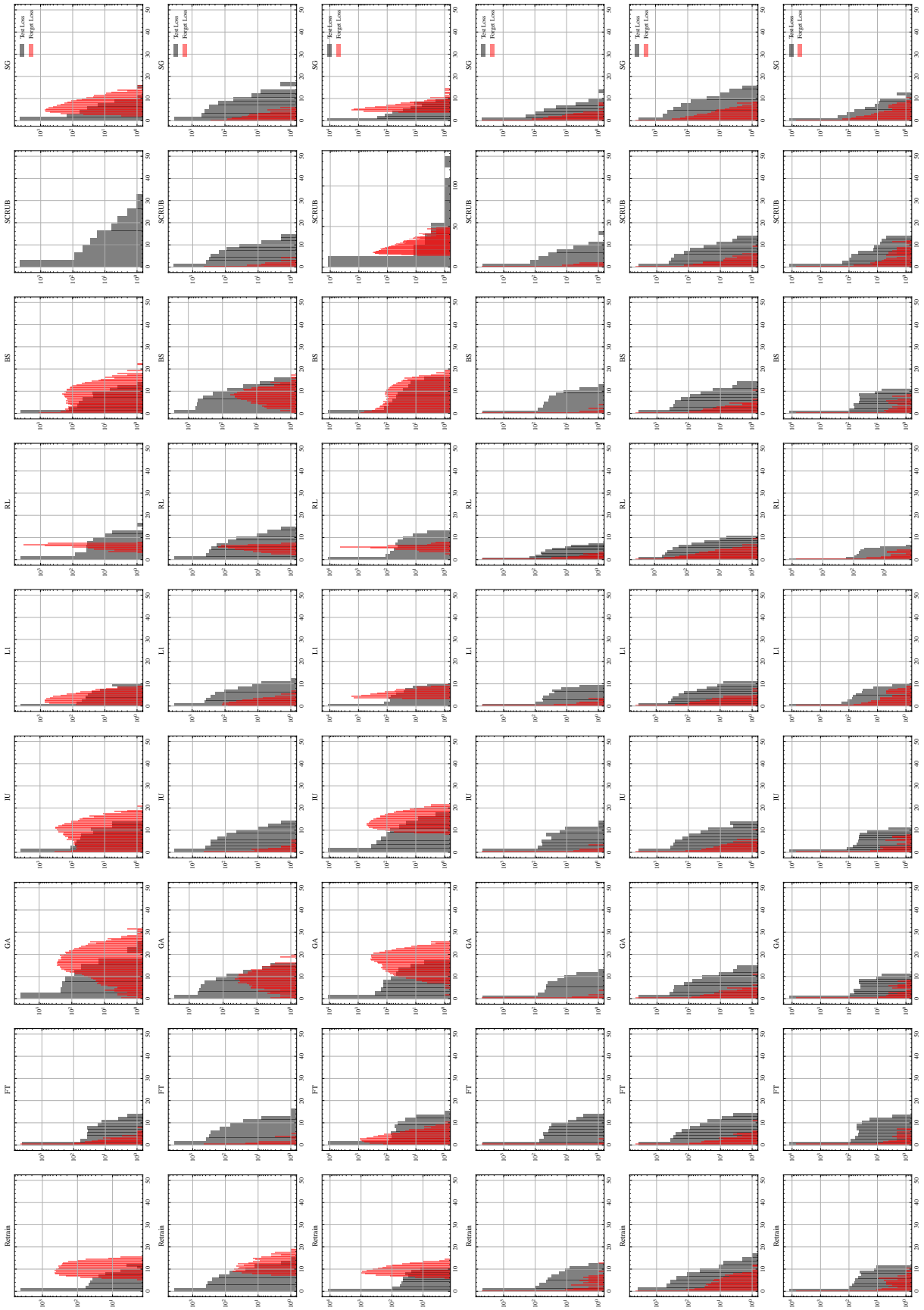


Figure 4: The distributions of the cross-entropy losses for the forget and test instances from the unlearned models. The y-axis is in log scale for better visualization. From the first to the last figure, they are random forgetting on CIFAR-10, SVHN and class-wise forgetting on CIFAR-10, CIFAR-100, SVHN.

