Hard to Forget: Poisoning Attacks on Certified Machine Unlearning

Neil G. Marchant,[1] **Benjamin I. P. Rubinstein,**[1] **Scott Alfeld**[2]

[1] School of Computing and Information Systems, University of Melbourne, Parkville, Australia
[2] Department of Computer Science, Amherst College, Amherst, USA
nmarchant@unimelb.edu.au, brubinstein@unimelb.edu.au, salfeld@amherst.edu

Abstract

The right to erasure requires removal of a user's information from data held by organizations, with rigorous interpretations extending to downstream products such as learned models. Retraining from scratch with the particular user's data omitted fully removes its influence on the resulting model, but comes with a high computational cost. Machine "unlearning" mitigates the cost incurred by full retraining: instead, models are updated incrementally, possibly only requiring retraining when approximation errors accumulate. Rapid progress has been made towards privacy guarantees on the indistinguishability of unlearned and retrained models, but current formalisms do not place practical bounds on computation. In this paper we demonstrate how an attacker can exploit this oversight, highlighting a novel attack surface introduced by machine unlearning. We consider an attacker aiming to increase the computational cost of data removal. We derive and empirically investigate a poisoning attack on certified machine unlearning where strategically designed training data triggers complete retraining when removed.

1 Introduction

Much of modern machine learning has been advanced by the availability of large volumes of data. However, organizations that collect user data must comply with data protection regulations and often prioritize privacy to mitigate risk of civil litigation and reputational damage. Prominent regulations—i.e., GDPR (Council of the EU 2016) and CCPA (California State Legislature 2018)—encode a *right to erasure*—organizations must "erase personal data without undue delay" when requested. When tested, such regulations have been found to extend to products derived from personal data, including trained models (FTC 2021). Veale, Binns, and Edwards (2018) argue that models could be legally classified as personal data, which agrees with the draft guidance from the UK ICO (2020) that "erasure of the data... may not be possible without retraining the model... or deleting the model altogether". Such concerns are justified by a legitimate privacy risk: training data can be practically reconstructed from models (Fredrikson, Jha, and Ristenpart 2015; Shokri et al. 2017; Veale, Binns, and Edwards 2018).

Naïvely erasing data from a model by full retraining can be computationally costly. This has motivated the recent study of *machine unlearning*: how to efficiently undo the impact of data on a model. Guo et al. (2020) develop an approximate update which is 3 orders of magnitude faster than retraining, or supporting up to 10,000 approximate updates before retraining is required, with only a small drop in accuracy. Ginart et al. (2019) find a $100\times$ improvement in deletion efficiency with comparable clustering quality. Theoretical investigations has led to guarantees on privacy (indistinguishability of unlearned models with retrained models), but have largely ignored computation cost.

In an adversarial setting where risks to privacy motivate unlearning, we advocate that computation should also be viewed under an adversarial lens. While the aforementioned empirical results paint an optimistic picture for machine unlearning, such work assumes passive users who are trusted: they work within provided APIs, but also don't attempt to harm system performance in any way. In this paper we consider an active adversary as is more typical in the adversarial learning field (Huang et al. 2011). Users in this work can issue strategically chosen data for learning, and then request their data be unlearned. We adopt a typical threat model where users may apply a bounded perturbation to their (natural) data. Boundedness is motivated by a desire to remain undetected. Under this threat model **we identify a new vulnerability: poisoning attacks that effectively slow down unlearning.**

We highlight several perspectives on our results. They call into question the extent to which unlearning improves performance over full retraining which achieves complete indistinguishability. Our work highlights the risk of deploying approximate unlearning algorithms with data-dependent running times. And more broadly we show that data poisoning can harm *computation* beyond only accuracy—analogous to conventional denial-of-service attacks.

Contributions. We identify a new vulnerability of machine learning systems. Our suite of experiments considers: the attack's effect in a wide range of parameter settings; both white-box and grey-box attacks; a range of perturbation geometries and bounds; trade-offs between optimality of the attacker's objective and time to compute the attack; and the feasibility of running the attack long term.

2 Preliminaries

This section sets the scene for our attack. We introduce the target of our attack in Sec. 2.1—learning systems that support data erasure requests via machine unlearning. We then review a certified machine unlearning method that we use to demonstrate our attack. Sec. 2.3 closes with a discussion of the vulnerability and our assumed threat model. Table 1 summarizes notation used throughout the paper.

2.1 Machine learning on user data

Consider an organization that trains machine learning models using data from consenting users. In line with privacy regulations, the organization allows users to revoke consent at any time and submit an erasure request for some (or all) of their data. When fulfilling an erasure request, the organization takes a cautious approach to user privacy—erasing the user's raw data and removing the data's influence on trained models. We assume erasing raw data (e.g. from a database) is straightforward and focus here on the task of removing data from a trained model.

Formally, let $\mathcal{Z}^*$ be the space of training datasets consisting of samples from an example space $\mathcal{Z}$. Given a training dataset $D \in \mathcal{Z}^*$, the organization deploys a *learning algorithm* that returns a model $A(D)$ in hypothesis space $\mathcal{H}$.[1] To support erasure requests, the organization also deploys an *unlearning algorithm* M which, given a training dataset $D \in \mathcal{Z}^*$, a subset $D' \subset D$ to erase and a model $h \sim A(D)$, returns a sanitized model $M(D, D', h) \in \mathcal{H}$ that is approximately (to be made mathematically precise below) $A(D \setminus D')$. We allow A and M to be randomized algorithms—each can be interpreted as specifying a distribution over $\mathcal{H}$ conditioned on their inputs.

To ensure no trace of the erased data is left behind, we assume A and M satisfy a rigorous definition of unlearning. Specifically, we assume the distribution of unlearned models $M(D, D', A(D))$ is *statistically indistinguishable* from the distribution of models retrained on the remaining data $A(D \setminus D')$. This is known as *approximate unlearning* and there are variations depending on how statistical indistinguishability is defined. For concreteness, we adopt the unlearning definition and algorithms of Guo et al. (2020), which we review in the next section.

2.2 Certified removal

This section reviews learning/unlearning algorithms for linear models proposed by Guo et al. (2020). The algorithms satisfy a definition of approximate unlearning called *certified removal*, where indistinguishability is defined in a similar manner as (ϵ, δ)-differential privacy (Dwork et al. 2006).

