Certified Data Removal from Machine Learning Models

Chuan Guo [1] **Tom Goldstein** [2] **Awni Hannun** [2] **Laurens van der Maaten** [2]

Abstract

Good data stewardship requires removal of data at the request of the data's owner. This raises the question if and how a trained machine-learning model, which implicitly stores information about its training data, should be affected by such a removal request. Is it possible to "remove" data from a machine-learning model? We study this problem by defining *certified removal*: a very strong theoretical guarantee that a model from which data is removed cannot be distinguished from a model that never observed the data to begin with. We develop a certified-removal mechanism for linear classifiers and empirically study learning settings in which this mechanism is practical.

1. Introduction

Machine-learning models are often trained on third-party data, for example, many computer-vision models are trained on images provided by Flickr users (Thomee et al., 2016). When a party requests that their data be removed from such online platforms, this raises the question how such a request should impact models that were trained prior to the removal. A similar question arises when a model is negatively impacted by a data-poisoning attack (Biggio et al., 2012). Is it possible to "remove" data from a model without re-training that model from scratch?

We study this question in a framework we call *certified removal*, which theoretically guarantees that an adversary cannot extract information about training data that was removed from a model. Inspired by differential privacy (Dwork, 2011), certified removal bounds the max-divergence between the model trained on the dataset with some instances removed and the model trained on the dataset that never contained those instances. This guarantees that membership-inference attacks (Yeom et al., 2018; Carlini et al., 2019) are unsuccessful on data that was removed from the model. We emphasize that certified removal is a very strong notion of removal; in practical applications, less constraining notions may equally fulfill the data owner's expectation of removal.

We develop a certified-removal mechanism for L_2-regularized linear models that are trained using a differentiable convex loss function, *e.g.*, logistic regressors. Our removal mechanism applies a Newton step on the model parameters that largely removes the influence of the deleted data point; the residual error of this mechanism decreases quadratically with the size of the training set. To ensure that an adversary cannot extract information from the small residual (*i.e.*, to certify removal), we mask the residual using an approach that randomly perturbs the training loss (Chaudhuri et al., 2011). We empirically study in which settings the removal mechanism is practical.

2. Certified Removal

Let $\mathcal{D}$ be a fixed training dataset and let A be a (randomized) learning algorithm that trains on $\mathcal{D}$ and outputs a model $h \in \mathcal{H}$, that is, $A : \mathcal{D} \rightarrow \mathcal{H}$. Randomness in A induces a probability distribution over the models in the hypothesis set $\mathcal{H}$. We would like to remove a training sample, $\mathbf{x} \in \mathcal{D}$, from the output of A.

To this end, we define a data-removal mechanism M that is applied to $A(\mathcal{D})$ and aims to remove the influence of $\mathbf{x}$. If removal is successful, the output of M should be difficult to distinguish from the output of A applied on $\mathcal{D} \setminus \mathbf{x}$. Given $\epsilon > 0$, we say that removal mechanism M performs ϵ-*certified removal* (ϵ-CR) for learning algorithm A if $\forall \mathcal{T} \subseteq \mathcal{H}, \mathcal{D} \subseteq \mathcal{X}, \mathbf{x} \in \mathcal{D}$:

$$e^{-\epsilon} \leq \frac{P(M(A(\mathcal{D}), \mathcal{D}, \mathbf{x}) \in \mathcal{T})}{P(A(\mathcal{D} \setminus \mathbf{x}) \in \mathcal{T})} \leq e^{\epsilon}. \qquad (1)$$

This definition states that the ratio between the likelihood of (i) a model from which sample $\mathbf{x}$ was removed and (ii) a model that was never trained on $\mathbf{x}$ to begin with is close to one for all models in the hypothesis set, for all possible data sets, and for all removed samples. Note that although the definition requires that the mechanism M is universally applicable to all training data points $\mathbf{x} \in \mathcal{D}$, it is also allowed to be data-dependent, *i.e.*, both the training set $\mathcal{D}$ and the

[1]Department of Computer Science, Cornell University, New York, USA [2]Facebook AI Research, New York, USA. Correspondence to: Chuan Guo <cg563@cornell.edu>, Laurens van der Maaten <lvdmaaten@gmail.com>.

Proceedings of the 37th International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

data point to be removed $\mathbf{x}$ are given as inputs to M.