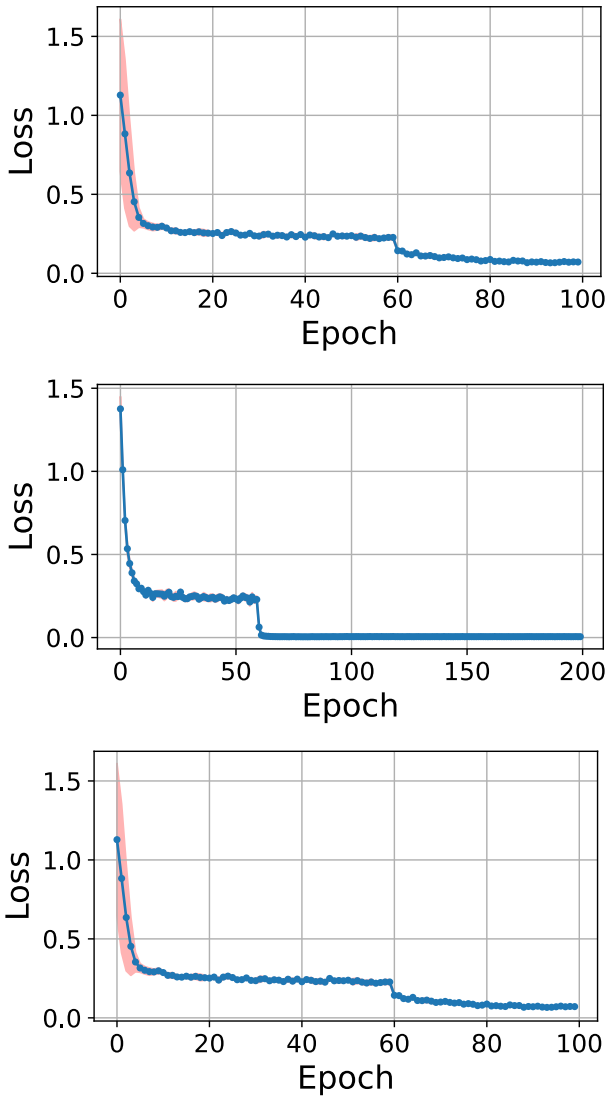


Figure 5: The training loss for the retrain baseline. For CIFAR10 and CIFAR100, the learning rate is multiplied by 0.1 when epoch is at 60, 120, 160; for SVHN, the same multiplication is done at epoch 60, 120. **Top to bottom**: CIFAR10, CIFAR100, SVHN.

### A.4 An example of the condition in (5)

In this section, we provide a concrete example of the KKT conditions for linear support vector machines (SVM). As described in Section 1, the KKT conditions are key to relating the attacker's model parameters, denoted as $\theta_a$, with the auditing set $\tilde{D}_{\theta_u}$, which allows us to derive the gradient $\partial\theta_a/\partial\tilde{D}_{\theta_u}$. The conditions $f$ can be similarly derived for any model where the learning problem is convex. To simplify the notations, we use $\{(x_i, y_i)\}_{i=1}^q$ to represent $\tilde{D}_{\theta_u}$. A standard formulation of the linear SVM is as follows

$$\begin{aligned} \min_{\theta_a, b} \quad & \frac{1}{2}\|\theta_a\|^2 \\ s.t. \quad & y_i \cdot (\theta_a^\top x_i + b) \geq 1, \forall i, \end{aligned} \tag{7}$$

where $b$ is the bias term. The standard form is typically formulated as a minimization problem, so the attacker is to maximize $V = -\frac{1}{2}\|\theta_a\|^2$. Eq. (7) is a convex program, and the optimal solution (i.e., $\theta_a^*$ and $b^*$) is characterized by the

Table 6: Experimental results ($\text{Mean}_{\text{std}}$) on SVHN for classwise forgetting. The highlighted metrics are the closest to those of retraining, which is considered as the best performance compared with the other baselines.