Definition 1. *Given $\epsilon, \delta > 0$ a removal mechanism M performs (ϵ, δ)-certified removal for learning algorithm A if $\forall \mathcal{T} \subseteq \mathcal{H}, D \in \mathcal{Z}^*, \mathbf{z} \in D$:*

$$P(M(D, \{\mathbf{z}\}, A(D)) \in \mathcal{T}) \leq e^{\epsilon} P(A(D \setminus \{\mathbf{z}\}) \in \mathcal{T}) + \delta$$

[1] In practice, a "model" may contain parameters used at inference time, as well as metadata used to speed up unlearning. For notational brevity, we absorb the metadata into the definition of $\mathcal{H}$.

Table 1: Summary of notation.

Notation	Explanation
A	Learning algorithm
M	Unlearning algorithm
D, D_{psn}, D_{cln}	Dataset (generic, poisoned, clean)
$\mathbf{X}$, $\mathbf{X}_{\text{psn}}$, $\mathbf{X}_{\text{cln}}$	Feature matrix (generic, poisoned, clean)
$\mathcal{H}$	Hypothesis space for model
$\boldsymbol{\theta}$	Model parameters
R, $R_{\mathbf{b}}$	Learning objective (unperturbed, perturbed)
$\mathbf{b}$	Objective perturbation coefficients
σ	Standard deviation of $\mathbf{b}$
λ	Regularization strength
ℓ	Loss function
γ	Lipschitz constant of ℓ'
ϵ, δ	Indistinguishability parameters
β	Bound on gradient norm of $R_{\mathbf{b}}$
C	Attacker's estimate of unlearning cost
$\{g_j\}$	Attacker's inequality constraints

and

$$P(A(D \setminus \{\mathbf{z}\}) \in \mathcal{T}) \leq e^{\epsilon} P(M(D, \{\mathbf{z}\}, A(D)) \in \mathcal{T}) + \delta.$$

The learning/unlearning algorithms we consider support linear models that minimize a regularized empirical risk of the form:

$$R(\boldsymbol{\theta}; D) = \sum_{(\mathbf{x}, y) \in D} \left\{ \ell(\boldsymbol{\theta}^\top \mathbf{x}, y) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 \right\} \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a vector of learned weights, $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a convex loss that is differentiable everywhere, and $\lambda > 0$ is a regularization hyperparameter. Note that we are assuming the instance space $\mathcal{Z}$ is the product of a feature space $\mathcal{X} \subseteq \mathbb{R}^d$ and a label space $\mathcal{Y} \subseteq \mathbb{R}$. Thus supervised regression and classification are supported.

Learning algorithm. Since unlearning is inexact (see below), random noise is injected during learning to ensure indistinguishability. Concretely, the original minimization objective $R(\boldsymbol{\theta}; D)$ is replaced by a perturbed objective

$$R_{\mathbf{b}}(\boldsymbol{\theta}; D) = R(\boldsymbol{\theta}; D) + \mathbf{b}^\top \boldsymbol{\theta}, \tag{2}$$

where $\mathbf{b} \in \mathbb{R}^d$ is a vector of coefficients drawn i.i.d. from a standard normal distribution with standard deviation σ. The resulting perturbed learning problem can be solved using standard convex optimization methods. This procedure is summarized in pseudocode below.

Algorithm 1: Learning algorithm for certified removal

procedure $A(D)$
  $\mathbf{b} \sim \text{Normal}(\mathbf{0}, \sigma \cdot \mathbb{I}_d)$
  $\boldsymbol{\theta}^\star \leftarrow \arg\min_{\boldsymbol{\theta}} R_{\mathbf{b}}(\boldsymbol{\theta}; D)$
  $\beta \leftarrow 0$ ▷ initial gradient residual norm is zero
  return $\boldsymbol{\theta}^\star, \beta$
end procedure

Unlearning algorithm. Approximate unlearning is done by perturbing the model parameters $\boldsymbol{\theta}$ towards the optimizer of the new objective $R_{\mathbf{b}}(\boldsymbol{\theta}; D \setminus D')$. The perturbation is computed as

$$\Delta\boldsymbol{\theta} = -\mathcal{I}(D'; D, \boldsymbol{\theta})$$

where D is the initial training data, $D' \subset D$ is the data to erase, $\boldsymbol{\theta}$ are the initial model parameters and

$$\mathcal{I}(D'; D, \boldsymbol{\theta}) = -\left[\mathrm{H}_{\boldsymbol{\theta}} R_{\mathbf{b}}(\boldsymbol{\theta}; D)\right]^{-1} \nabla_{\boldsymbol{\theta}} R(\boldsymbol{\theta}; D') \quad (3)$$

is the *influence function* (Cook and Weisberg 1980; Koh and Liang 2017). The influence function approximates the change in $\boldsymbol{\theta}$ when D is augmented by D'.[2] The first factor in (3) is the inverse Hessian of the objective evaluated on D and the second factor is the gradient of the objective evaluated on D'.

Since the update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$ is approximate, it is not guaranteed to minimize the new objective exactly. This becomes a problem for indistinguishability if the error in $\boldsymbol{\theta}$ is so large that it is not masked by the noise added during learning. To ensure indistinguishability, the unlearning algorithm maintains an upper bound on the gradient residual norm (GRN) $\|\nabla R_{\mathbf{b}}(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}; D \setminus D')\|_2$ and resorts to retraining from scratch if a trigger value is exceeded. Given an initial upper bound on the GRN of β (before D' is removed and $\boldsymbol{\theta}$ is updated), the upper bound is updated as

$$\beta \leftarrow \beta + \gamma \|\mathbf{X}\|_2 \|\Delta\boldsymbol{\theta}\|_2 \|\mathbf{X}\Delta\boldsymbol{\theta}\|_2 \quad (4)$$

where γ is the Lipschitz constant of ℓ'', $\mathbf{X}$ is the feature matrix associated with $D \setminus D'$ and the feature vectors are assumed to satisfy $\|\mathbf{x}\|_2 \leq 1$ for all $(\mathbf{x}, y) \in D$. Whenever β exceeds the trigger value

$$\beta_{\text{trigger}} = \sigma\epsilon / \sqrt{2 \log 3/2\delta} \quad (5)$$

the algorithm resorts to retraining from scratch to ensure certified removal, as proved by Guo et al. (2020).