We also define a more relaxed notion of (ϵ, δ)-*certified removal* for $\delta > 0$ if $\forall \mathcal{T} \subseteq \mathcal{H}, \mathcal{D} \subseteq \mathcal{X}, \mathbf{x} \in \mathcal{D}$:

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

Conceptually, δ upper bounds the probability for the max-divergence bound in Equation 1 to fail.

A trivial certified-removal mechanism M with $\epsilon = 0$ completely ignores $A(\mathcal{D})$ and evaluates $A(\mathcal{D} \setminus \mathbf{x})$ directly, that is, it sets $M(A(\mathcal{D}), \mathcal{D}, \mathbf{x}) = A(\mathcal{D} \setminus \mathbf{x})$. Such a removal mechanism is impractical for many models, as it may require re-training the model from scratch every time a training sample is removed. We seek mechanisms M with removal cost $O(n)$ or less, with small constants, where $n = |\mathcal{D}|$ is the training set size.

Insufficiency of parametric indistinguishability. One alternative relaxation of exact removal is by asserting that the output of $M(A(\mathcal{D}), \mathcal{D}, \mathbf{x})$ is sufficiently close to that of $A(\mathcal{D} \setminus \mathbf{x})$. It is easy to see that a model satsifying such a definition still retains information about $\mathbf{x}$. Consider a linear regressor trained on the dataset $\mathcal{D} = \{(\mathbf{e}_1, 1), (\mathbf{e}_2, 2), \ldots, (\mathbf{e}_d, d)\}$ where $\mathbf{e}_i$'s are the standard basis vectors for $\mathbb{R}^d$. A regressor that is initialized with zeros, or that has a weight decay penalty, will place a non-zero weight on $\mathbf{w}_i$ if $(\mathbf{e}_i, i)$ is included in $\mathcal{D}$, and a zero weight on $\mathbf{w}_i$ if not. In this case, an approximate removal algorithm that leaves $\mathbf{w}_i$ small but non-zero still reveals that $(\mathbf{e}_i, i)$ appeared during training.

Relationship to differential privacy. Our formulation of certified removal is related to that of differential privacy (Dwork, 2011) but there are important differences. Differential privacy states that:

$$\forall \mathcal{T} \subseteq \mathcal{H}, \mathcal{D}, \mathcal{D}' : e^{-\epsilon} \leq \frac{P(A(\mathcal{D}) \in \mathcal{T})}{P(A(\mathcal{D}') \in \mathcal{T})} \leq e^{\epsilon}, \qquad (2)$$

where $\mathcal{D}$ and $\mathcal{D}'$ differ in only one sample. Since $\mathcal{D}$ and $\mathcal{D} \setminus \mathbf{x}$ only differ in one sample, it is straightforward to see that differential privacy of A is a sufficient condition for certified removal, *viz.*, by setting removal mechanism M to the identity function. Indeed, if algorithm A never memorizes the training data in the first place, we need not worry about removing that data.

Even though differential privacy is a sufficient condition, it is not a necessary condition for certified removal. For example, a nearest-neighbor classifier is not differentially private but it is trivial to certifiably remove a training sample in $O(1)$ time with $\epsilon = 0$. We note that differential privacy is a very strong condition, and most differentially private models suffer a significant loss in accuracy even for large ϵ (Chaudhuri et al., 2011; Abadi et al., 2016). We therefore view the study of certified removal as analyzing the trade-off between utility and removal efficiency, with re-training from scratch and differential privacy at the two ends of the spectrum, and removal in the middle.

3. Removal Mechanisms

We focus on certified removal from parametric models, as removal from non-parametric models (*e.g.,* nearest-neighbor classifiers) is trivial. We first study linear models with strongly convex regularization before proceeding to removal from deep networks.

3.1. Linear Classifiers

Denote by $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ the training set of n samples, $\forall i : \mathbf{x}_i \in \mathbb{R}^d$, with corresponding targets y_i. We assume learning algorithm A tries to minimize the regularized empirical risk of a linear model:

$$L(\mathbf{w}; \mathcal{D}) = \sum_{i=1}^{n} \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \frac{\lambda n}{2} \|\mathbf{w}\|_2^2,$$

where $\ell(\mathbf{w}^\top \mathbf{x}, y)$ is a convex loss that is differentiable everywhere. We denote $\mathbf{w}^* = A(\mathcal{D}) = \text{argmin}_{\mathbf{w}} L(\mathbf{w}; \mathcal{D})$ as it is uniquely determined.

