Adaptive Machine Unlearning

Varun Gupta[1], Christopher Jung[1], Seth Neel[2],
Aaron Roth[1], Saeed Sharifi-Malvajerdi[1], and Chris Waites[3]

[1]University of Pennsylvania
[2]Harvard University
[3]Stanford University

June 9, 2021

Abstract

Data deletion algorithms aim to remove the influence of deleted data points from trained models at a cheaper computational cost than fully retraining those models. However, for sequences of deletions, most prior work in the non-convex setting gives valid guarantees only for sequences that are chosen *independently* of the models that are published. If people choose to delete their data as a function of the published models (because they don't like what the models reveal about them, for example), then the update sequence is *adaptive*. In this paper, we give a general reduction from deletion guarantees against adaptive sequences to deletion guarantees against non-adaptive sequences, using differential privacy and its connection to max information. Combined with ideas from prior work which give guarantees for non-adaptive deletion sequences, this leads to extremely flexible algorithms able to handle arbitrary model classes and training methodologies, giving strong provable deletion guarantees for adaptive deletion sequences. We show in theory how prior work for non-convex models fails against adaptive deletion sequences, and use this intuition to design a practical attack against the SISA algorithm of Bourtoule et al. [2021] on CIFAR-10, MNIST, Fashion-MNIST.

1 Introduction

Businesses like Facebook and Google depend on training sophisticated models on user data. Increasingly—in part because of regulations like the European Union's General Data Protection Act and the California Consumer Privacy Act—these organizations are receiving requests to delete the data of particular users. But what should that mean? It is straightforward to delete a customer's data from a database and stop using it to train *future* models. But what about models that have already been trained using an individual's data? These are not necessarily safe; it is known that individual training data can be exfiltrated from models trained in standard ways via *model inversion* attacks [Shokri et al., 2017, Veale et al., 2018, Fredrikson et al., 2015]. Regulators are still grappling with when a trained model should be considered to contain personal data of individuals in the training set and the potential legal implications. In 2020 draft guidance, the U.K.'s Information Commissioner's Office addressed how to comply with data deletion requests as they pertain to ML models:

> *If the request is for rectification or erasure of the data, this may not be possible without re-training the model...or deleting the model altogether* [ICO, 2020].

Fully retraining the model every time a deletion request is received can be prohibitive in terms of both time and money—especially for large models and frequent deletion requests. The problem of *data deletion* (also known as *machine unlearning*) is to find an algorithmic middle ground between the compliant but impractical baseline of retraining, and the potentially illegal standard of doing nothing. We iteratively update

models as deletion requests come in, with the twin goals of having computational cost that is substantially less than the cost of full retraining, and the guarantee that the models we produce are (almost) indistinguishable from the models that would have resulted from full retraining.

After an initial model is deployed deletion requests arrive over time as users make decisions about whether to delete their data. It is easy to see how these decisions may be *adaptive* with respect to the models. For example, security researchers may publish a new model inversion attack that identifies a specific subset of people in the training data, thus leading to increased deletion requests for people in that subset. In this paper we give the first machine unlearning algorithms that both have rigorous deletion guarantees against these kind of adaptive deletion sequence, and can accommodate arbitrary non-convex models like deep neural networks without requiring pretraining on non-user data.

1.1 Main Results

The deletion guarantees proven for several prior methods crucially rely on the implicit assumption that the points that are deleted are independent of the randomness used to train the models. However this assumption fails unless the sequence of deletion requests is chosen independently of the information that the model provider has made public. This is a very strong assumption, because users may wish to delete their data *exactly because of what deployed models reveal about them.*

We give a generic reduction. We show that if:

1. A data deletion algorithm $\mathcal{R}_{\mathcal{A}}$ for a learning algorithm $\mathcal{A}$ has deletion guarantees for *oblivious* sequences of deletion requests (as those from past work do), and

2. Information about the internal randomness of $\mathcal{R}_{\mathcal{A}}$ is revealed only in a manner that satisfies *differential privacy*, then

$(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ *also satisfies data deletion guarantees against an adaptive sequence of deletion requests*, that can depend in arbitrary ways on the information that the model provider has made public.

This generic reduction can be used to give adaptive data deletion mechanisms for a wide variety of problems by leveraging past work on deletion algorithms for non-adaptive sequences, and a line of work on differentially private aggregation [Papernot et al., 2018, Dwork and Feldman, 2018]. Since prior deletion algorithms themselves tend to use existing learning algorithms in a black-box way, the entire pipeline is modular and easy to bolt-on to existing methods. In Section 4, we show how this can be accomplished by using a variant of the SISA framework of Bourtoule et al. [2021] together with a differentially private aggregation method.

In Section 5, we complement our main result with a theoretical example and a set of experimental results on CIFAR-10, MNIST, and Fashion-MNIST that serve to illustrate two points:

1. Past method's lack of guarantees for adaptive sequences is not simply a failure of analysis, but an actual failure of these methods to satisfy deletion guarantees for adaptive deletion sequences. As an exemplar, we use two variants of SISA from Bourtoule et al. [2021] that both satisfy perfect deletion guarantees for non-adaptive deletion sequences and exhibit adaptive deletion sequences that strongly separate the resulting distribution on models compared to the retraining baseline.

2. That differential privacy may be useful in giving adaptive guarantees beyond the statement of our theorems. Specifically we show that small amounts of noise addition (insufficient for our theorems to apply) already serve to break the adaptive deletion strategies that we use to falsify the adaptive deletion guarantees in our experiments described in point 1.

1.2 Related Work

Data deletion was introduced by Cao and Yang [2015]; we adopt the randomized formulation of Ginart et al. [2019]. Ginart et al. [2019] anticipate the problem of deletion requests that might be correlated with internal state of the algorithm, and define (and propose as a study for future work) *robust* data deletion which is a

data deletion guarantee that holds for adversaries with knowledge of the internal state. Our insight is that we can provide deletion guarantees against adaptive sequences by instead obscuring the internal state of the algorithm using techniques from differential privacy.

We are the first to explicitly consider the problem of adaptive sequences of deletion requests, but some techniques from past work *do* have deletion guarantees that extend to adaptive sequences. Deterministic methods and methods that depend only on randomness that is sampled after the deletion request are already robust to adaptive deletion. This includes techniques that find an approximately optimal solution to a strongly convex problem and then perturb the solution to obscure the optimizer within a small radius e.g. Guo et al. [2019], Neel et al. [2021], Sekhari et al. [2021]. It also includes the approach of Golatkar et al. [2020a,b] which pre-trains a nonconvex model on data that will never be deleted and then does convex fine-tuning on user data on top of that. Techniques whose deletion guarantees depend on randomness sampled at training in general do not have guarantees against adaptive deletions. This includes algorithms given in Ginart et al. [2019], Bourtoule et al. [2021], Neel et al. [2021] — the SISA framework of Bourtoule et al. [2021] being of particular interest as it is agnostic to the class of models and training methodology, and so is extremely flexible.

Differential privacy has been used as a mitigation for adaptivity since the work of Dwork et al. [2015c,a]. In machine learning, it has been used to mitigate the bias of adaptive data gathering strategies as used in bandit learning algorithms [Neel and Roth, 2018]. The application that is most similar to our work is Hassidim et al. [2020], which uses differential privacy of the internal randomness of an algorithm (as we do) to reduce streaming algorithms with guarantees against adaptive adversarial streams to streaming algorithms with guarantees against oblivious adversaries. Our techniques differ; while Hassidim et al. [2020] reduce to the so-called "transfer theorem for linear and low sensitivity queries" developed over a series of works Dwork et al. [2015c], Bassily et al. [2021], Jung et al. [2020], we use a more general connection between differential privacy and "max-information" established in Dwork et al. [2015b], Rogers et al. [2016].

2 Preliminaries

Let $\mathcal{Z}$ be the data domain. A dataset D is a multi-set of elements from $\mathcal{Z}$. We consider update requests of two types: deletion and addition. These update requests are formally defined below, similar to how they are defined in [Neel et al., 2021].

Definition 2.1 (Update Operations and Sequences)**.** *An update u is a pair $(z, \bullet)$ where $z \in \mathcal{Z}$ is a datapoint and $\bullet \in \mathcal{T} = \{\texttt{'add'}, \texttt{'delete'}\}$ determines the type of the update. An update sequence U is a sequence $(u^1, u^2, \ldots)$ where $u^t \in \mathcal{Z} \times \mathcal{T}$ for all t. Given a dataset D and an update $u = (z, \bullet)$, the update operation is defined as:*

$$D \circ u \triangleq \begin{cases} D \cup \{z\} & \text{if } \bullet = \texttt{'add'} \\ D \setminus \{z\} & \text{if } \bullet = \texttt{'delete'} \end{cases}$$

Given an update sequence $U = (u^1, u^2, \ldots)$, we have $D \circ U \triangleq (((D \circ u^1) \circ u^2) \circ \ldots)$.

We use Θ to denote the space of models. A *learning* or *training* algorithm is a mapping $\mathcal{A} : \mathcal{Z}^* \to \Theta^*$ that maps a dataset $D \in \mathcal{Z}^*$ to a collection of models $\theta \in \Theta^*$. An *unlearning* or *update* algorithm for $\mathcal{A}$ is a mapping $\mathcal{R}_\mathcal{A} : \mathcal{Z}^* \times (\mathcal{Z} \times \mathcal{T}) \times \mathcal{S} \to \Theta^*$ which takes in a data set $D \in \mathcal{Z}^*$, an update request $u \in \mathcal{Z} \times \mathcal{T}$, and some current state for the algorithms $s \in \mathcal{S}$ (the domain $\mathcal{S}$ can be arbitrary), and outputs an updated collection of models $\theta' \in \Theta^*$. In this paper we consider a setting in which a stream of update requests arrive in sequence. We note that in this sequential framework, the update algorithm $\mathcal{R}_\mathcal{A}$ also updates the state of the algorithm after each update request is processed; however, for notational economy, we do not explicitly write the updated state as an output of the algorithm.

At each round, we provide access to the models through a mapping $f^t_{\text{publish}} : \Theta^* \to \Psi$ that takes in the collection of models and outputs some object $\psi \in \Psi$. A published object $\psi \in \Psi$ can, for instance, be the aggregate predictions of the learned models on a data set, or, some aggregation of the models. To model

Algorithm 1: Interaction between $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ and $\texttt{UpdReq}$

Input: Data set D
Let $D^0 \leftarrow D$.
Train $\theta^0 \leftarrow \mathcal{A}(D)$.
Publish $\psi^0 \leftarrow f^0_{\text{publish}}(\theta^0)$.
Save the initial state s^0.
for $t = 1, 2, \ldots$ **do**
  The update requester requests a new update, given the history of interaction:
    $u^t \leftarrow \texttt{UpdReq}\left(\psi^0, u^1, \psi^1, u^2, \ldots, u^{t-1}, \psi^{t-1}\right)$.
  The algorithms update, given u^t:
    Update the models $\theta^t \leftarrow \mathcal{R}_{\mathcal{A}}\left(D^{t-1}, u^t, s^{t-1}\right)$.
    Publish $\psi^t \leftarrow f^t_{\text{publish}}\left(\theta^t\right)$.
    Save the updated state s^t.
    Update the data set $D^t \leftarrow D^{t-1} \circ u^t$.

adaptively chosen update sequences, we define an arbitrary "update requester" who interacts with the learning and unlearning algorithms $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ through the publishing function f_{publish} in rounds to generate a sequence of updates. The update requester is denoted by $\texttt{UpdReq}$ and defined in Definition 2.2, and the interaction between the algorithms and the update requester is described in Algorithm 1.