The entire procedure is summarized in pseudocode below.

Algorithm 2: Unlearning algorithm for certified removal

```
1:  procedure M(D, D', h)
2:      θ, β ← h
3:      Δθ ← −I(D'; D, θ)
4:      D ← D \ D'
5:      X ← feature matrix of D
6:      β ← β + γ‖X‖₂‖Δθ‖₂‖XΔθ‖₂
7:      if β > β_trigger then
8:          θ, β ← A(D)          ▷ Retrain from scratch (slow)
9:      else
10:         θ ← θ + Δθ           ▷ Approximate update (fast)
11:     end if
12:     return θ, β
13: end procedure
```

[2] The influence function is derived assuming $\boldsymbol{\theta}$ is the optimizer $\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} R_{\mathbf{b}}(\boldsymbol{\theta}; D)$. We include $\boldsymbol{\theta}$ as an argument since deviations from $\boldsymbol{\theta}^\star$ may occur during unlearning. Note that the *negative* influence function captures the scenario where D' is *subtracted* from D.

2.3 Vulnerability and threat model

One critique of the unlearning algorithm introduced in the previous section is that it does not guarantee a reduction in computation cost compared to full retraining. In the worst case, the error of the approximate unlearning update exceeds the allowed threshold ($\beta > \beta_{\text{trigger}}$) and the algorithm resorts to full retraining (line 8 in Alg. 2). We exploit this vulnerability in this paper to design an attack that triggers retraining more frequently, thereby diminishing (or even nullifying) the promised efficiency gains of unlearning.

Capabilities of the attacker. We assume the attacker controls one or more users, who are able to modify (within limits) their training data and initiate erasure requests. In practical settings, the attacker could increase the number of users under their control by creating fake user accounts. Following standard threat models in the data poisoning literature (Biggio, Nelson, and Laskov 2012; Muñoz González et al. 2017; Shafahi et al. 2018; Shen, Zhu, and Ma 2019; Huang et al. 2020) we consider attackers with *white-box* and *grey-box* access. In our white-box setting, the attacker can access training data of benign users, the architecture of the deployed model and the model state (excluding the coefficients $\mathbf{b}$ of the random perturbation term). While this is a perhaps overly pessimistic view, it allows for a worst-case evaluation of the attack. In our grey-box setting, the attacker still has knowledge of the model architecture, but can no longer access training data of benign users nor the model state. In place of real training data, the grey-box attacker acquires surrogate data from the same or a similar distribution.

Capabilities of the defender. For the majority of this paper, we assume the organization (the defender) is unaware that unlearning is vulnerable to slow-down attacks and does not mount any specific defenses. However, we do assume the organization conducts basic data validity checks (e.g. ensuring image pixel values are within valid ranges) and occasional manual auditing of data. To evade detection during manual auditing, we assume the attacker makes (small) bounded modifications to clean data which are unlikely to arouse suspicion.

3 Slow-down attack

In this section, we present an attack on unlearning algorithms with data-dependent running times. We begin by formulating our attack as a generalized data poisoning problem in Sec. 3.1. Then in Sec. 3.2, we describe a practical solution based on projected gradient descent under some simplifying assumptions. Finally, in Sec. 3.3 we propose concrete objectives for the attacker and suggest further relaxations to speed-up the attack.

3.1 Formulation as a data poisoning problem

Let D be data used by the organization (the defender) to train (with learning algorithm A) an initial model $\hat{h} = A(D)$. Suppose the attacker is able to poison a subset D_{psn} of instances in D, and denote the remaining clean instances by $D_{\text{cln}} = D \setminus D_{\text{psn}}$. We introduce a function $C : \mathcal{H}, \mathcal{Z}^* \to \mathbb{R}$ which measures the *computational cost* $C(\hat{h}, D')$ of erasing

a subset of data $D' \subset D$ from trained model $\hat{h} \in \mathcal{H}$ using unlearning algorithm M.

The attacker's aim is to craft D_{psn} to maximize $C(\hat{h}, D_{\text{psn}})$. Assuming the attacker enforces constraints on D_{psn} to pass the defender's checks/auditing, we express the attacker's problem formally as follows:

$$\max_{D_{\text{psn}}} \quad C(\hat{h}, D_{\text{psn}}) \tag{6a}$$

$$\text{subject to} \quad \hat{h} = \mathbb{E}[A(D_{\text{cln}} \cup D_{\text{psn}})] \tag{6b}$$

$$g_j(D_{\text{psn}}) \leq 0 \quad \forall j \in \{1, \ldots, J\} \tag{6c}$$

This is a *bilevel optimization problem* (Mei and Zhu 2015) if the learning algorithm A solves a deterministic optimization problem. Although solving (6a)–(6c) exactly is infeasible in general, we can find locally-optimal solutions using gradient-based methods, which we discuss next.

3.2 PGD-based crafting strategy

We now outline an approximate solution to (6a)–(6c) based on projected gradient descent (PGD). Our solution makes the following assumptions:

1. The learning/unlearning algorithms adopted by the organization (defender) are those presented in Sec. 2.2 for regularized linear models (Algs. 1 and 2).
2. The attacker crafts the poisoned data D_{psn} using clean reference data D_{ref} as a basis. In addition, we assume the attacker only modifies features in the reference data, leaving labels unchanged. Mathematically, if we write $D_{\text{psn}} = (\mathbf{X}_{\text{psn}}, \mathbf{y}_{\text{psn}})$ and $D_{\text{ref}} = (\mathbf{X}_{\text{ref}}, \mathbf{y}_{\text{ref}})$ where $\mathbf{X}_{\text{psn}}, \mathbf{X}_{\text{ref}} \in \mathbb{R}^{m \times d}$ are feature matrices and $\mathbf{y}_{\text{psn}}, \mathbf{y}_{\text{ref}} \in \mathbb{R}^m$ are label vectors, then the attacker optimizes $\mathbf{X}_{\text{psn}}$ and fixes $\mathbf{y}_{\text{psn}} = \mathbf{y}_{\text{ref}}$.
3. The inequality constraints in (6c) are of the form
$$g_j(D_{\text{psn}}) = \sup_{\mathbf{x} \in \text{rows}(\mathbf{X}_{\text{psn}} - \mathbf{V}_j)} \|\mathbf{x}\|_{p_j} - r_j \tag{7}$$
where $\mathbf{x} \in \mathbb{R}^{1 \times d}$, $\mathbf{V}_j \in \mathbb{R}^{m \times d}$, $r_j > 0$ and $p_j \in \mathbb{Z}_{>0}$. This ensures each row of $\mathbf{X}_{\text{psn}}$ is confined to an ℓ_{p_j}-ball of radius r_j centered on the corresponding row in $\mathbf{V}_j$.