| SVHN | $Acc_r$ | $Acc_{te}$ | $Acc_f$ | $\lvert Acc_f - Acc_{te}\rvert$ | MIA acc. | MIA AUC | MIA F1 | KS Stat. | W. Dist. | RTE (min., ↓) |
|---|---|---|---|---|---|---|---|---|---|---|
| Retrain | $0.9963_{0.0001}$ | $0.9639_{0.0007}$ | $0.0000_{0.0000}$ | 0.9639 | $0.9950_{0.0005}$ | $0.9986_{0.0004}$ | $0.9951_{0.0005}$ | $0.9909_{0.0003}$ | $8.6924_{0.0750}$ | 20.46 |
| FT | $\mathbf{0.9978}_{0.0002}$ | $\mathbf{0.9622}_{0.0020}$ | $0.0995_{0.0179}$ | 0.8627 | $\mathbf{0.9945}_{0.0005}$ | $\mathbf{0.9985}_{0.0006}$ | $\mathbf{0.9946}_{0.0007}$ | $0.9533_{0.0038}$ | $2.3220_{0.0216}$ | 3.17 |
| GA | $0.9444_{0.0055}$ | $0.9144_{0.0047}$ | $\mathbf{0.0000}_{0.0000}$ | 0.9144 | $0.9969_{0.0004}$ | $0.9998_{0.0000}$ | $0.9970_{0.0002}$ | $0.9849_{0.0012}$ | $16.4834_{0.2720}$ | 0.98 |
| IU | $0.8044_{0.1177}$ | $0.8061_{0.0978}$ | $\mathbf{0.0000}_{0.0000}$ | 0.8061 | $0.9998_{0.0003}$ | $1.0000_{0.0000}$ | $0.9998_{0.0003}$ | $\mathbf{0.9936}_{0.0056}$ | $15.0697_{1.7117}$ | 0.41 |
| $\ell_1$-sparse | $0.9799_{0.0004}$ | $0.9580_{0.0017}$ | $\mathbf{0.0000}_{0.0000}$ | 0.9580 | $0.9921_{0.0013}$ | $0.9966_{0.0003}$ | $0.9921_{0.0012}$ | $0.9818_{0.0030}$ | $4.5139_{0.2512}$ | 3.73 |
| RL | $0.9959_{0.0001}$ | $0.9612_{0.0013}$ | $\mathbf{0.0000}_{0.0000}$ | 0.9612 | $0.9912_{0.0013}$ | $0.9971_{0.0016}$ | $0.9913_{0.0012}$ | $0.9813_{0.0016}$ | $\mathbf{5.5978}_{0.0357}$ | 2.60 |
| BE | $0.9880_{0.0008}$ | $0.9546_{0.0012}$ | $0.2812_{0.0061}$ | 0.6734 | $0.9976_{0.0005}$ | $0.9995_{0.0002}$ | $0.9976_{0.0006}$ | $0.9106_{0.0050}$ | $4.3816_{0.0659}$ | 0.46 |
| BS | $0.9864_{0.0010}$ | $0.9537_{0.0010}$ | $0.3109_{0.0052}$ | 0.6428 | $0.9975_{0.0003}$ | $0.9995_{0.0002}$ | $0.9976_{0.0003}$ | $0.9072_{0.0031}$ | $4.4290_{0.1169}$ | 0.82 |
| SCRUB | $0.9916_{0.0007}$ | $0.9616_{0.0014}$ | $\mathbf{0.0000}_{0.0000}$ | **0.9616** | $0.9999_{0.0001}$ | $1.0000_{0.0001}$ | $0.9999_{0.0001}$ | $0.9989_{0.0008}$ | $24.4590_{2.2852}$ | 3.91 |
| SG | $0.9716_{0.0007}$ | $0.9601_{0.0014}$ | $\mathbf{0.0000}_{0.0000}$ | 0.9601 | $0.9928_{0.0001}$ | $0.9954_{0.0001}$ | $0.9929_{0.0001}$ | $0.9907_{0.0008}$ | $5.0148_{2.2852}$ | 5.92 |

KKT conditions. The Lagrangian of the above is as follows where $\alpha_i \geq 0$ are the Lagrantian multipliers:

$$L(\theta_a, b, \alpha_i) = \frac{1}{2}\|\theta_a\|^2 - \sum_{i=1}^{q} \alpha_i \left(y_i \cdot (\theta_a^\top x_i + b) - 1\right). \tag{8}$$

Following sandard procedures [32], the KKT conditions are as folllows

$$f(\tilde{D}_{\theta_u}, \theta_a) = \begin{cases} \theta_a - \sum_{i=1}^{q} \alpha_i y_i x_i = 0 \\ -\sum_{i=1}^{q} \alpha_i y_i = 0 \\ y_i \cdot (\theta_a^\top x_i + b) \geq 1 \\ \alpha_i \geq 0, \forall i \\ \alpha_i (y_i (\theta_a^\top x_i + b) - 1) = 0, \forall i \end{cases}, \tag{9}$$

which implicitly define a function between $\theta_a$ and the data $\tilde{D}_{\theta_u} = \{(x_i, y_i)\}_{i=1}^{q}$. In practice, we describe the optimization problem (7) using `cvxpy` [34]. Then, we employ an off-the-shelf package called `cvxpylayers` [7] to automatically derive the KKT conditions and compute the gradient $\partial\theta_a / \partial\tilde{D}_{\theta_u}$.