Throughout we will use u^t to denote the update request at round t. We will use D^t to denote the data set at round t: D^0 is the initial training data set and for all $t \geq 1$, $D^t = D^{t-1} \circ u^t$. We will use θ^t to denote the learned models at round t: θ^0 is generated by the initial training algorithm $\mathcal{A}$, and θ^t for $t \geq 1$ denotes the updated models at round t generated by the update algorithm $\mathcal{R}_{\mathcal{A}}$. ψ^t denotes the published object at round t: $\psi^t = f^t_{\text{publish}}(\theta^t)$.

Definition 2.2 (Update Requester ($\texttt{UpdReq}$))**.** *The update sequence is generated by an update requester which is modeled by a (possibly randomized) mapping* $\texttt{UpdReq} : \Psi^* \times (\mathcal{Z} \times \mathcal{T})^* \to (\mathcal{Z} \times \mathcal{T})$ *that takes as input the history of interaction between herself and the algorithms, and outputs a new update for the current round. Given an update requester* $\texttt{UpdReq}$*, algorithms* $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ *and publishing functions* $\{f^t_{publish}\}_t$*, the update sequence* $U = \{u^t\}_t$ *can be written as*

$$u^1 = \texttt{UpdReq}\left(\psi^0\right), \; u^2 = \texttt{UpdReq}\left(\psi^0, u^1, \psi^1\right), \ldots, \; u^t = \texttt{UpdReq}\left(\psi^0, u^1, \psi^1, \ldots, u^{t-1}, \psi^{t-1}\right)$$

We say an update requester $\texttt{UpdReq}$ *is nonadaptive if it is independent of the published objects, i.e., if there exists a mapping* $\texttt{UpdReq}' : (\mathcal{Z} \times \mathcal{T})^* \to (\mathcal{Z} \times \mathcal{T})$ *such that for all* $t \geq 1$*,*

$$u^t = \texttt{UpdReq}\left(\psi^0, u^1, \psi^1, u^2, \ldots, u^{t-1}, \psi^{t-1}\right) = \texttt{UpdReq}'\left(u^1, u^2, \ldots, u^{t-1}\right)$$

This is equivalent to saying that the update sequence is fixed before the interaction occurs.

Following [Ginart et al., 2019], we propose the following definition for an unlearning algorithm in the sequential update setting ([Ginart et al., 2019] gives a definition for a single deletion request, whereas here we define a natural extension for an arbitrarily long sequence of deletions, as well as additions, that can be chosen adaptively.). Informally, we require that at every round, and for all possible update requesters, with high probability over the draw of the update sequence, no subset of models resulting from deletion occurs with substantially higher probability than it would have under full retraining.

Definition 2.3 ((α, β, γ)-unlearning)**.** *We say that* $\mathcal{R}_{\mathcal{A}}$ *is an* (α, β, γ)*-unlearning algorithm for* $\mathcal{A}$*, if for all datasets* $D = D^0$ *and all update requesters* $\texttt{UpdReq}$*, the following condition holds: For every update step* $t \geq 1$*, with probability at least* $1 - \gamma$ *over the draw of the update sequence* $u^{\leq t} = (u^1, \ldots, u^t)$ *from* $\texttt{UpdReq}$*,*

$$\forall E \subseteq \Theta^* : \quad \Pr\left[\mathcal{R}_{\mathcal{A}}\left(D^{t-1}, u^t, s^{t-1}\right) \in E \,\middle|\, u^{\leq t}\right] \leq e^{\alpha} \cdot \Pr\left[\mathcal{A}\left(D^t\right) \in E\right] + \beta$$

We say $\mathcal{R}_{\mathcal{A}}$ is a nonadaptive (α, β, γ)-unlearning algorithm for $\mathcal{A}$ if the above condition holds for any nonadaptive `UpdReq`.

Remark 2.1. *Our definition of unlearning is reminiscent of differential privacy, but following [Ginart et al., 2019], we ask only for a* one-sided *guarantee: that the probability of any event under the unlearning scheme is not too much larger than the probability of the same event under full retraining, but not vice versa. The reason is that we do not want there to be events that can substantially increase an observer's confidence that we did* not *engage in full retraining, but we do not object to observers who strongly update their beliefs that we* did *engage in full retraining. Our events E are defined directly over the sets of models in Θ^* output by $\mathcal{A}$ and $\mathcal{R}_{\mathcal{A}}$ — note that because of information processing inequalities, this is only stronger than defining events E over the observable outcome space Ψ.*

2.1 Differential Privacy and Max-Information

Differential privacy will be a key tool in our results. Let $\mathcal{X}$ denote an arbitrary data domain. We use $x \in \mathcal{X}$ to denote an individual element of $\mathcal{X}$, and $X \in \mathcal{X}^*$ to denote a collection of elements from $\mathcal{X}$ — which we call a data set. We say two data sets $X, X' \in \mathcal{X}^*$ are neighboring if they differ in at most one element. We say an algorithm $M : \mathcal{X}^n \to \mathcal{O}$ is differentially private if its output distributions on neighboring data sets are close, formalized below.

Definition 2.4 (Differential Privacy (DP) [Dwork et al., 2006b,a])**.** *An algorithm $M : \mathcal{X}^m \to \mathcal{O}$ is (ϵ, δ)-differentially private, if for every neighboring X and X', and for every $O \subseteq \mathcal{O}$, we have* $\Pr[M(X) \in O] \leq e^{\epsilon} \Pr[M(X') \in O] + \delta$.

We remark at the outset that the "datasets" to which we will eventually ask for differential privacy with respect to will not be the datasets on which our learning algorithms are trained, but will instead be collections of random bits parameterizing our randomized algorithms.

Differentially private algorithms are robust to data-independent post-processing:

Lemma 2.1 (Post-processing preserves DP [Dwork et al., 2006b])**.** *If $M : \mathcal{X}^m \to \mathcal{O}$ is (ϵ, δ)-differentially private, then for all $f : \mathcal{O} \to \mathcal{R}$, we have $f \circ M : \mathcal{X}^m \to \mathcal{R}$ defined by $f \circ M(X) = f(M(X))$ is (ϵ, δ)-differentially private.*

The max-information between two jointly distributed random variables measures how close their joint distribution is to the product of their corresponding marginal distributions.

Definition 2.5 (Max-Information [Dwork et al., 2015b])**.** *Let X and Y be jointly distributed random variables over the domain $(\mathcal{X}, \mathcal{Y})$. The β-approximate max-information between X and Y is:*

$$I_{\infty}^{\beta}(X;Y) = \log \sup_{E \subseteq (\mathcal{X},\mathcal{Y}), \Pr[(X,Y)\in E] > \beta} \frac{\Pr[(X,Y) \in E] - \beta}{\Pr[(X \otimes Y) \in E]}$$

where $(X \otimes Y)$ represents the product distribution of X and Y.

The max-information of an algorithm M that takes a dataset X as input and outputs $M(X)$, is defined as the max-information between X and $M(X)$ for the worst case product distribution over X:

Definition 2.6 (Max-Information of an Algorithm [Dwork et al., 2015b])**.** *Let $M : \mathcal{X}^m \to \mathcal{O}$ be an Algorithm. We say M has β-approximate max-information of k, written $I_{\infty}^{\beta}(M, m) \leq k$, if for every distribution $\mathcal{P}$ over $\mathcal{X}$, we have $I_{\infty}^{\beta}(X; M(X)) \leq k$ when $X \sim \mathcal{P}^m$.*

In this paper, we will use the fact that differentially private algorithms have bounded max-information:

Theorem 2.1 (DP implies bounded max-information [Rogers et al., 2016])**.** *Let $M : \mathcal{X}^m \to \mathcal{O}$ be an (ϵ, δ)-differentially private algorithm for $0 < \epsilon \leq 1/2$ and $0 < \delta < \epsilon$. Then, $I_{\infty}^{\beta}(M, m) = O\left(\epsilon^2 m + m\sqrt{\delta/\epsilon}\right)$ for $\beta = e^{-\epsilon^2 m} + O\left(m\sqrt{\delta/\epsilon}\right)$.*

3 A Reduction from Adaptive to Nonadaptive Update Requesters

In our analysis we imagine without loss of generality that the learning algorithm $\mathcal{A}$ draws an *i.i.d.* sequence of random variables $r \sim \mathcal{P}^m$ (that encodes all the randomness to be used over the course of the updates) from some distribution $\mathcal{P}$, and passes it to the unlearning algorithm $\mathcal{R}_{\mathcal{A}}$. Note r is drawn once in the initial training, and given r, $\mathcal{A}$ and $\mathcal{R}_{\mathcal{A}}$ become deterministic mappings. We can also view the state s^t as a deterministic mapping of r, the update requests so far $u^{\leq t} = (u^1, \ldots, u^t)$, and the original data set D^0. We write $s^t = g^t(D^0, u^{\leq t}, r)$ for some deterministic mapping g^t. We can therefore summarize the trajectory of the algorithms $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ as follows.

- $t = 0$: draw $r \sim \mathcal{P}^m$, let $\theta^0 = \mathcal{A}(D) \equiv \mathcal{A}(D; r)$, and $\psi^0 = f^0_{\text{publish}}\left(\theta^0\right)$.
- $t \geq 1$: $\theta^t = \mathcal{R}_{\mathcal{A}}(D^{t-1}, u^t, s^{t-1})$ where $s^{t-1} = g^{t-1}(D^0, u^{\leq t-1}, r)$, and $\psi^t = f^t_{\text{publish}}\left(\theta^t\right)$.

In this view, the randomness r used by the learning algorithm $\mathcal{A}$ and the subsequent invocations of the unlearning algorithm $\mathcal{R}_{\mathcal{A}}$ is represented as part of the internal state. Past analyses of unlearning algorithms have crucially assumed that r is statistically independent of the updates $(u^1, u^2, \ldots)$ (which is the case for non-adaptive update requesters, but not for adaptive update requesters). In the following general theorem, we show that if a learning/unlearning pair satisfies unlearning guarantees against non-adaptive update requesters, and the publishing function is differentially private *in the internal randomness* r, then the resulting algorithms also satisfy unlearning guarantees against adaptive update requesters. Note that what is important is that the publishing algorithms are differentially private in the *internal randomness* r, not in the datapoints used for training.

Theorem 3.1 (A General Theorem)**.** *Fix a pair of learning and unlearning algorithms $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ and the publishing functions $\{f^t_{publish}\}_t$. Suppose for every round t, the sequence of publishing functions $\{f^{t'}_{publish}\}_{t' \leq t}$ is (ϵ, δ)-differentially private in $r \sim \mathcal{P}^m$, for $0 < \epsilon \leq 1/2$ and $0 < \delta < \epsilon$. Suppose $\mathcal{R}_{\mathcal{A}}$ is a non-adaptive (α, β, γ)-unlearning algorithm for $\mathcal{A}$. Then $\mathcal{R}_{\mathcal{A}}$ is an $(\alpha', \beta', \gamma')$-unlearning algorithm for $\mathcal{A}$ for $\alpha' = \alpha + \epsilon', \beta' = \beta e^{\epsilon'} + \sqrt{\delta'}, \gamma' = \gamma + \sqrt{\delta'}$ where $\epsilon' = O\left(\epsilon^2 m + m\sqrt{\delta/\epsilon}\right)$ and $\delta' = e^{-\epsilon^2 m} + O\left(m\sqrt{\delta/\epsilon}\right)$.*

The proof can be found in the Appendix, but at an intuitive level, it proceeds as follows. Because it does not change the joint distribution on update requests and internal state, we can imagine in our analysis that r is redrawn after each update request from its conditional distribution, conditioned on the observed update sequence so far. Because the publishing function is differentially private in r, by the fact that post-processing preserves differential privacy (Lemma 2.1), so is the update sequence. We may therefore apply the max-information bound (Theorem 2.1), which allows us to relate the conditional distribution on r to its original (prior) distribution $\mathcal{P}^m$. But resampling r from $\mathcal{P}^m$ removes the dependence between r and the update sequence, which places us in the non-adaptive case, and allows us to apply the hypothesized unlearning guarantees for nonadaptive update requesters.