In addition to these assumptions, we approximate the expectation in (6b) since it cannot be computed exactly. Applying a zero-th order expansion of the $\arg\max$ function, we make the replacement:

$$\mathbb{E}[A(D)] \to \arg\max_{\boldsymbol{\theta}} \mathbb{E}[R_{\mathbf{b}}(\boldsymbol{\theta}; D)] = \arg\max_{\boldsymbol{\theta}} R(\boldsymbol{\theta}; D).$$

The equality follows from the definition of the perturbed objective in (2) and the fact that the expectation of the perturbation is zero.

Putting everything together, we recast (6a)–(6c) as a single-level constrained optimization problem:

$$\min_{\mathbf{X}_{\text{psn}} \in \mathbb{R}^{m \times d}} \quad f(\mathbf{X}_{\text{psn}}) \tag{8a}$$

$$g_j(D_{\text{psn}}) \leq 0 \quad \forall j \in \{1, \ldots, J\} \tag{8b}$$

where we define

$$f(\mathbf{X}_{\text{psn}}) = -C(\hat{\boldsymbol{\theta}}(D), D_{\text{psn}}) \text{ and } \hat{\boldsymbol{\theta}}(D) = \arg\min_{\boldsymbol{\theta}} R(\boldsymbol{\theta}; D). \tag{9}$$

This problem can be solved using projected gradient descent (PGD) as detailed in Appendix A.

3.3 Measuring the computational cost

Recall that the attacker's goal is to maximize the computational cost $C(\hat{\boldsymbol{\theta}}, D_{\text{psn}})$ of erasing poisoned data D_{psn} from the model $\hat{\boldsymbol{\theta}}$ trained on $D = D_{\text{psn}} \cup D_{\text{cln}}$. In this section, we discuss practical choices for C assuming the organization implements the learning/unlearning algorithms in Sec. 2.2.

We begin with the observation that the computational cost of erasing data in a call to M (Alg. 2) varies considerably depending on whether full retraining is triggered or not. Guo et al. (2020) report that full retraining is three orders of magnitude slower than an approximate update, making it the dominant contribution when averaged over a series of calls to M. Complete retraining is triggered when the gradient residual norm bound β exceeds a threshold, and therefore we model the attacker as aiming to maximize β.

Using the expression for β in (4), we therefore set

$$C(\hat{\boldsymbol{\theta}}, D_{\text{psn}}) = \|\mathbf{X}\|_2 \|\Delta\boldsymbol{\theta}\|_2 \|\mathbf{X}\Delta\boldsymbol{\theta}\|_2 \tag{10}$$

where $\mathbf{X}$ is the feature matrix associated with $D \setminus D_{\text{psn}}$ and $\Delta\boldsymbol{\theta} = -\mathcal{I}(D_{\text{psn}}; D, \hat{\boldsymbol{\theta}})$.[3] We note that this cost function assumes D_{psn} is erased in a *single call* to M. However our experiments in Sec. 4 indicate that (10) is effective even if D_{psn} is erased in a *sequence of calls* to M.

Faster cost surrogates. While (10) governs whether retraining is triggered, it may be expensive for the attacker to evaluate, particularly due to the presence of $\mathbf{X}$ which requires operations that scale linearly in $n - m$ (the size of the remaining training data). We therefore consider simpler surrogates for the computational cost, based on alternate upper bounds for the gradient residual norm (GRN). We study the effect of these surrogates empirically in Sec. 4.

The first surrogate is based on a data-independent upper bound of the GRN from (Guo et al. 2020, Theorem 1):

$$C(\hat{\boldsymbol{\theta}}, D_{\text{psn}}) = \|\mathcal{I}(D_{\text{psn}}; D, \hat{\boldsymbol{\theta}})\|_2. \tag{11}$$

Since this is a looser bound on the GRN than (10), we expect it will produce less effective attacks.

The second surrogate upper bounds (11). Observing that

$$\begin{aligned}\|\mathcal{I}(D_{\text{psn}}; D, \hat{\boldsymbol{\theta}})\|_2 &\leq \|\mathrm{H}_{\boldsymbol{\theta}} R_{\mathbf{b}}(\hat{\boldsymbol{\theta}}; D)\|_2 \cdot \|\nabla_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}; D_{\text{psn}})\|_2 \\ &\leq \frac{1}{\lambda(|D| - 1)} \cdot \|\nabla_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}; D_{\text{psn}})\|_2\end{aligned}$$

we propose

$$C(\hat{\boldsymbol{\theta}}, D_{\text{psn}}) = \|\nabla_{\boldsymbol{\theta}} R(\hat{\boldsymbol{\theta}}; D_{\text{psn}})\|_2. \tag{12}$$

This doesn't depend on the Hessian (given $\hat{\boldsymbol{\theta}}$) and is therefore significantly cheaper for the attacker to compute.

Ignoring model dependence. Another way of reducing the attacker's computational effort is to assume the model trained on $D = D_{\text{psn}} \cup D_{\text{cln}}$ is approximately independent of D_{psn}. Specifically, we make the replacement $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} R(\boldsymbol{\theta}; D_{\text{cln}})$ in (9) so that $\hat{\boldsymbol{\theta}}$ is constant with respect to D_{psn}. This means evaluating the objective (or its

[3] For readability, we omit factors that are independent of D when defining cost functions.

gradient) no longer requires retraining and simplifies gradient computations (see Appendix A). Although this approximation is based on a dubious assumption—the model must be somewhat sensitive to D_{psn}—we find it performs well empirically (see Table 4).