Our approach to certified removal is as follows. We first define a removal mechanism that, given a training data point $(\mathbf{x}, y)$ to remove, produces a model $\mathbf{w}^-$ that is approximately equal to the unique minimizer of $L(\mathbf{w}; \mathcal{D} \setminus (\mathbf{x}, y))$. The model produced by such a mechanism may still contain information about $(\mathbf{x}, y)$. In particular, if the gradient $\nabla L(\mathbf{w}^-; \mathcal{D} \setminus (\mathbf{x}, y))$ is non-zero, it reveals information about the removed data point. To hide this information, we apply a sufficiently large random perturbation to the model parameters at training time. This allows us to guarantee certified removal; the values of ϵ and δ depend on the approximation quality of the removal mechanism and on the distribution from which the model perturbation is sampled.

Removal mechanism. We assume without loss of generality that we aim to remove the last training sample, $(\mathbf{x}_n, y_n)$. Specifically, we define an efficient removal mechanism that approximately minimizes $L(\mathbf{w}; \mathcal{D}')$ with $\mathcal{D}' = \mathcal{D} \setminus (\mathbf{x}_n, y_n)$. First, denote the loss gradient at sample $(\mathbf{x}_n, y_n)$ by $\Delta = \lambda \mathbf{w}^* + \nabla \ell((\mathbf{w}^*)^\top \mathbf{x}_n, y_n)$ and the Hessian of $L(\cdot; \mathcal{D}')$ at $\mathbf{w}^*$ by $H_{\mathbf{w}^*} = \nabla^2 L(\mathbf{w}^*; \mathcal{D}')$. We consider the *Newton update removal mechanism* M:

$$\mathbf{w}^- = M(\mathbf{w}^*, \mathcal{D}, (\mathbf{x}_n, y_n)) := \mathbf{w}^* + H_{\mathbf{w}^*}^{-1} \Delta, \qquad (3)$$

which is a one-step Newton update applied to the gradient influence of the removed point $(\mathbf{x}_n, y_n)$. The update $H_{\mathbf{w}^*}^{-1} \Delta$

is also known as the influence function of the training point $(\mathbf{x}_n, y_n)$ on the parameter vector $\mathbf{w}^*$ (Cook and Weisberg, 1982; Koh and Liang, 2017).

The computational cost of this Newton update is dominated by the cost of forming and inverting the Hessian matrix. The Hessian matrix for $\mathcal{D}$ can be formed offline with $O(d^2n)$ cost. The subsequent Hessian inversion makes the removal mechanism $O(d^3)$ at removal time; the inversion can leverage efficient linear algebra libraries and GPUs.

To bound the approximation error of this removal mechanism, we observe that the quantity $\nabla L(\mathbf{w}^-; \mathcal{D}')$, which we refer to hereafter as the *gradient residual*, is zero only when $\mathbf{w}^-$ is the unique minimizer of $L(\cdot; \mathcal{D}')$. We also observe that the gradient residual norm, $\|\nabla L(\mathbf{w}^-; \mathcal{D}')\|_2$, reflects the error in the approximation $\mathbf{w}^-$ of the minimizer of $L(\cdot; \mathcal{D}')$. We derive an upper bound on the gradient residual norm for the removal mechanism (*cf.* Equation 3).

Theorem 1. *Suppose that $\forall (\mathbf{x}_i, y_i) \in \mathcal{D}, \mathbf{w} \in \mathbb{R}^d$: $\|\nabla \ell(\mathbf{w}^\top \mathbf{x}_i, y_i)\|_2 \leq C$. Suppose also that ℓ'' is γ-Lipschitz and $\|\mathbf{x}_i\|_2 \leq 1$ for all $(\mathbf{x}_i, y_i) \in \mathcal{D}$. Then:*

$$\begin{aligned} \|\nabla L(\mathbf{w}^-; \mathcal{D}')\|_2 &= \|(H_{\mathbf{w}_\eta} - H_{\mathbf{w}^*})H_{\mathbf{w}^*}^{-1}\Delta\|_2 \qquad (4) \\ &\leq \gamma(n-1)\|H_{\mathbf{w}^*}^{-1}\Delta\|_2^2 \leq \frac{4\gamma C^2}{\lambda^2(n-1)}, \end{aligned}$$

where $H_{\mathbf{w}_\eta}$ denotes the Hessian of $L(\cdot; \mathcal{D}')$ at the parameter vector $\mathbf{w}_\eta = \mathbf{w}^* + \eta H_{\mathbf{w}^*}^{-1}\Delta$ for some $\eta \in [0, 1]$.