4 Distributed Algorithms

In this section, we describe a general family of distributed learning and unlearning algorithms that are in the spirit of the "SISA" framework of Bourtoule et al. [2021] (with one crucial modification). At a high level, the SISA framework operates by first randomly dividing the data into k "shards", and separately training a model on each shard. When a new point is deleted, it is removed from the shards that contained it, and only the models corresponding to those shards are retrained. The flexibility of this methodology is that the models and training procedures used in each shard can be arbitrary, as can the aggregation done at the end to convert the resulting ensemble into predictions: however these choices are instantiated, this framework gives a $(0, 0, 0)$-unlearning algorithm against any non-adaptive update requester (Lemma 4.1). Here we show that if the k shards are selected *independently* of one another, then we can apply our reduction given in the previous section with $m = k$ and obtain algorithms that satisfy deletion guarantees against adaptive update requesters.

Algorithm 2: $\mathcal{A}^{\text{distr}}$: Distributed Learning Algorithm

Input: dataset $D \equiv D^0$ of size n
Draw the shards: $D_i^0 = \texttt{Sampler}(D^0, p)$, for every $i \in [k]$.
Train the models: $\theta_i^0 = \mathcal{A}^{\text{single}}(D_i^0)$, for every $i \in [k]$.
Save the state: $s^0 = (\{D_i^0\}_{i\in[k]}, \{\theta_i^0\}_{i\in[k]})$ `// to be used for the 1st update.`
Output: $\{\theta_i^0\}_{i\in[k]}$

Algorithm 3: $\mathcal{R}_{\mathcal{A}^{\text{distr}}}$: Distributed Unlearning Algorithm: t'th round of unlearning

Input: dataset D^{t-1}, update $u^t = (z^t, \bullet^t)$, state $s^{t-1} = (\{D_i^{t-1}\}_{i\in[k]}, \{\theta_i^{t-1}\}_{i\in[k]})$
if $\bullet^t = \texttt{'delete'}$ **then**
  $S = \{i \in [k] : z^t \in D_i^{t-1}\}$ `// the shards` z^t `belongs to.`
else
  $S = \{i \in [k] : \texttt{Sampler}_i(\{z^t\}, p) \neq \{\}\}$ `// the shards` z^t `will be added to.`
Update the shards: $D_i^t = \begin{cases} D_i^{t-1} \circ u^t & \text{if } i \in S \\ D_i^{t-1} & \text{otherwise} \end{cases}$, for every $i \in [k]$.
Update the models: $\theta_i^t = \begin{cases} \mathcal{A}^{\text{single}}(D_i^t) & \text{if } i \in S \\ \theta_i^{t-1} & \text{otherwise} \end{cases}$, for every $i \in [k]$.
Update the state: $s^t = (\{D_i^t\}_{i\in[k]}, \{\theta_i^t\}_{i\in[k]})$ `// to be used for the next update.`
Output: $\{\theta_i^t\}_{i\in[k]}$

A *distributed* learning algorithm $\mathcal{A}^{\text{distr}} : \mathcal{Z}^* \to \Theta^*$ is described by a single-shard learning algorithm $\mathcal{A}^{\text{single}} : \mathcal{Z}^* \to \Theta$ and a routine `Sampler`, used to select the points in a shard. `Sampler`, given a dataset D and some probability $p \in [0, 1]$, includes each element of D in the shard with probability p.

Distributed learning algorithm $\mathcal{A}^{\text{distr}}$ creates k independent shards from the dataset D of size n by running `Sampler` k times and training a model with $\mathcal{A}^{\text{single}}$ on each shard $i \in [k]$ to form an ensemble of k models. To emphasize that the randomness across shards is independent, we will instantiate k independent samplers $\texttt{Sampler}_i$ and training algorithms $\mathcal{A}_i^{\text{single}}$ for each shard $i \in [k]$. We formally describe $\mathcal{A}^{\text{distr}}$ in Algorithm 2.

The state s of the unlearning algorithm $\mathcal{R}_{\mathcal{A}^{\text{distr}}}$ records the k shards $\{D_i\}_i$ and the ensemble of k models $\{\theta_i\}_i$. Thus $\mathcal{S} = \{\mathcal{Z}^*\}^k \times \Theta^k$. As an update request u is received, the update function removes the data point from every shard that contains it (for deletion) or adds the new point to each shard with probability p (for addition). In either case, only the models corresponding to shards that have been updated are retrained using $\mathcal{A}^{\text{single}}$. We formally describe $\mathcal{R}_{\mathcal{A}^{\text{distr}}}$ in Algorithm 3.

First, we show that if the update requester is non-adaptive, $\mathcal{R}_{\mathcal{A}^{\text{distr}}}$ is a $(0, 0, 0)$-unlearning algorithm:

Lemma 4.1. *$\mathcal{R}_{\mathcal{A}^{distr}}$ is a non-adaptive $(0, 0, 0)$-unlearning algorithm for $\mathcal{A}^{distr}$.*

Now, by combining Lemma 4.1 and our general Theorem 3.1, we can show the following:

Theorem 4.1 (Unlearning Guarantees)**.** *If for every round t, the sequence of publishing functions $\{f_{publish}^{t'}\}_{t'\leq t}$ is (ϵ, δ)-differentially private in the random seeds $r \sim \mathcal{P}^k$ of the algorithms for $0 < \epsilon \leq 1/2$ and $0 < \delta < \epsilon$, then $\mathcal{R}_{\mathcal{A}^{distr}}$ is an (α, β, γ)-unlearning algorithm for $\mathcal{A}^{distr}$ where*

$$\alpha = O\left(\epsilon^2 k + k\sqrt{\delta/\epsilon}\right), \quad \beta = \gamma = O\left(\sqrt{e^{-\epsilon^2 k} + k\sqrt{\delta/\epsilon}}\right)$$

Next, we bound the time complexity of our algorithms:

Theorem 4.2 (Run-time Guarantees)**.** *Let $p = 1/k$. Suppose the publishing functions satisfy the differential privacy requirement of Theorem 4.1. Let N^t denote the number of times $\mathcal{R}_{\mathcal{A}}^{distr}$ calls $\mathcal{A}^{single}$ at round t. We*

have that $N^0 = k$*, and for every round* $t \geq 1$*: 1) if the update requester is non-adaptive, for every* ξ*, with probability at least* $1 - \xi$*,* $N^t \leq 1 + \sqrt{2\log(1/\xi)}$*. 2) if the update requester is adaptive, for every* ξ*, with probability at least* $1 - \xi$*,* $N^t \leq 1 + \sqrt{2\log((n+t)/\xi)}$*. Furthermore, for* $\xi > \delta'$*, with probability at least* $1 - \xi$*, we have*

$$N^t \leq 1 + \min\left\{\sqrt{2\log(2(n+t)/(\xi - \delta'))}, \sqrt{2\epsilon' + 2\log(2/(\xi - \delta'))}\right\}$$

where $\epsilon' = O\left(\epsilon^2 k + k\sqrt{\delta/\epsilon}\right)$ *and* $\delta' = e^{-\epsilon^2 k} + O\left(k\sqrt{\delta/\epsilon}\right)$

The proof can be found in the appendix, but at a high level it proceeds as follows. For a deletion request, we must retrain every shard that contains the point to be deleted. For a non-adaptive deletion request, we retrain one shard in expectation and we can obtain a high probability upper bound by using a Hoeffding bound. In the adaptive case, this may no longer be true, but there are two ways to obtain upper bounds that correspond to the two bounds in our Theorem. We can provide a worst-case upper bound on the number of shards that *any* of the n data points belongs to, which incurs a cost of order $\sqrt{\log n}$. Alternately, we can apply max-information bounds to reduce to the non-adaptive case, using an argument that is similar to our reduction for deletion guarantees.

4.1 Private Aggregation

We briefly describe how we serve prediction requests by privately aggregating the output of the ensemble of models such that the published predictions are differentially private in the random seeds r. At each round t, while $\mathcal{R}_{\mathcal{A}}^{\text{distr}}$ is waiting for the next update request u^{t+1}, we receive prediction requests x and serve predictions $\hat{y}$. For each prediction request, we privately aggregate the predictions made by the ensemble of models $\{\theta_i^t\}_i$; Dwork and Feldman [2018] show several ways to privately aggregate predictions (one simple technique is to use the exponential mechanism to approximate the majority vote). Suppose we aggregate the predictions made by the ensemble of models using $\texttt{PrivatePredict}_{\epsilon'}^k : \Theta^k \times \mathcal{X} \to \mathcal{Y}$, which takes in an ensemble of k models and a data point, aggregates predictions from the ensemble models, and outputs a label that is ϵ'-differentially private in the models. If we receive l^t many prediction requests $(x_1^t, \ldots, x_{l^t}^t)$ before our next update request u^{t+1}, we can write $(\hat{y}_1^t, \ldots, \hat{y}_{l^t}^t) = f_{\text{publish}}^t(\{\theta_i^t\}_i)$ where $\hat{y}_j^t = \texttt{PrivatePredict}_{\epsilon'}^k(\{\theta_i^t\}_i, x_j^t)$.

Theorem 4.1, tells us that desired unlearning parameters (α, β, γ) can be obtained by guaranteeing that the sequence of predictions is (ϵ, δ) differentially private in the models (and hence r), for target parameters ϵ, δ. As we serve prediction requests using $\texttt{PrivatePredict}_{\epsilon'}^k$ our privacy loss will accumulate and eventually exhaust our budget of (ϵ, δ)-differential privacy. Hence we must track our accumulated privacy loss in the state of our unlearning algorithm, and when it is exhausted, fully retrain using $\mathcal{A}^{\text{distr}}$. This resamples r and hence resets our privacy budget. Standard composition theorems (see Dwork and Roth [2014]) show that we exhaust our privacy budget (and need to fully retrain) every time the number of prediction requests made since the last full retraining exceeds $\left\lfloor \frac{\epsilon^2}{8(\epsilon')^2 \ln(\frac{1}{\delta})} \right\rfloor$. We formally describe this process denoted as $\texttt{PrivatePredictionInteraction}(\epsilon', \epsilon, \delta, k)$ in the appendix and state its unlearning guarantee in Theorem 4.3.

Theorem 4.3. *The models* $\{\{\theta_i^t\}_i\}_t$ *in* $\texttt{PrivatePredictionInteraction}(\epsilon', \epsilon, \delta, k)$ *satisfy* (α, β, γ)*-unlearning guarantee for* $\mathcal{A}^{distr}$ *where* $\alpha = O\left(\epsilon^2 k + k\sqrt{\delta/\epsilon}\right)$ *and* $\beta, \gamma = O\left(\sqrt{e^{-\epsilon^2 k} + k\sqrt{\delta/\epsilon}}\right)$*, if* $0 < \epsilon \leq 1/2$ *and* $0 < \delta < \epsilon$*.*

5 Evaluation of Unlearning Guarantees

In this section we demonstrate that the deletion guarantees of algorithms in the SISA framework [Bourtoule et al., 2021] fail for adaptive deletion sequences. In Section 5.1 we give a clean toy construction which shows algorithms in the SISA framework fail to have nontrivial adaptive deletion guarantees even in the black-box setting when the models within each shard are not made public, only aggregations of their

classification outputs. In the Appendix we experimentally evaluate a more realistic instantiation of this construction. In Section 5.2 we consider the white-box setting in which the models in each shard are made public. SISA continues to have perfect deletion guarantees against *non-adaptive* deletion sequences in this setting. Experimental results on CIFAR-10 [Krizhevsky and Hinton, 2009], MNIST [Lecun et al., 1998], and Fashion-MNIST [Xiao et al., 2017] show both the failure of SISA to satisfy adaptive deletion guarantees, and give evidence that differential privacy can mitigate this problem well beyond the setting of our theorems while achieving accuracy only modestly worse than SISA. The code for our experiments can be found at `https://github.com/ChrisWaites/adaptive-machine-unlearning`.