4 Experiments

We investigate the impact of our attack on the computational cost of unlearning in a variety of simulated settings. We study the effect of the organization's parameters—the regularization strength λ and the magnitude of the objective perturbation σ—as well as the attacker's parameters—the cost function C, and the type of norm and radius r used to bound the perturbations. We further evaluate the effectiveness of our attack over time, as additional poisoned erasure requests are processed. Finally, we investigate a transfer setting where the attacker uses a surrogate model (trained from surrogate data) in lieu of the defender's true model. Code is published at https://www.github.com/ngmarchant/unlearning-attack.

4.1 Datasets and setup

We consider MNIST (LeCun et al. 1998) and Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), both of which contain $d = 28 \times 28$ single-channel images from ten classes. We also generate a smaller binary classification dataset from MNIST we call Binary-MNIST which contains classes 3 and 8. Beyond the image domain, we consider human activity recognition (HAR) data (Anguita et al. 2013). It contains windows of processed sensor signals ($d = 561$) corresponding to 6 activity classes.

All datasets come with predefined train/test splits. The way the splits are used varies for each attack setting (defined in Sec. 2.3). In the white-box setting, the attacker accesses the train split and *replaces* instances under their control[4] with poisoned instances. The test split is used to estimate model accuracy. In the grey-box setting, the train split is inaccessible to the attacker. Instead the attacker uses the test split as surrogate data to craft poisoned instances, which are then *added* to the initial train split. We do not report model accuracy for this setting. In both settings, the resulting training data D is used by the organization to learn a logistic regression model, after which the attacker submits erasure requests for the poisoned instances $D_{\text{psn}} \subset D$ one-by-one.

The main quantity of interest is the *retrain interval*—the number of erasure requests handled via fast approximate updates before full retraining is triggered. A more effective attack achieves a smaller retrain interval. Further details about the experiments, including hardware, parameter settings poisoning ratios, and the number of trials in each experiment are provided in Appendix B.

4.2 Results

Varying model sensitivity. Table 2 shows that our attack can reduce the retrain interval by 70–100% over a range of settings for λ and σ. The defender can adjust the model's sensitivity to training data by varying the regularization strength λ and the magnitude of the objective perturbation σ. Our attack is most effective in an absolute sense for smaller values of λ and σ, where the retrain interval is reduced to zero. This means retraining is triggered immediately upon processing a poisoned erasure request. Table 2 also illustrates a trade-off between model accuracy and unlearning efficiency: larger values of λ and σ generally allow for more efficient unlearning (larger retrain interval) at the expense of model accuracy.

Table 2: Attack effectiveness on Binary-MNIST as a function of the regularization strength λ and magnitude of the objective perturbation σ. The accuracy is reported for the initial model, prior to processing erasure requests.

σ	λ	Accuracy		Retrain interval		
		Benign	Attack	Benign	Attack	% ↓
1	10^{-5}	0.962	0.962	3.58	0.07	98.0
	10^{-4}	0.968	0.968	5.80	0	100
	10^{-3}	0.958	0.959	16.4	0.22	98.7
	10^{-2}	0.926	0.926	188	48.4	74.3
10	10^{-5}	0.919	0.919	0	0	–
	10^{-4}	0.932	0.931	9.32	0.27	97.1
	10^{-3}	0.954	0.955	132	8.15	93.8
	10^{-2}	0.926	0.920	1640	512	68.8

Varying the peturbation constraint. Table 3 shows how the effectiveness of our attack varies depending on the choice of ℓ_p-norm and radius r. As expected, the attack is most effective when there is no perturbation constraint ($p = \infty$ and $r = 1$ or $p = 1$ and $r = d$), however the poisoned images are no longer recognizable as digits. The ℓ_1-norm constraint appears more effective, as it can better exploit sensitivities in the weights of individual pixels.

Varying the cost function. We study whether the attack remains effective when: (i) faster surrogate cost functions are used and (ii) the model dependence on the poisoned data D_{psn} is ignored (see Sec. 3.3). Table 4 shows the attack can be executed with a similar effectiveness (retrain interval) using the influence norm cost function (11) in place of the GRNB cost function (10), while reducing the attacker's computation. The gradient norm cost function (12) is the least computationally demanding for the attacker, but less effective. We see that it is reasonable to ignore the model dependence on D_{psn}, as there is no marked difference in effectiveness and a significant reduction in the attacker's computation. We expect this is due to the relatively small size of D_{psn} (reflecting realistic capabilities of an attacker) which limits the impact of poisoning on the model.

Long-term effectiveness. We are interested in whether the attack effectiveness varies over time—as poisoned erasure requests are processed. Figure 1 shows a moderate increase in effectiveness, indicated by the increasing rate of retraining. The is likely due to the fact that the defender's model is initially far from the model used by the attacker

[4]The instances under the attackers control are randomly-selected in each trial.

Table 3: Attack effectiveness on Binary-MNIST as a function of the perturbation constraint.

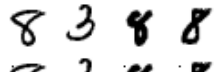
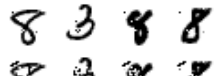
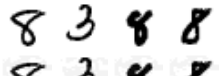
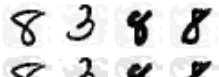
Constraint		Retrain interval	Poisoned examples
Norm	r		
ℓ_1	0	131	
	$d/200$	72.3	
	$d/20$	8.15	
	$d/2$	3.42	
	d	3.54	
ℓ_∞	0.000	131	
	0.050	82.2	
	0.100	48.9	
	0.500	5.63	
	1.000	3.54	

Table 4: Attack effectiveness and computation time for different choices of the cost function.

Cost function	Ignore model dep.	Retrain interval	Attack time (s)
GRNB (10)	No	6.96	39.2
	Yes	7.08	24.4
Influence norm (11)	No	7.98	29.0
	Yes	8.15	8.72
Gradient norm (12)	No	16.34	23.1
	Yes	19.21	3.54

(the former is trained on D, the latter on D_{cln}). However as poisoned instances are erased, the defender's model approaches the attacker's model, and the poisoned instances become more effective.

Transferability. We investigate whether the attack transfers if *surrogate data* is used to craft the attack in place of the defender's training data. This corresponds to the grey-box setting described in Sec. 2.3. Table 5 shows that the attack transfers well—there is very little difference in the retrain interval in the grey- versus white-box settings.

Varying datasets. Most of the results reported above are for Binary-MNIST. Table 6 reports results for other datasets described in Sec. 4.1. While the results are qualitatively similar across datasets, we note some variance in attack effectiveness—depending on the dataset we observe a reduction in the retrain interval of 67–94%. The size of each dataset is likely a factor, since a model trained on more data is less sensitive to individual instances and more difficult

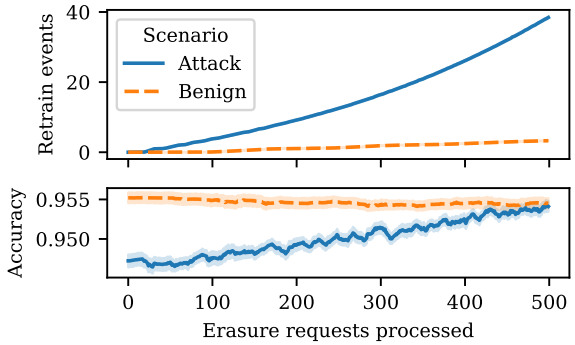


Figure 1: Number of times retraining is triggered as a function of the number of poisoned erasure requests processed. After processing 500 requests, all poisoned instances are erased and only clean data remains.

Table 5: Effectiveness of the attack in white-box versus grey-box settings.

Dataset	Retrain interval	
	Surrogate data (Grey-box)	Training data (White-box)
MNIST	19.0	18.5
Binary-MNIST	7.88	8.15

to attack. We also expect the ℓ_1-bounded perturbations are likely to have a different impact depending on the characteristics of the data. For instance, Fashion-MNIST has a wider variation of intensities across all pixels compared to MNIST, which may make ℓ_1-bounded perturbations less effective.

5 Related Work

Machine unlearning. Recent interest in efficient data erasure for learned models was spawned by Cao and Yang (2015), who were motivated by privacy and security applications. They coined the term "machine unlearning" and proposed a strict definition—requiring that an unlearning algorithm return a model *identical* to one retrained from scratch on the remaining data. Efficient unlearning algorithms are known for some canonical models under this definition, including linear regression (Chambers 1971; Hansen and Larsen 1996), naive Bayes and non-parametric models such as k-nearest neighbors (Schelter 2020). These algorithms are immune to our attack since they compute updates in closed form with data-independent running times.

A strict definition of unlearning has proven difficult to satisfy for general models. One solution has been to design new models or adapt existing ones with unlearning efficiency in mind. Examples of such models include ensembles of randomized decision trees (Schelter, Grafberger, and Dunning 2021), variants of k-means clustering (Ginart et al. 2019), and a modeling framework called SISA that takes advantage of data sharding and caching (Bourtoule et al. 2021).

Others have expanded the scope of unlearning by adopt-

Table 6: Effectiveness of the attack for various datasets.

Dataset	Accuracy		Retrain interval		
	Benign	Attack	Benign	Attack	% ↓
MNIST	0.867	0.867	71.6	18.5	74.1
Binary-MNIST	0.954	0.954	132	8.15	93.8
Fashion-MNIST	0.756	0.756	119	38.9	67.5
HAR	0.838	0.836	42.2	8.95	78.8

ing more relaxed (but still rigorous) definitions. Ginart et al. (2019) were first to introduce an approximate definition of unlearning—requiring that it be *statistically indistinguishable* from retraining. They identified a connection to differential privacy (Dwork et al. 2006) and suggested adversarial robustness as a direction for future work. Subsequent work (Guo et al. 2020; Neel, Roth, and Sharifi-Malvajerdi 2021; Sekhari et al. 2021) has adopted similar approximate definitions to provide certified unlearning guarantees for strongly-convex learning problems. Neel, Roth, and Sharifi-Malvajerdi (2021) and Sekhari et al. (2021) study asymptotic time complexity, however they do not test their methods empirically and they rely on data-independent bounds which are too loose in practice according to Guo et al. (2020). Since the approach by Guo et al. is empirically validated, we use it for our proof-of-concept exploit in this paper.

Certified unlearning for non-convex models such as deep neural networks (DNNs) remains elusive. Golatkar, Achille, and Soatto (2020a,b) have made some progress in this area based on a notion of approximate unlearning. However, their methods do not guarantee approximate unlearning is satisfied and they require further approximations to scale to DNNs. Since their methods run for a predetermined number of iterations, the running time is data-independent, making them immune to slow-down attacks. However, our slow-down attack may cause damage in a different way—preventing proper data erasure or leaving the unlearned model in an unpredictable state.

While our focus in this paper is on attacking unlearning efficiency, others have considered privacy vulnerabilities. Chen et al. (2020) demonstrated that a series of published unlearned models are vulnerable to differencing attacks if not appropriately treated. Gupta et al. (2021) considered an adaptive unlearning setting, where erasure requests may depend on previously published models. They also demonstrated an attack on the SISA algorithm (Bourtoule et al. 2021). Sommer et al. (2020) considered verification of unlearning—an important privacy feature for users who cannot trust organizations to erase their data.

Adversarial machine learning. Our work builds on standard methods for *data poisoning*—a class of attacks that manipulate learned models by modifying or augmenting training data (see survey Schwarzschild et al. 2021). Two commonly studied variants of data poisoning are *availability attacks*, which aim to degrade model test performance (e.g. Biggio, Nelson, and Laskov 2012; Muñoz González et al. 2017; Koh and Liang 2017; Shafahi et al. 2018; Zhu et al. 2019; Huang et al. 2020), and *backdoor attacks*, which aim to cause misclassification of test-time samples that contain a trigger (e.g. Chen et al. 2017; Dai, Chen, and Li 2019; Saha, Subramanya, and Pirsiavash 2020). Unlike these attacks, our attack does not aim to harm test performance—rather the aim is to cause trouble at the unlearning stage. This adds another layer of complexity, as we must account for learning and unlearning to craft an effective attack.

Most work on data poisoning assumes learning is done offline, as we do in this paper. However one could imagine an online scenario, where an organization continually updates a model as new data arrives and old data is erased. There has been some work on data poisoning for online learning (Wang and Chaudhuri 2018; Zhang, Zhu, and Lessard 2020), however existing work does not account for the ability of an attacker to request that their data be erased.