5.1 Theory for the Label-Only Setting

The first setting we consider directly corresponds to the setting in which our final algorithms operate: what is made public is the aggregate predictions of the ensemble of models, but not the models themselves. For non-adaptive sequences of deletions, distributed algorithms of the sort described in Section 4 have perfect deletion guarantees. We demonstrate via a simple example that these guarantees dramatically fail for adaptive deletion sequences.

Suppose we have a dataset consisting of real-valued points with binary labels $\{(x_i, y_i)\}_{i=1}^{2n}$, $x_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$ in which there are exactly two copies of each distinct training example. Consider a simplistic classification model, resembling a lookup table, which given a point x_i predicts the label y_i if the model has been trained on (x_i, y_i) and a dummy prediction value "$\perp$" otherwise:

$$f_{\mathcal{D}}(x_i) = \begin{cases} y_i & \text{if } (x_i, y_i) \in \mathcal{D}, \\ \perp & \text{otherwise} \end{cases}$$

Consider what happens when the training algorithm randomly partitions this dataset into three pieces and trains such a model on each partition. This constructs an ensemble which, at query time, predicts the class with the majority vote. On this dataset, the ensemble will predict the labels of roughly 2/3 of the training points correctly—that is, exactly those points for which the duplicates have fallen into distinct partitions, so that the ensemble gets the majority vote right.

We construct an adaptive adversary who chooses to delete exactly those training points that the ensemble correctly classifies (which are those points for whom the duplicates have fallen into distinct shards). The result is that the model resulting from this deletion sequence will misclassify every remaining training point. Full retraining (because it would rerandomize the partition) would again lead to training accuracy of approximately 2/3. Recalling that our deletion notion requires that the probability of any event under the unlearning scheme is not much larger than the probability of the same event under full retraining, this demonstrates that there are algorithms in the SISA framework — even if the models are not directly exposed — that do not satisfy (α, β, γ)-deletion guarantees for any nontrivial value of α. We formalize this below:

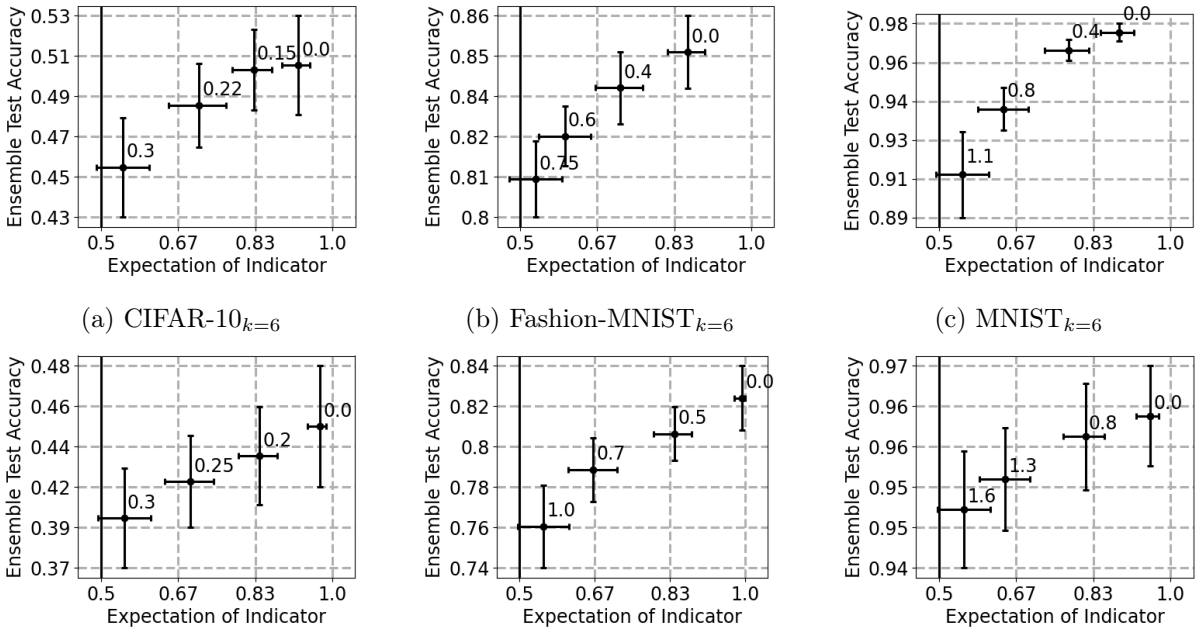
Theorem 5.1. *There are learning and unlearning algorithms in the SISA framework $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ such that for any α, and any $\beta, \gamma < 1/4$, $\mathcal{R}_{\mathcal{A}}$ is not an (α, β, γ)-unlearning algorithm for $\mathcal{A}$.*

A proof of this theorem can be found in the appendix.

5.2 Experiments for the Full-Model Setting

We train SISA with an ensemble of convolutional neural networks on several datasets of points with categorical labels. Given a new point at query time, each model in the ensemble votes on the most likely label and aggregates their votes. The models are exposed publicly. This scheme has perfect non-adaptive deletion guarantees.

To construct an adaptive deletion sequence to falsify the hypothesis that the scheme has adaptive deletion guarantees, we exploit the observation that neural networks are often *overconfident* in the correct label for points on which they have been trained. For each training point, we guess that it falls into the shard corresponding to the model that has the highest confidence for the correct label. We then delete points for



(a) CIFAR-10$_{k=6}$ (b) Fashion-MNIST$_{k=6}$ (c) MNIST$_{k=6}$

(d) CIFAR-10$_{k=2}$ (e) Fashion-MNIST$_{k=2}$ (f) MNIST$_{k=2}$

Figure 1: The top row and bottom row show experiments with $k = 6$ and $k = 2$ shards respectively. The 3 columns report on 3 datasets. The x axis denotes estimated expectation of our test statistic (the null hypothesis is expectation 0.5). The y axis denotes the accuracy of the ensemble after deletion. Each point is annotated with the noise multiplier used in DP-SGD, the standard deviation of Gaussian noise applied to gradients during training. A label of 0.0 for a point represents the baseline case of no noise (original SISA algorithm). Points are affixed with 95% confidence intervals along both axes (over the randomness of repeating the training/deletion experiment). Horizontal confidence intervals that overlap the line denoting expectation 0.5 fail to reject the null hypothesis that the algorithm has adaptive data deletion guarantees at $p \leq 0.05$. We get to this point with a level of noise addition that results in only a modest degradation in ensemble performance compared to SISA.

which we guess that they fall into the first $k/2$ of the shards, and do not delete any others. After deleting the targeted points, we compute a test statistic: the indicator of whether the average accuracy of the models from the targeted shards is lower than the average accuracy of the models from the non-targeted shards. Under full retraining, by the symmetry of the random partition, the expectation of this test statistic is 0.5. Thus under the null hypothesis that the deletion algorithm satisfies perfect deletion guarantees, the test statistic also has expectation 0.5. Therefore, to the extent that the expectation of the indicator differs from 0.5, we falsify the null hypothesis that SISA has adaptive data deletion guarantees, and larger deviations from 0.5 falsify weaker deletion guarantees.

We run this experiment on three datasets (CIFAR-10, MNIST, and Fashion-MNIST), and plot the results in Figure 1. We then repeat the experiment by adding various amounts of noise to the gradients in the model training process to guarantee finite levels of differential privacy (though much weaker privacy guarantees than would be needed to invoke our theorems). We observe that on each dataset, modest amounts of noise are sufficient to break our attack (i.e. 95% confidence intervals for the expectation of our indicator include 0.5, and hence fail to falsify the null hypothesis) while still approaching the accuracy of our models trained without differential privacy. This is also plotted in Figure 1. This gives evidence that differential privacy can improve deletion guarantees in the presence of adaptivity even in regimes beyond which our theory gives nontrivial guarantees.

Full experimental details can be found in the appendix.

6 Conclusion and Discussion

We identify an important blindspot in the data deletion literature (the tenuous implicit assumption that deletion requests are independent of previously released models), and provide a very general methodology to reduce adaptive deletion guarantees to oblivious deletion guarantees. Through this reduction we get the first model and training algorithm agnostic methodology that allows for deletion of arbitrary sequences of adaptively chosen points while giving rigorous guarantees. The constants that our theorems inherit from the max information bounds of Rogers et al. [2016] are such that in most realistic settings they will not give useful parameters. But we hope that these constants will be improved in future work, and we give empirical evidence that differential privacy mitigates adaptive deletion "attacks" at very practical levels, beyond the promises of our theoretical results. We note that like for differential privacy, the (α, β, γ)-deletion guarantees we give in this paper are *parameterized*, and are not meaningful absent a specification of those parameters. There is a risk with such technologies that they will be used with large values of the parameters that give only very weak guarantees, but will be described publicly in a way that glosses over this issue. We therefore recommend that if adopted in deployed products, deletion guarantees always be discussed in public in a way that is precise about what they promise, including the relevant parameter settings.

References

Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. *SIAM Journal on Computing*, (0):STOC16–377, 2021.

Lucas Bourtoule, Varun Chandrasekaran, Christopher Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy*, San Francisco, CA., 2021.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

Cynthia Dwork and Vitaly Feldman. Privacy-preserving prediction. *CoRR*, abs/1803.10266, 2018. URL `http://arxiv.org/abs/1803.10266`.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006b.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015a.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2350–2358, 2015b.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126, 2015c.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1322–1333. ACM, 2015. doi: 10.1145/2810103.2813677. URL https://doi.org/10.1145/2810103.2813677.

Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making AI forget you: Data deletion in machine learning. *CoRR*, abs/1907.05012, 2019. URL http://arxiv.org/abs/1907.05012.

Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. *arXiv preprint arXiv:2012.13431*, 2020a.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. In *European Conference on Computer Vision*, pages 383–398. Springer, 2020b.

Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0172d289da48c48de8c5ebf3de9f7ee1-Abstract.html.

The U.K. Information Commissioner's Office ICO. Guidance on the ai auditing framework. Draft Consultation, 2020. URL https://ico.org.uk/media/about-the-ico/consultations/2617219/guidance-on-the-ai-auditing-framework-draft-for-consultation.pdf.

Christopher Jung, Katrina Ligett, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Moshe Shenfeld. A new analysis of differential privacy's generalization guarantees. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, page 31. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

Seth Neel and Aaron Roth. Mitigating bias in adaptive data gathering via differential privacy. In *International Conference on Machine Learning*, pages 3720–3729. PMLR, 2018.

Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. In *Algorithmic Learning Theory*, pages 931–962. PMLR, 2021.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate, 2018.

Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Ulfar Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. *The 35th AAAI Conference on Artificial Intelligence*, 2021.

Ryan Rogers, Aaron Roth, Adam Smith, and Om Thakkar. Max-information, differential privacy, and post-selection hypothesis testing. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 487–494. IEEE, 2016.

Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *arXiv preprint arXiv:2103.03279*, 2021.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: Model inversion attacks and data protection law. *CoRR*, abs/1807.04644, 2018. URL http://arxiv.org/abs/1807.04644.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.

A Proof of Theorem 3.1

We first state the following Lemma which we will use to prove Theorem 3.1.

Lemma A.1 ([Rogers et al., 2016]). *Let $M : \mathcal{X}^m \to \mathcal{O}$ be an (ϵ, δ)-differentially private algorithm for $0 < \epsilon \leq 1/2$ and $0 < \delta < \epsilon$. Then,*

$$\Pr_{(x,m')\sim(X,M(X))}\left[\log\left(\frac{\Pr\left[X = x, M(X) = m'\right]}{\Pr\left[X = x\right]\Pr\left[M(X) = m'\right]}\right) \geq k\right] \leq \beta$$

where the probability is taken with respect to the joint distribution of X and $M(X)$, and

$$k = O\left(\epsilon^2 m + m\sqrt{\frac{\delta}{\epsilon}}\right), \quad \beta = e^{-\epsilon^2 m} + O\left(m\sqrt{\frac{\delta}{\epsilon}}\right)$$

Theorem 3.1 (A General Theorem). *Fix a pair of learning and unlearning algorithms $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ and the publishing functions $\{f^t_{publish}\}_t$. Suppose for every round t, the sequence of publishing functions $\{f^{t'}_{publish}\}_{t' \leq t}$ is (ϵ, δ)-differentially private in $r \sim \mathcal{P}^m$, for $0 < \epsilon \leq 1/2$ and $0 < \delta < \epsilon$. Suppose $\mathcal{R}_{\mathcal{A}}$ is a non-adaptive (α, β, γ)-unlearning algorithm for $\mathcal{A}$. Then $\mathcal{R}_{\mathcal{A}}$ is an $(\alpha', \beta', \gamma')$-unlearning algorithm for $\mathcal{A}$ for $\alpha' = \alpha + \epsilon', \beta' = \beta e^{\epsilon'} + \sqrt{\delta'}, \gamma' = \gamma + \sqrt{\delta'}$ where $\epsilon' = O\left(\epsilon^2 m + m\sqrt{\delta/\epsilon}\right)$ and $\delta' = e^{-\epsilon^2 m} + O\left(m\sqrt{\delta/\epsilon}\right)$.*

Proof. Fix a data set D and an update requester `UpdReq`. Fix any unlearning step $t \geq 1$. Note that the sequence of updates up to round t, i.e. $u^{\leq t} = (u^1, \ldots, u^t)$, can be seen as a post-processing of the sequence of published objects up to round $t-1$, i.e. $\psi^{\leq t-1} = (\psi^0, \ldots, \psi^{t-1})$, where the post-processing function is defined by `UpdReq` (see Definition 2.2). But we know that $\{f^{t'}_{\text{publish}}\}_{t' \leq t-1}$ that generates $\psi^{\leq t-1}$ is (ϵ, δ)-differentially private in r. Hence, given that post-processing preserves differential privacy (Lemma 2.1), we have that $u^{\leq t}$ is also (ϵ, δ)-differentially private in r. Consequently, we can apply the fact that DP implies bounded max-information (Lemma A.1) to get that

$$\Pr_{(r,u^{\leq t})}\left[\log\frac{\Pr\left[r|u^{\leq t}\right]}{\Pr\left[r\right]} \geq \epsilon'\right] = \Pr_{(r,u^{\leq t})}\left[\log\frac{\Pr\left[r, u^{\leq t}\right]}{\Pr\left[r\right]\Pr\left[u^{\leq t}\right]} \geq \epsilon'\right] \leq \delta' \tag{1}$$

where the probability is taken with respect to the joint distribution of $(r, u^{\leq t})$, and that

$$\epsilon' \triangleq O\left(\epsilon^2 m + m\sqrt{\frac{\delta}{\epsilon}}\right), \quad \delta' \triangleq e^{-\epsilon^2 m} + O\left(m\sqrt{\frac{\delta}{\epsilon}}\right)$$

Now define the "Good" event for the update sequence $u^{\leq t}$:

$$G = \left\{u^{\leq t} : \Pr_{r|u^{\leq t}}\left[\log\frac{\Pr\left[r|u^{\leq t}\right]}{\Pr\left[r\right]} \geq \epsilon'\right] \leq \sqrt{\delta'}\right\}$$

We have that

$$\begin{aligned}
\Pr_{u^{\leq t}}\left[u^{\leq t} \notin G\right] &= \Pr_{u^{\leq t}}\left[\Pr_{r|u^{\leq t}}\left[\log\frac{\Pr\left[r|u^{\leq t}\right]}{\Pr\left[r\right]} \geq \epsilon'\right] > \sqrt{\delta'}\right] \\
&\leq \frac{\mathbb{E}_{u^{\leq t}}\left[\Pr_{r|u^{\leq t}}\left[\log\frac{\Pr[r|u^{\leq t}]}{\Pr[r]} \geq \epsilon'\right]\right]}{\sqrt{\delta'}} \\
&= \frac{\Pr_{(r,u^{\leq t})}\left[\log\frac{\Pr[r|u^{\leq t}]}{\Pr[r]} \geq \epsilon'\right]}{\sqrt{\delta'}} \\
&\leq \sqrt{\delta'}
\end{aligned}$$

where the first inequality is an application of Markov's inequality, and the last one follows from Equation (1). Therefore, if we condition on $\{u^{\leq t} \in G\}$ which happens with probability at least $1 - \sqrt{\delta'}$, we have the following guarantee.

$$\Pr_{r|u^{\leq t}}\left[\log \frac{\Pr\left[r|u^{\leq t}\right]}{\Pr\left[r\right]} \geq \epsilon'\right] \leq \sqrt{\delta'}$$

which in turn implies, with probability $1 - \sqrt{\delta'}$ over the draw of $u^{\leq t}$, that for every event F in the space of random seeds (r),

$$\Pr\left[r \in F \,|\, u^{\leq t}\right] \leq e^{\epsilon'} \Pr\left[r \in F\right] + \sqrt{\delta'} \tag{2}$$

Now we condition on $\{u^{\leq t} \in G\}$. Fix any event $E \subseteq \Theta^*$ in the space of models, and let $F = \{r : \mathcal{R}_{\mathcal{A}}(D^{t-1}, u^t, s^{t-1}) \in E\}$ be the event that the output models of the unlearning algorithm on round t belongs to E, recalling that $s^{t-1} = g^{t-1}(D^0, u^{\leq t-1}, r)$. Substituting F in Equation (2), we get that

$$\Pr\left[\mathcal{R}_{\mathcal{A}}(D^{t-1}, u^t, s^{t-1}) \in E \,|\, u^{\leq t}\right] \leq e^{\epsilon'} \Pr\left[\mathcal{R}_{\mathcal{A}}(D^{t-1}, u^t, s^{t-1}) \in E\right] + \sqrt{\delta'} \tag{3}$$

Note that because on the right hand side we do not condition the probability on the update sequence, we are taking the probability over the distribution of output models of round t for a nonadaptively chosen update sequence. Therefore by the unlearning guarantees for nonadaptive update requesters, we have that with probability at least $1 - \gamma$ over the draw of $u^{\leq t}$,

$$\Pr\left[\mathcal{R}_{\mathcal{A}}(D^{t-1}, u^t, s^{t-1}) \in E\right] \leq e^{\alpha} \Pr\left[\mathcal{A}(D^t) \in E\right] + \beta \tag{4}$$

Now we can combine Equations (3) and (4) to conclude that, with probability $1 - \gamma - \sqrt{\delta'}$ over $u^{\leq t}$,

$$\Pr\left[\mathcal{R}_{\mathcal{A}}(D^{t-1}, u^t, s^{t-1}) \in E \,|\, u^{\leq t}\right] \leq e^{\alpha+\epsilon'} \Pr\left[\mathcal{A}(D^t) \in E\right] + \beta e^{\epsilon'} + \sqrt{\delta'}$$

completing the proof. □

B Missing Details from Section 4

Lemma B.1. *Consider the distributed learning and unlearning algorithms $\mathcal{A}^{distr}$ and $\mathcal{R}_{\mathcal{A}}^{distr}$. If the update requester is non-adaptive, for every t: for every shard i, we have D_i^t is an independent draw from the distribution of* $\texttt{Sampler}(D^t, p)$.

Proof. We prove this via induction. It's easy to see that this holds true at round $t = 0$ because we explicitly set $D_i^0 = \texttt{Sampler}(D^0, p)$. Now, suppose that $D_i^{\tau-1}$ is an independent draw from the distribution of $\texttt{Sampler}(D^{\tau-1}, p)$ for some $\tau \geq 1$. If the update request $u^\tau = (z^\tau, \texttt{'delete'})$ is a deletion request, then it's easy to see that simply deleting the point z^τ from every shard that contains it will maintain that each element is chosen to be in the shard with probability p. And $D_i^\tau | u^{\tau-1}$ and $D_i^\tau | u^\tau$ must be identically distributed because the update request u^τ is non-adaptive and has been fixed prior to the interaction — and hence is statistically independent of $D^{\tau-1}$. More formally, we have that for any $z \in D^\tau$,

$$\Pr[z \in D_i^\tau] = \Pr[z \in D_i^\tau | u^{\leq \tau}] = \Pr[z \in D_i^\tau | u^{\leq \tau-1}] = \Pr[z \in D_i^{\tau-1} | u^{\leq \tau-1}] = p.$$

The same argument applies for the addition request where $\mathcal{R}_{\mathcal{A}}^{\text{distr}}$ adds the element requested to be added with probability p. More formally, we have $\Pr[z \in D_i^\tau] = p$ for any $z \in D^{\tau-1}$ and $\Pr[z^\tau \in D_i^\tau] = p$ by construction. □

Lemma 4.1. *$\mathcal{R}_{\mathcal{A}^{distr}}$ is a non-adaptive $(0, 0, 0)$-unlearning algorithm for $\mathcal{A}^{distr}$.*

Proof. Fix any arbitrary round $t \in [T]$. For a non-adaptive `UpdReq`, we can think of the update sequence $u^{\leq t}$ as fixed prior to the start of the interaction between the learning procedure and the `UpdReq`. Now, in order to show $(0,0,0)$-deletion guarantee of the unlearning algorithm, we need to show that for any $E \subseteq \Theta^*$,

$$\Pr\left[\mathcal{R}_{\mathcal{A}^{\text{distr}}}(D^{t-1}, u^t, s^{t-1}) \in E | u^{\leq t}\right] = \Pr\left[\mathcal{A}^{\text{distr}}(D^t) \in E\right].$$

Note that it is equivalent to show that for any $i \in [k]$ and $E \subseteq \Theta$, we have

$$\Pr\left[\theta_i^t \in E | u^{\leq t}\right] = \Pr\left[\mathcal{A}_i^{\text{single}}(\texttt{Sampler}_i(D^t, p)) \in E\right]$$

because $\texttt{Sampler}_i$ and $\mathcal{A}_i^{\text{single}}$ behave independently across $i \in [k]$ in both $\mathcal{R}_{\mathcal{A}^{\text{distr}}}$ and $\mathcal{A}^{\text{distr}}$. Hence, from here on, we focus on some fixed $i \in [k]$.