Our attack also borrows ideas from the adversarial examples literature. The ℓ_p-norm constraints that we impose on the attacker's perturbations are commonly used to generate adversarial examples (Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015). While ℓ_p-norm constraints are easy to work with, they have been criticized for failing to capture perceptual distortion (Sharif, Bauer, and Reiter 2018), which has lead to work on more effective perturbation constraints (Zhao, Liu, and Larson 2020; Ballet et al. 2019). These could be leveraged to enhance our attack.

While adversarial examples are predominantly used for evasion attacks, they have also been deployed for other purposes. Hong et al. (2021) showed that techniques for generating adversarial examples can be modified to slow down inference of multi-exit DNNs by a factor of 1.5–5×. This attack is qualitatively similar to ours, in that it targets computation cost, however the formulation is different. While our attack operates at training-time and unlearning-time, the attack by Hong et al. operates at test-time. Other novel applications of adversarial examples include data poisoning (Tao et al. 2021) and protecting data from being used to train models (Huang et al. 2021).

6 Discussion

We have demonstrated a broad range of settings where a poisoning attack can be successfully launched against state-of-the-art machine unlearning to significantly undo the advantages of unlearning over full retraining. Our results suggest that theory should incorporate a trade-off between indistinguishability, computational cost, and accuracy. Indeed in parallel work to our own, Neel, Roth, and Sharifi-Malvajerdi (2021) define "strong" unlearning algorithms as those for which accuracy bounds are independent of the number of unlearning requests t, with a computation cost that grows at most logarithmically in t. They explore trade-offs for an unlearning algorithm called perturbed gradient descent. However this approach hasn't been evaluated empirically to our knowledge, and a number of open problems remain.

Future work could apply or extend our methods to attack other unlearning algorithms, beyond those by Guo et al. (2020), which were the focus of this paper. We expect this would be easiest for algorithms based on a similar design—e.g. algorithms by Neel, Roth, and Sharifi-Malvajerdi (2021) and Sekhari et al. (2021) also perturb the original model to

find an approximate solution to a strongly-convex learning objective post-erasure. Other unlearning algorithms, such as Ginart et al.'s algorithm for k-means clustering, are also vulnerable to slow-down attacks, however crafting poisons to trigger retraining would be more difficult due to non-differentiability of the cost function. If certified unlearning algorithms are developed for deep models in the future, then assessing the impact of slow-down attacks in that setting—where data and models are typically at a much larger scale—would also be of immense interest.

Another direction suggested by our work is counter measures against slow-down attacks on unlearning. One might adopt an unlearning algorithm that never resorts to retraining from scratch (Golatkar, Achille, and Soatto 2020a). Such an algorithm might still require more computational effort to remove poisoned examples than benign examples, exposing a similar attack surface. It is also unclear whether such an algorithm could maintain a form of indistinguishability indefinitely.

One might seek to filter out poisoned examples so they never need to be unlearned. This might involve off-the-shelf anomaly detection methods, however poisoned examples are notoriously difficult to detect when they are crafted as small perturbations to clean examples. Metzen et al. (2017) for instance, have explored detection of adversarial perturbations. Alternatively, one could filter out examples with a large influence (which the attacker is trying to artificially inflate in our attack), however some clean examples have a naturally large influence.

Another approach would be a robust model that is less sensitive to individual examples. Directions for achieving this include: increasing the degree of regularization, introducing noise, adversarial training, and using more training data. However, like their analogues in existing adversarial learning research, these mitigations tend to harm accuracy.

Acknowledgment

This research was undertaken using the HPC-GPGPU Facility hosted at the University of Melbourne, established with the assistance of ARC LIEF Grant LE170100200. This research was also supported in part by the Defence Science and Technology Group's AMLC Next Generation Technologies Fund project.

References

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; and Reyes-Ortiz, J. L. 2013. A Public Domain Dataset for Human Activity Recognition using Smartphones. In *21st International European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*.

Armijo, L. 1966. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1): 1–3.

Balles, L.; Pedregosa, F.; and Roux, N. L. 2020. The Geometry of Sign Gradient Descent. arXiv:2002.08056.

Ballet, V.; Renard, X.; Aigrain, J.; Laugel, T.; Frossard, P.; and Detyniecki, M. 2019. Imperceptible Adversarial Attacks on Tabular Data. arXiv:1911.03274.

Bertsekas, D. P. 1999. *Nonlinear Programming*. Athena Scientific.

Biggio, B.; Nelson, B.; and Laskov, P. 2012. Poisoning Attacks against Support Vector Machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 1467–1474.

Birgin, E. G.; and Raydan, M. 2005. Robust Stopping Criteria for Dykstra's Algorithm. *SIAM Journal on Scientific Computing*, 26(4): 1405–1414.

Bourtoule, L.; Chandrasekaran, V.; Choquette-Choo, C. A.; Jia, H.; Travers, A.; Zhang, B.; Lie, D.; and Papernot, N. 2021. Machine Unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, 141–159.

Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M. J.; Leary, C.; Maclaurin, D.; Necula, G.; Paszke, A.; VanderPlas, J.; Wanderman-Milne, S.; and Zhang, Q. 2018. JAX: composable transformations of Python+NumPy programs.

California State Legislature. 2018. California Consumer Privacy Act of 2018. Cal. Civ. Code §1798.100. https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5.

Cao, Y.; and Yang, J. 2015. Towards Making Systems Forget with Machine Unlearning. In *2015 IEEE Symposium on Security and Privacy*, 463–480.

Chambers, J. M. 1971. Regression Updating. *Journal of the American Statistical Association*, 66(336): 744–748.

Chen, M.; Zhang, Z.; Wang, T.; Backes, M.; Humbert, M.; and Zhang, Y. 2020. When Machine Unlearning Jeopardizes Privacy. arXiv:2005.02205.

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. arXiv:1712.05526.

Cook, R. D.; and Weisberg, S. 1980. Characterizations of an Empirical Influence Function for Detecting Influential Cases in Regression. *Technometrics*, 22(4): 495–508.

Council of the EU. 2016. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L 119: 1–88.

Dai, J.; Chen, C.; and Li, Y. 2019. A Backdoor Attack Against LSTM-Based Text Classification Systems. *IEEE Access*, 7: 138872–138878.

Duchi, J.; Shalev-Shwartz, S.; Singer, Y.; and Chandra, T. 2008. Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 272–279.