Now, we argue that it is sufficient to show that the distribution over D_i^t conditional on $u^{\leq t}$ that is being kept in the state s^t of the unlearning algorithm is exactly the same as that of $\texttt{Sampler}_i(D^t, p)$, which we have already proved in Lemma B.1. Using the fact that update sequence is non-adaptive with respect to the algorithm's randomness, we have for any realization path for shard i until round t (i.e. how the initial shard D_i^0 was formed and whether each addition request until round t was actually added to shard i or not)

$$\begin{aligned}
\Pr[\theta_i^t \in E | u^{\leq t}] &= \Pr[\theta_i^{t'} \in E | u^{\leq t}] \\
&= \Pr[\theta_i^{t'} \in E | u^{\leq t'}] \\
&= \Pr[\mathcal{A}_i^{\text{single}}(D_i^{t'}) \in E | u^{\leq t'}] \\
&= \Pr[\mathcal{A}_i^{\text{single}}(D_i^{t}) \in E | u^{\leq t}]
\end{aligned}$$

where $t' = \min\{\tau \leq t : D_i^\tau = D_i^t\}$ is the time at which we last trained the model for shard i in the unlearning algorithm. □

Theorem 4.1 (Unlearning Guarantees)**.** *If for every round t, the sequence of publishing functions $\{f_{publish}^{t'}\}_{t' \leq t}$ is (ϵ, δ)-differentially private in the random seeds $r \sim \mathcal{P}^k$ of the algorithms for $0 < \epsilon \leq 1/2$ and $0 < \delta < \epsilon$, then $\mathcal{R}_{\mathcal{A}^{distr}}$ is an (α, β, γ)-unlearning algorithm for $\mathcal{A}^{distr}$ where*

$$\alpha = O\left(\epsilon^2 k + k\sqrt{\delta/\epsilon}\right), \quad \beta = \gamma = O\left(\sqrt{e^{-\epsilon^2 k} + k\sqrt{\delta/\epsilon}}\right)$$

Proof. Lemma 4.1 provides that $\mathcal{R}_{\mathcal{A}^{\text{distr}}}$ is a $(0,0,0)$-unlearning algorithm for $\mathcal{A}^{\text{distr}}$ against any nonadaptive update requester.

Note that because the randomness used in each shard $i \in [k]$ is always independent and there is a symmetry across these shards in both $\mathcal{A}^{\text{distr}}$ and $\mathcal{R}^{\text{iter}}_{\mathcal{A}^{\text{distr}}}$, we can imagine drawing all the randomness required for each shard throughout the interaction prior to the interaction $r \sim \mathcal{P}^k$ such that each shard $i \in [k]$ relies r_i on as the source of its randomness.

Now, note that the state kept by $\mathcal{R}^{\text{distr}}_{\mathcal{A}}$ consists of the shards $\{D_i^{t-1}\}_i$ and the models trained via $\mathcal{A}^{\text{single}}$ on those shards $\{\theta_i^{t-1}\}_i$. Hence, at any round t, given access to initial dataset D^0, previous update requests $u^{\leq t-1}$, and the randomness that has been drawn prior to the interaction r, we can deterministically determine the state $s^{t-1} = (\{D_i^{t-1}\}_i, \{\theta_i^{t-1}\}_i)$, meaning there exists some deterministic mapping g^{t-1} such that $s^{t-1} = g^{t-1}(D^0, u^{\leq t-1}, r)$.

Therefore, we can combine the $(0,0,0)$-deletion guarantee promised by Lemma 4.1 with Theorem 3.1 to conclude that $\mathcal{R}^{\text{distr}}_{\mathcal{A}}$ must be (α, β, γ)-unlearning algorithm for $\mathcal{A}^{\text{distr}}$. □

Theorem 4.2 (Run-time Guarantees)**.** *Let $p = 1/k$. Suppose the publishing functions satisfy the differential privacy requirement of Theorem 4.1. Let N^t denote the number of times $\mathcal{R}^{distr}_{\mathcal{A}}$ calls $\mathcal{A}^{single}$ at round t. We have that $N^0 = k$, and for every round $t \geq 1$: 1) if the update requester is non-adaptive, for every ξ, with probability at least $1 - \xi$, $N^t \leq 1 + \sqrt{2\log(1/\xi)}$. 2) if the update requester is adaptive, for every ξ, with*

probability at least $1-\xi$, $N^t \le 1+\sqrt{2\log((n+t)/\xi)}$. *Furthermore, for* $\xi > \delta'$, *with probability at least* $1-\xi$, *we have*

$$N^t \le 1 + \min\left\{\sqrt{2\log\left(2(n+t)/(\xi-\delta')\right)}, \sqrt{2\epsilon' + 2\log\left(2/(\xi-\delta')\right)}\right\}$$

where $\epsilon' = O\left(\epsilon^2 k + k\sqrt{\delta/\epsilon}\right)$ *and* $\delta' = e^{-\epsilon^2 k} + O\left(k\sqrt{\delta/\epsilon}\right)$

Proof. Throughout we use $Bin(k,p)$ to denote a binomial random variable with parameters k (number of trials) and p (success probability). First we state the following fact:

Fact B.1 (Binomial Tail Bound)**.** *Let* $X \sim Bin(k,p)$ *and let* $\mu := kp$. *We have that for every* $\eta \ge 0$,

$$\Pr\left[X \ge (1+\eta)\mu\right] \le e^{-\frac{\eta^2\mu}{2+\eta}}$$

which in turn implies, for every δ, *with probability at least* $1-\delta$,

$$X \le \left(1 + \frac{\sqrt{\log^2(1/\delta) + 8\mu\log(1/\delta)} - \log(1/\delta)}{2\mu}\right)\mu \le \mu + \sqrt{2\mu\log(1/\delta)}$$

Fix any round $t \ge 1$ of the update, and let $\mu = kp$ throughout. Suppose the update requester is non-adaptive. If the update of round t is an addition, then $N^t \sim Bin(k,p)$ by construction. If the update of round t is a deletion: $u^t = (z^t, \texttt{'delete'})$, then

$$N^t = \sum_{i=1}^{k} \mathbb{1}\left[z^t \in D_i^{t-1}\right]$$

But the update requester being non-adaptive (implying z^t is independent of the randomness of the algorithms), together with Lemma B.1, imply that N^t is a sum of *independent* Bernoulli random variables with parameter p; hence, $N^t \sim Bin(k,p)$. Therefore, if the update requester is non-adaptive, we can apply Fact B.1 to conclude that for every ξ, with probability at least $1-\xi$, we have

$$N^t \le \mu + \sqrt{2\mu\log(1/\xi)}$$

which proves the first part of the theorem for the choice of $p = 1/k$. Now suppose the update requester is adaptive. If the update of round t is an addition, then $N^t \sim Bin(k,p)$ by construction, and therefore using Fact B.1, with probability at least $1-\xi$, we have $N^t \le \mu + \sqrt{2\mu\log(1/\xi)}$. Now suppose the update is a deletion: $u^t = (z^t, \texttt{'delete'})$. We have in this case that

$$N^t = \sum_{i=1}^{k} \mathbb{1}\left[z^t \in D_i^{t-1}\right]$$

First note that we have the following upper bound

$$N^t \le \sup_{z \in D^{t-1}} \sum_{i=1}^{k} \mathbb{1}\left[z \in D_i^{t-1}\right] \le \sup_{z \in D^0 \cup \{z^1,\ldots,z^{t-1}\}} \sum_{i=1}^{k} \mathbb{1}\left[z \in D_i^{t-1}\right] \tag{5}$$

where $\{z^1,\ldots,z^{t-1}\}$ are the data points that have been requested to be added or deleted by the update requester in the previous rounds. Here, in the worst case (to get upper bounds), we are assuming that all previous $t-1$ updates are addition requests. Note that for every $z \in D^0 \cup \{z^1,\ldots,z^{t-1}\}$, the number of shards that contain z is an independent draw from a $Bin(k,p)$ distribution, by construction. We therefore have that

$$\sup_{z \in D^0 \cup \{z^1,\ldots,z^{t-1}\}} \sum_{i=1}^{k} \mathbb{1}\left[z \in D_i^{t-1}\right] \overset{d}{=} \sup_{1 \le j \le n+t-1} X_j \tag{6}$$

where the equality is in distribution, and $X_j \sim Bin(k,p)$. Now, combining Equations (5) and (6), and using Fact B.1, we get that for every $\eta \geq 0$,

$$\Pr\left[N^t \geq (1+\eta)\,\mu\right] \leq \sum_{j=1}^{n+t-1} \Pr\left[X_j \geq (1+\eta)\,\mu\right] \leq (n+t)e^{-\frac{\eta^2\mu}{2+\eta}}$$

which implies, for every $\xi \geq 0$, with probability at least $1-\xi$,

$$N^t \leq \mu + \sqrt{2\mu \log\left((n+t)/\xi\right)}. \tag{7}$$

We will prove another upper bound using the max-information bound. Recall that our distributed algorithms can be seen as drawing all the randomness $r \sim \mathcal{P}^k$ upfront for some distribution $\mathcal{P}$ (one draw from $\mathcal{P}$ per shard). Since the update sequence $u^{\leq t}$ (which is a post processing of the published objects) is guaranteed to be (ϵ,δ)-differentially private in r, we get using the max-information bound that, for every $\eta \geq 0$,

$$\Pr\left[N^t \geq (1+\eta)\,\mu\right] \leq e^{\epsilon'} \Pr_{(r \otimes u^{\leq t})}\left[N^t \geq (1+\eta)\,\mu\right] + \delta' \tag{8}$$

where on the left hand side the probability is taken with respect to the joint distribution of r and $u^{\leq t}$, and on the right hand side $(r \otimes u^{\leq t})$ means r and $u^{\leq t}$ are drawn independently from their corresponding marginal distributions. But when r and $u^{\leq t}$ are drawn independently (i.e., the update requester is non-adaptive), $N^t \sim Bin(k,p)$ as we have shown in the first part of this theorem.

$$\Pr_{(r \otimes u^t)}\left[N^t \geq (1+\eta)\,\mu\right] = \Pr\left[Bin(k,p) \geq (1+\eta)\,\mu\right] \leq e^{-\frac{\eta^2\mu}{2+\eta}} \tag{9}$$

Therefore, combining Equations (8) and (9), we get that

$$\Pr\left[N^t \geq (1+\eta)\,\mu\right] \leq e^{\epsilon' - \frac{\eta^2\mu}{2+\eta}} + \delta'$$

which in turn implies, for every $\xi > \delta'$, with probability at least $1-\xi$,

$$N^t \leq \mu + \sqrt{2\mu\left(\epsilon' + \log\left(1/(\xi-\delta')\right)\right)} \tag{10}$$

Combining the bounds of Equations (7) and (10), we get that for every $\xi > \delta'$, with probability $1-\xi$,

$$N^t \leq \mu + \min\left\{\sqrt{2\mu\log\left(2(n+t)/(\xi-\delta')\right)}, \sqrt{2\mu\left(\epsilon' + 2\log\left(2/(\xi-\delta')\right)\right)}\right\}$$

which completes the proof by the choice of $p = 1/k$ ($\mu = kp = 1$). □

Lemma B.2. *Assume $\epsilon < 1$ and $\delta > 0$. Then, $(\hat{y}_1, \ldots, \hat{y}_l)$ is (ϵ,δ)-differentially private in $\{\theta_i\}_i$ where $\hat{y}_j = \texttt{PrivatePredict}^k_{\epsilon'}(\{\theta_i\}, x_j)$ and*

$$l = \left\lfloor \frac{\epsilon^2}{8(\epsilon')^2 \ln(\frac{1}{\delta})} \right\rfloor.$$

Proof. This claim holds immediately by the $(\epsilon', 0)$-differential privacy of $\texttt{PrivatePredict}^k_{\epsilon'}$ and the advanced composition theorem. See Corollary 3.21 in Dwork and Roth [2014] for details. □

Theorem 4.3. *The models $\{\{\theta^t_i\}_i\}_t$ in $\texttt{PrivatePredictionInteraction}(\epsilon', \epsilon, \delta, k)$ satisfy (α,β,γ)-unlearning guarantee for $\mathcal{A}^{distr}$ where $\alpha = O\left(\epsilon^2 k + k\sqrt{\delta/\epsilon}\right)$ and $\beta,\gamma = O\left(\sqrt{e^{-\epsilon^2 k} + k\sqrt{\delta/\epsilon}}\right)$, if $0 < \epsilon \leq 1/2$ and $0 < \delta < \epsilon$.*

Algorithm 4: `PrivatePredictionInteraction`$(\epsilon', \epsilon, \delta, k)$

$l = 0$
for $t = 1, \ldots, T$ **do**
  if $l > \left\lfloor \frac{\epsilon^2}{8(\epsilon')^2 \ln(\frac{1}{\delta})} \right\rfloor$ // "Restart" $\mathcal{R}^{\texttt{distr}}_{\mathcal{A}}$ when privacy budget is exhausted **then**
    $D^t_i = \texttt{Sampler}_i(D^t, p)$ and $\theta^t_i = \mathcal{A}^{\text{single}}_i(D^t_i)$ for each $i \in [k]$
    Update $s^t = (\{D^t_i\}_i, \{\theta^t_i\}_i)$
    $l = 0$
  else
    $\{\theta^t_i\}_i = \mathcal{R}_{\mathcal{A}^{\text{distr}}}(D^{t-1}, u^t, s^{t-1})$
  while there is a prediction request for some x **do**
    Publish $\hat{y} = \texttt{PrivatePredict}^k_{\epsilon'}(\{\theta^t_i\}_i, x)$
    $l = l + 1$

Algorithm 5: $\mathcal{A}^{\text{SISA}}$: Learning Algorithm for SISA

Proof. **Input**: dataset $D \equiv D^0$ of size n
Draw the shards: $D^0_{i\in[k]} = \texttt{RandomAssignPartition}(D^0, k)$.
Train the models: $\theta^0_{i\in[k]} = \mathcal{A}^{\text{single}}(D^0_i)$, for every $i \in [k]$.
Save the state: $s^0 = (\{D^0_i\}_{i\in[k]}, \{\theta^0_i\}_{i\in[k]})$
Output: $\{\theta^0_i\}_{i\in[k]}$

Proof. Suppose full retraining occurs in rounds $(t_1, t_2, \ldots, t_G)$ where we always have $t_1 = 0$ and $l > \left\lfloor \frac{\epsilon^2}{8(\epsilon')^2 \ln(\frac{1}{\delta})} \right\rfloor$ at round t_g for any $g > 1$.

At any round t_g when full retraining occurs, we can imagine restarting $\mathcal{R}^{\text{distr}}_{\mathcal{A}}$ by resetting the internal round as $t = 0$ and drawing fresh randomness $r \sim \mathcal{P}^k$, which determines the new initial state s^0. Therefore, for any $g \in [G-1]$ and $t_g \le t < t_{g+1}$, we must have that $\{f^{t'}_{\text{publish}}\}_{t_g \le t' \le t}$ are (ϵ, δ)-differentially private in the randomness r drawn in round t_g. Then, we can appeal to Theorem 4.1 to conclude that for any $g \in [G-1]$ and $t_g \le t < t_{g+1}$, we have

$$\forall E \subseteq \Theta^* : \quad \Pr\left[\{\theta^t_i\}_i \in E \,\middle|\, (u_{t_g}, \ldots, u_t)\right] \le e^\alpha \cdot \Pr\left[\mathcal{A}\left(D^t\right) \in E\right] + \beta.$$

Because we are redrawing fresh randomness $r \sim \mathcal{P}^k$ at t_g, we can combine combine all the previous unlearning guarantees in the previous $(t_{g'-1}, t_{g'})$ for $g' < g$ to conclude that at any round $t \in [T]$

$$\forall E \subseteq \Theta^* : \quad \Pr\left[\{\theta^t_i\}_i \in E \,\middle|\, u^{\le t}\right] \le e^\alpha \cdot \Pr\left[\mathcal{A}\left(D^t\right) \in E\right] + \beta.$$

□

C Details From Section 5

C.1 Proof of Theorem 5.1

Theorem 5.1. *There are learning and unlearning algorithms in the SISA framework $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ such that for any α, and any $\beta, \gamma < 1/4$, $\mathcal{R}_{\mathcal{A}}$ is not an (α, β, γ)-unlearning algorithm for $\mathcal{A}$.*

Define $\mathcal{A}^{\text{SISA}}$ and $\mathcal{R}_{\mathcal{A}^{\text{SISA}}}$ as Algorithms 5 and 6 respectively instantiated with the "lookup table" model $\mathcal{A}^{\text{single}}(D) = D$ and "lookup table" prediction rule f_θ. In Algorithm 5, $\texttt{RandomAssignPartition}(D, k)$ assigns

Algorithm 6: $\mathcal{R}_{\mathcal{A}^{\text{SISA}}}$: Unlearning Algorithm for SISA: t'th round of unlearning

Input: dataset D^{t-1}, update $u^t = (z^t, \bullet^t)$, state $s^{t-1} = (\{D_i^{t-1}\}_{i\in[k]}, \{\theta_i^{t-1}\}_{i\in[k]})$

if $\bullet^t = \texttt{'delete'}$ **then**

$i = j \in [k]$, where $z^t \in D_j^{t-1}$

else

$i = \texttt{randint}(1, 2, \ldots, k)$

Update the shards: $D_i^t = \begin{cases} D_j^{t-1} \circ u^t & \text{if } i = j \\ D_j^{t-1} & \text{otherwise} \end{cases}$, for every $j \in [k]$.

Update the models: $\theta_j^t = \begin{cases} \mathcal{A}^{\text{single}}(D_j^t) & \text{if } i = j \\ \theta_j^{t-1} & \text{otherwise} \end{cases}$, for every $i \in [k]$.

Update the state: $s^t = (\{D_i^t\}_{i\in[k]}, \{\theta_i^t\}_{j\in[k]})$

Output: $\{\theta_i^t\}_{i\in[k]}$

every $(x, y) \in D$ to one of the k partitions uniformly at random. The prediction rule, given parameter $\theta = D$ and query point x, outputs y if $(x, y) \in \theta$ and $\perp$ otherwise:

$$f_\theta(x) = \begin{cases} y & \text{if } (x, y) \in \theta, \\ \perp & \text{otherwise.} \end{cases}$$

We wish to show that there exists a dataset D^0 and adaptive update requester `UpdReq` such that for some update step $t \geq 1$, with probability at least $1 - \gamma$ over the draw of the update sequence $u^{\leq t} = (u^1, \ldots, u^t)$ from `UpdReq`, $\exists E \subseteq \Theta^* : \quad \Pr\left[\mathcal{R}_{\mathcal{A}}\left(D^{t-1}, u^t, s^{t-1}\right) \in E \,\middle|\, u^{\leq t}\right] > e^\alpha \cdot \Pr\left[\mathcal{A}\left(D^t\right) \in E\right] + \beta$. We prove this with the following example, instantiated for $k = 3$.

Consider dataset D^0 consisting of training examples $\{(x_i, y_i)\}_{i\in[2n]}$, $n \in \mathbb{Z}^+$ such that D^0 contains 2 copies each of n distinct feature vectors x. Both copies of each distinct feature vector x are paired with the same (arbitrary) label y.

Further, given ensemble model parameters $\{\theta_i\}_{i\in[k]} = \mathcal{A}^{\text{distr}}(D)$, let the ensemble output the mode of the predictions made by the underlying models:

$$\hat{y}_i = \texttt{Mode}\left(\left\{f_{\theta_j}(x_i)\right\}_{j\in[k]}\right).$$

Let ψ^0, the published object after initial training, be the ensemble's predictions for each training point: $\psi^0 = f^0_{\text{publish}} = (\hat{y}^0_1, \hat{y}^0_2, \ldots, \hat{y}^0_{2n})$.

Given these predictions, let $I = \{i_1, i_2, \ldots, i_t\} \subseteq [2n]$ be the indices for the points which were classified correctly. That is, $\forall i \in [2n] : i \in I$ if $y_i = \hat{y}_i$. Given ψ^0, let `UpdReq` be a function which outputs the deletion sequence $(u^1, u^2, \ldots, u^t)$ where each update request is responsible for deleting one of the correctly predicted points: $\forall j \in [t] : u^j = ((x_{i_j}, y_{i_j}), \texttt{'delete'})$.

Recall that our model is parameterized by a set of model parameters $\{\theta_i\}_{i\in[k]}$ and each θ_i is the dataset that shard is trained on. We now define the event E of interest: the set of all models such that the ensemble attains zero accuracy on the remaining points $D^t = D^0 \circ (u^1, u^2, \ldots, u^t)$, which happens if and only if all identical points (both copies of the same point) fall into the same shard.

$$E^t = \left\{\{\theta_i\}_{i\in[k]} \text{ where } |\{\theta_i : (x, y) \in \theta_i\}| = 1 \text{ for all } (x, y) \in D^t\right\}$$

To make our final assertion, first note that $\Pr[\mathcal{R}_{\mathcal{A}^{\text{SISA}}}(D^{t-1}, u^t, s^{t-1}) \in E | u^{\leq t}] = 1$ as `UpdReq` has requested all the correctly classified points to be deleted. We therefore need to show that

$$\Pr\left[\mathcal{R}_{\mathcal{A}^{\text{SISA}}}\left(D^{t-1}, u^t, s^{t-1}\right) \in E \,\middle|\, u^{\leq t}\right] = 1 > e^\alpha \cdot \Pr\left[\mathcal{A}^{\text{SISA}}\left(D^t\right) \in E\right] + \beta$$

equivalently, $\frac{1-\beta}{e^\alpha} > \Pr\left[\mathcal{A}^{\text{SISA}}\left(D^t\right) \in E\right]$ with probability $1-\gamma$ over the randomness of the update sequence (which in this case is simply the randomness of the initial partition).

Note that t, the number of copies of points that were initially classified correctly is distributed as $\texttt{Binomial}(n, \frac{2}{3})$ because for each pair of identical $(x, y) \in D^0$, the probability that they fall in different shards initially is exactly 2/3. Also, note that for any fixed $t \leq n-1$,

$$\Pr[\mathcal{A}^{\text{distr}}(D^t) \in E] = \frac{1}{3^{n-t}}.$$

Using the tail bound for the Binomial distribution (Fact B.1), we have that with probability $1-\gamma$,

$$t \leq \frac{2n}{3} + \sqrt{\frac{4n}{3}\log\left(\frac{1}{\gamma}\right)}.$$

When $n \geq 13\log(1/\gamma)$, we have $\frac{2n}{3} + \sqrt{\frac{4n}{3}\log\left(\frac{1}{\gamma}\right)} \leq 0.99n$. Hence, for sufficiently large n, we can conclude that with probability $1-\gamma$,

$$\Pr[\mathcal{A}^{\text{distr}}(D^t) \in E] \leq \frac{1}{3^{0.01n}}.$$

Finally, for any $c = \frac{1-\beta}{e^\alpha} > 0$, there exists a D^0 such that $c > \Pr\left[\mathcal{A}^{\text{SISA}}\left(D^t\right) \in E\right]$ with probability $1-\gamma$ because we can choose a sufficiently large n such that $n \geq 13\log(1/\gamma)$ and $\frac{1}{3^{0.01n}} \leq c$, i.e., we can choose:

$$n \geq \max\left\{13\log(1/\gamma), \frac{100\log(1/c)}{\log 3}\right\}$$

□

C.2 Failures in (0, 0, 0)-Unlearning Beyond Section 5.1

Observable failures in unlearning guarantees for algorithms in the SISA framework go beyond the simplistic setting constructed in Section 5.1. In this section, we describe a more natural setting in which we employ the learning and unlearning algorithms for SISA $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ and are able to construct an adaptive deletion sequence (only given discrete predictions through f_{publish}) which, to a high degree of confidence, rejects the null hypothesis that $(\mathcal{A}, \mathcal{R}_{\mathcal{A}})$ satisfy a perfect $(0, 0, 0)$-unlearning guarantee.