Dwork, C.; Kenthapadi, K.; McSherry, F.; Mironov, I.; and Naor, M. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology - EUROCRYPT 2006*, 486–503.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 1322–1333.

FTC. 2021. California Company Settles FTC Allegations It Deceived Consumers about use of Facial Recognition in Photo Storage App. https://www.ftc.gov/news-events/press-releases/2021/01/california-company-settles-ftc-allegations-it-deceived-consumers.

Ginart, A.; Guan, M.; Valiant, G.; and Zou, J. Y. 2019. Making AI Forget You: Data Deletion in Machine Learning. In *Advances in Neural Information Processing Systems*, volume 32.

Golatkar, A.; Achille, A.; and Soatto, S. 2020a. Eternal Sunshine of the Spotless Net: Selective Forgetting in Deep Networks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9301–9309.

Golatkar, A.; Achille, A.; and Soatto, S. 2020b. Forgetting Outside the Box: Scrubbing Deep Networks of Information Accessible from Input-Output Observations. In *Computer Vision – ECCV 2020*, 383–398.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

Guo, C.; Goldstein, T.; Hannun, A.; and Van Der Maaten, L. 2020. Certified Data Removal from Machine Learning Models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, 3832–3842.

Gupta, V.; Jung, C.; Neel, S.; Roth, A.; Sharifi-Malvajerdi, S.; and Waites, C. 2021. Adaptive Machine Unlearning. arXiv:2106.04378.

Hansen, L. K.; and Larsen, J. 1996. Linear unlearning for cross-validation. *Advances in Computational Mathematics*, 5: 269–280.

Hong, S.; Kaya, Y.; Modoranu, I.-V.; and Dumitras, T. 2021. A Panda? No, It's a Sloth: Slowdown Attacks on Adaptive Multi-Exit Neural Network Inference. In *International Conference on Learning Representations*.

Huang, H.; Ma, X.; Erfani, S. M.; Bailey, J.; and Wang, Y. 2021. Unlearnable Examples: Making Personal Data Unexploitable. In *International Conference on Learning Representations*.

Huang, L.; Joseph, A. D.; Nelson, B.; Rubinstein, B. I.; and Tygar, J. D. 2011. Adversarial Machine Learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, 43—-58.

Huang, W. R.; Geiping, J.; Fowl, L.; Taylor, G.; and Goldstein, T. 2020. MetaPoison: Practical General-purpose Clean-label Data Poisoning. In *Advances in Neural Information Processing Systems*, volume 33, 12080–12091.

Koh, P. W.; and Liang, P. 2017. Understanding Black-Box Predictions via Influence Functions. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1885–1894.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Liu, D. C.; and Nocedal, J. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1): 503–528.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

Mei, S.; and Zhu, X. 2015. Using Machine Teaching to Identify Optimal Training-Set Attacks on Machine Learners. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2871–2877.

Metzen, J. H.; Genewein, T.; Fischer, V.; and Bischoff, B. 2017. On Detecting Adversarial Perturbations. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*.

Muñoz González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E. C.; and Roli, F. 2017. Towards Poisoning of Deep Learning Algorithms with Back-Gradient Optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 27–38.

Neel, S.; Roth, A.; and Sharifi-Malvajerdi, S. 2021. Descent-to-Delete: Gradient-Based Methods for Machine Unlearning. In *Proceedings of the 32nd International Conference on Algorithmic Learning Theory*, volume 132, 931–962.

Saha, A.; Subramanya, A.; and Pirsiavash, H. 2020. Hidden Trigger Backdoor Attacks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(7): 11957–11965.

Schelter, S. 2020. "Amnesia" – Machine Learning Models That Can Forget User Data Very Fast. In *Online Proceedings of the 10th Conference on Innovative Data Systems Research*.

Schelter, S.; Grafberger, S.; and Dunning, T. 2021. HedgeCut: Maintaining Randomised Trees for Low-Latency Machine Unlearning. In *Proceedings of the 2021 International Conference on Management of Data*, 1545–1557.

Schwarzschild, A.; Goldblum, M.; Gupta, A.; Dickerson, J. P.; and Goldstein, T. 2021. Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, 9389–9398.

Sekhari, A.; Acharya, J.; Kamath, G.; and Suresh, A. T. 2021. Remember What You Want to Forget: Algorithms for Machine Unlearning. arXiv:2103.03279.

Shafahi, A.; Huang, W. R.; Najibi, M.; Suciu, O.; Studer, C.; Dumitras, T.; and Goldstein, T. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6106–6116.

Sharif, M.; Bauer, L.; and Reiter, M. K. 2018. On the Suitability of Lp-Norms for Creating and Preventing Adversarial Examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Shen, J.; Zhu, X.; and Ma, D. 2019. TensorClog: An Imperceptible Poisoning Attack on Deep Neural Network Applications. *IEEE Access*, 7: 41498–41506.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18.

Sommer, D. M.; Song, L.; Wagh, S.; and Mittal, P. 2020. Towards Probabilistic Verification of Machine Unlearning. arXiv:2003.04247.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. arXiv:1312.6199.

Tao, L.; Feng, L.; Yi, J.; Huang, S.-J.; and Chen, S. 2021. Provable Defense Against Delusive Poisoning. arXiv:2102.04716.

UK ICO. 2020. Guidance on the AI auditing framework: Draft guidance for consultation. Version 1.0, https://ico.org.uk/media/about-the-ico/consultations/2617219/guidance-on-the-ai-auditing-framework-draft-for-consultation.pdf.

Veale, M.; Binns, R.; and Edwards, L. 2018. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133): 20180083.

Wang, Y.; and Chaudhuri, K. 2018. Data Poisoning Attacks against Online Learning. arXiv:1808.08994.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747.

Zhang, X.; Zhu, X.; and Lessard, L. 2020. Online Data Poisoning Attacks. In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120, 201–210.

Zhao, Z.; Liu, Z.; and Larson, M. 2020. Towards Large Yet Imperceptible Adversarial Image Perturbations With Perceptual Color Distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhu, C.; Huang, W. R.; Li, H.; Taylor, G.; Studer, C.; and Goldstein, T. 2019. Transferable Clean-Label Poisoning Attacks on Deep Neural Nets. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, 7614–7623.