In Section 5.1 we explicitly define a base model f_θ which relies on the fact that each point is copied twice to reveal perfect information about how points were partitioned through its predictions. Here, we define a new model which relaxes this condition. Given a query point x, rather than return the label of an exactly matching point, the model $f_{D,\tau}(x)$ is additionally parameterized by a threshold τ. This model, reminiscent of 1-nearest neighbors, returns the label of the closest point $(x', y') \in D$ where $|x - x'|_2 \leq \tau$, and $\perp$ otherwise, essentially treating nearby points as "identical."

Here we define $\mathcal{A}^{\text{SISA}}$ and $\mathcal{R}_{\mathcal{A}^{\text{SISA}}}$ as Algorithms 5 and 6 respectively, instantiated with $\mathcal{A}^{\text{single}}(D) = D$ and prediction rule $f_{D,\tau}$. We assume the null hypothesis that $\mathcal{A}$ and $\mathcal{R}_{\mathcal{A}}$ satisfy a $(0, 0, 0)$-unlearning guarantee.

To make an assertion about this hypothesis, we train an ensemble using three shards as before. We then execute a similar experiment to that as described in Section 5.1 in which, after initial training, we publish the aggregated discrete predictions for each training point and delete a random subset of correctly classified points. We then observe the accuracy of the ensemble on the remaining training points. Our hypothesis, the same as before, is that the resulting accuracy will be lower in the adaptive deletion setting than the retrain setting with high probability.

We then define an event E of interest to be when the training accuracy after the adaptive deletion sequence falls below a cutoff $c \in [0\%, 100\%]$ after deleting all correctly classified points. We can then estimate the probability of this event by defining an indicator for each trial which is 1 if the training accuracy falls below

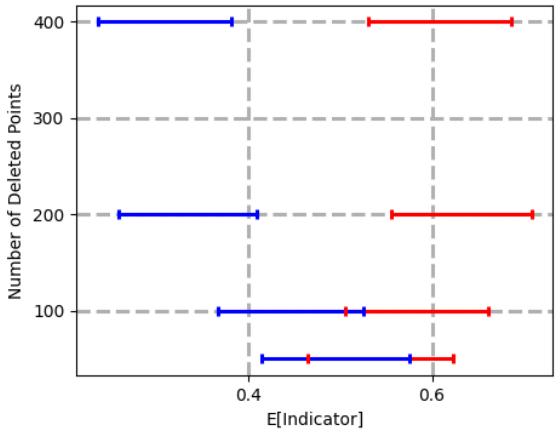


Figure 2: Confidence intervals for the indicator defined in Section C.2 as a function of the number of deleted points. Red confidence intervals correspond to the statistic after the adaptive deletion sequence, and blue confidence intervals correspond to the statistic after full retraining. For deletion sequences of 200 points or more we can induce a reliable enough difference in the confidence intervals to reject the null hypothesis at $p \leq 0.05$ that $\mathcal{A}^{\text{SISA}}$ and $\mathcal{R}_{\mathcal{A}^{\text{SISA}}}$ satisfy a perfect $(0, 0, 0)$-unlearning guarantee in a realistic setting.

this threshold and 0 otherwise. We then run many trials to calculate confidence intervals on our estimate of this probability under either setting. If the confidence intervals are non-overlapping at some confidence level, we can then reject the null hypothesis at some level of confidence.

Our concrete experiment samples 1,000 random points from MNIST, each being either a "0" or "1" (preprocessing each image by dividing each pixel value by 255). With $\tau = 6.5$, this setting is "plausible" in the sense that this model's performance on held-out test data is nontrivial (approximately 91.2% test accuracy before deletion) for a common benchmark task. We then delete t points (a uniformly random subset of correctly predicted points), observe the average accuracies across trials on remaining points under the adaptive setting and the retrain scenario. We grid search for the c which yields the largest difference in the confidence intervals (since the unlearning guarantee should hold for all c). Under these conditions we find that after 200 trials, we attain 97.5% confidence intervals on our statistic to be those shown in Figure 2. We see that for deletion sequences of 200 points or more we can induce a reliable difference in this statistic at a high level of confidence, rejecting the null hypothesis at $p \leq 0.05$ that $\mathcal{A}^{\text{SISA}}$ and $\mathcal{R}_{\mathcal{A}^{\text{SISA}}}$ satisfy a perfect $(0, 0, 0)$-unlearning guarantee.

C.3 Full Experiment Details of Section 5.2

Choices in hyperparameters and and model architecture for experiments presented in Section 5 were inspired by those used by Papernot et al. [2021]. All models were optimized using momentum with mass equal to 0.9. The clipping parameter (upper bound on maximum ℓ_2-norm of per-example gradients) used in DP-SGD for all experiments was equal to 0.1. For certain experiments, the batch size was reduced from what was presented in Papernot et al. [2021] to reduce computational cost. Each experiment was repeated with new random seeds across 300 trials to get the confidence intervals displayed in Figure 1. The precise model definition for each experiment is given below:

```
Sequential(
  Conv(out_chan=16, filter_shape=(8, 8), padding='SAME', strides=(2, 2)),
  Tanh,
  MaxPool(window_shape=(2, 2), strides=(1, 1)),
```

```
  Conv(out_chan=32, filter_shape=(4, 4), padding='VALID', strides=(2, 2)),
  Tanh,
  MaxPool(window_shape=(2, 2), strides=(1, 1)),
  Flatten,
  Dense(out_dim=32),
  Tanh,
  Dense(out_dim=num_classes)
)
```

In our experiments we make use of 3 common benchmark machine learning datasets. The MNIST database of handwritten digits given by Lecun et al. [1998] consists of 70,000 28×28 images of handwritten digits, each belonging to one of 10 classes characterizing the digit shown in each image. MNIST is made available under the Creative Commons Attribution-Share Alike 3.0 license. The Fashion-MNIST dataset given by Xiao et al. [2017] consists of 70,000 28×28 grayscale images of pieces of clothing, each belonging to one of 10 classes (e.g. t-shirt, dress, sneaker, etc.) Fashion-MNIST is made available under the MIT license. The CIFAR-10 dataset given by Krizhevsky and Hinton [2009] consists of 60,000 32×32 images in RGB format, each belonging to one of 10 classes characterizing the class of the object given in each image (e.g. airplane, automobile, bird, etc.) CIFAR-10 is made available under the MIT license.

With respect to computing environment, experiments were conducted using the JAX deep learning framework developed by Bradbury et al. [2018]. Experiments were run using 1 Tesla V100 GPU using CUDA version 11.0, where an individual trial (training a full ensemble, deleting targeted points, and retraining) would take approximately 1-6 minutes depending on the number of shards, iterations, image size, etc.

$\mathbb{E}$[Indicator]	Acc. (after)	Acc. (before)	Noise mult.	Shard pred. acc.
CIFAR-10$_{k=6}$				
$[0.890, 0.952]$	0.507 ± 0.025	0.572 ± 0.011	0	0.303 ± 0.005
$[0.784, 0.869]$	0.504 ± 0.020	0.554 ± 0.010	0.15	0.257 ± 0.004
$[0.670, 0.772]$	0.487 ± 0.021	0.525 ± 0.011	0.22	0.229 ± 0.003
$[0.490, 0.603]$	0.455 ± 0.025	0.484 ± 0.012	0.3	0.205 ± 0.003
CIFAR-10$_{k=2}$				
$[0.947, 0.987]$	0.448 ± 0.033	0.521 ± 0.019	0	0.655 ± 0.012
$[0.797, 0.881]$	0.433 ± 0.026	0.475 ± 0.015	0.2	0.587 ± 0.007
$[0.638, 0.744]$	0.419 ± 0.025	0.452 ± 0.015	0.25	0.567 ± 0.006
$[0.493, 0.607]$	0.399 ± 0.027	0.427 ± 0.015	0.3	0.550 ± 0.005
Fashion-MNIST$_{k=6}$				
$[0.819, 0.899]$	0.849 ± 0.011	0.874 ± 0.004	0	0.248 ± 0.007
$[0.662, 0.765]$	0.838 ± 0.011	0.854 ± 0.005	0.4	0.215 ± 0.005
$[0.540, 0.652]$	0.823 ± 0.009	0.834 ± 0.006	0.6	0.198 ± 0.004
$[0.477, 0.590]$	0.810 ± 0.012	0.820 ± 0.006	0.75	0.190 ± 0.004
Fashion-MNIST$_{k=2}$				
$[0.976, 0.999]$	0.826 ± 0.016	0.863 ± 0.006	0	0.597 ± 0.014
$[0.797, 0.881]$	0.808 ± 0.013	0.828 ± 0.007	0.5	0.555 ± 0.010
$[0.607, 0.715]$	0.791 ± 0.016	0.807 ± 0.008	0.7	0.538 ± 0.007
$[0.497, 0.610]$	0.763 ± 0.020	0.781 ± 0.009	1	0.523 ± 0.005
MNIST$_{k=6}$				
$[0.849, 0.922]$	0.973 ± 0.004	0.978 ± 0.002	0	0.201 ± 0.004
$[0.729, 0.824]$	0.965 ± 0.005	0.969 ± 0.003	0.4	0.186 ± 0.003
$[0.583, 0.694]$	0.940 ± 0.009	0.945 ± 0.004	0.8	0.178 ± 0.003
$[0.493, 0.607]$	0.913 ± 0.018	0.923 ± 0.007	1.1	0.176 ± 0.003
MNIST$_{k=2}$				
$[0.927, 0.976]$	0.962 ± 0.007	0.971 ± 0.003	0	0.540 ± 0.006
$[0.769, 0.857]$	0.959 ± 0.008	0.968 ± 0.003	0.8	0.534 ± 0.006
$[0.587, 0.697]$	0.953 ± 0.007	0.962 ± 0.004	1.3	0.530 ± 0.006
$[0.497, 0.610]$	0.949 ± 0.008	0.957 ± 0.004	1.6	0.527 ± 0.006

Table 1: Numerical representation of results displayed in Figure 1. The x axis in Figure 1 corresponds to column "$\mathbb{E}$[Indicator]", and the y axis corresponds to column "Acc. (after)". Column "$\mathbb{E}$[Indicator]" represents the 95% confidence interval of the indicator after 300 trials. Columns "Acc. (before)" and "Acc. (after)" represent the accuracy of the ensemble on a held-out test set (5,000 points each) before and after deleting approximately half of the points from the ensemble, with confidence intervals given by two standard deviations above and below the observed mean. "Noise multiplier" represents the standard deviation of Gaussian noise applied to each per-example gradient during DP-SGD. Shard prediction accuracy denotes the prediction accuracy of the adversary in targeting models when deleting points, where random guessing would achieve an accuracy of 1/(# shards).

Experiment	Points per shard	Batch Size	Iterations	Step size
CIFAR-10$_{k=6}$	8000	64	4000	1.0
CIFAR-10$_{k=2}$	8000	64	4000	1.0
Fashion-MNIST$_{k=6}$	6000	256	1500	4.0
Fashion-MNIST$_{k=2}$	6000	256	2000	4.0
MNIST$_{k=6}$	6000	64	2500	0.5
MNIST$_{k=2}$	6000	256	2000	0.5

Table 2: Remaining hyperparameter settings for each experiment, by dataset.