Loss perturbation. Obtaining a small gradient norm $\|\nabla L(\mathbf{w}^-; \mathcal{D}')\|_2$ via Theorem 1 does not guarantee certified removal. In particular, the direction of the gradient residual may leak information about the training sample that was "removed." To hide this information, we use the loss perturbation technique of Chaudhuri et al. (2011) at training time. It perturbs the empirical risk by a random linear term:

$$L_{\mathbf{b}}(\mathbf{w}; \mathcal{D}) = \sum_{i=1}^{n} \ell\left(\mathbf{w}^\top \mathbf{x}_i, y_i\right) + \frac{\lambda n}{2}\|\mathbf{w}\|_2^2 + \mathbf{b}^\top \mathbf{w},$$

with $\mathbf{b} \in \mathbb{R}^d$ drawn randomly from some distribution. We analyze how loss perturbation masks the information in the gradient residual $\nabla L_{\mathbf{b}}(\mathbf{w}^-; \mathcal{D}')$ through randomness in $\mathbf{b}$.

Let $A(\mathcal{D}')$ be an exact minimizer[1] for $L_{\mathbf{b}}(\cdot; \mathcal{D}')$ and let $\tilde{A}(\mathcal{D}')$ be an approximate minimizer of $L_{\mathbf{b}}(\cdot; \mathcal{D}')$, for example, our removal mechanism applied on the trained model. Specifically, let $\tilde{\mathbf{w}}$ be an approximate solution produced by $\tilde{A}$. This implies the gradient residual is:

$$\mathbf{u} := \nabla L_{\mathbf{b}}(\tilde{\mathbf{w}}; \mathcal{D}') = \sum_{i=1}^{n-1} \nabla \ell(\tilde{\mathbf{w}}^\top \mathbf{x}_i, y_i) + \lambda(n-1)\tilde{\mathbf{w}} + \mathbf{b}. \qquad (5)$$

We assume that $\tilde{A}$ can produce a gradient residual $\mathbf{u}$ with $\|\mathbf{u}\|_2 \leq \epsilon'$ for some pre-specified bound ϵ' that is *independent* of the perturbation vector $\mathbf{b}$.

Let $f_A(\cdot)$ and $f_{\tilde{A}}(\cdot)$ be the density functions over the model parameters produced by A and $\tilde{A}$, respectively. We bound the max-divergence between f_A and $f_{\tilde{A}}$ for any solution $\tilde{\mathbf{w}}$ produced by approximate minimizer $\tilde{A}$.

Theorem 2. *Suppose that $\mathbf{b}$ is drawn from a distribution with density function $p(\cdot)$ such that for any $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^d$ satisfying $\|\mathbf{b}_1 - \mathbf{b}_2\|_2 \leq \epsilon'$, we have that: $e^{-\epsilon} \leq \frac{p(\mathbf{b}_1)}{p(\mathbf{b}_2)} \leq e^{\epsilon}$. Then $e^{-\epsilon} \leq \frac{f_{\tilde{A}}(\tilde{\mathbf{w}})}{f_A(\tilde{\mathbf{w}})} \leq e^{\epsilon}$ for any $\tilde{\mathbf{w}}$ produced by $\tilde{A}$.*

Achieving certified removal. We can use Theorem 2 to prove certified removal by combining it with the gradient residual norm bound ϵ' from Theorem 1. The security parameters ϵ and δ depend on the distribution from which $\mathbf{b}$ is sampled. We state the guarantee of (ϵ, δ)-certified removal below for two suitable distributions $p(\mathbf{b})$.

Theorem 3. *Let A be the learning algorithm that returns the unique optimum of the loss $L_{\mathbf{b}}(\mathbf{w}; \mathcal{D})$ and let M be the Newton update removal mechanism (cf., Equation 3). Suppose that $\|\nabla L(\mathbf{w}^-; \mathcal{D}')\|_2 \leq \epsilon'$ for some computable bound $\epsilon' > 0$. We have the following guarantees for M:*

(i) If $\mathbf{b}$ is drawn from a distribution with density $p(\mathbf{b}) \propto e^{-\frac{\epsilon}{\epsilon'}\|\mathbf{b}\|_2}$, then M is ϵ-CR for A;
(ii) If $\mathbf{b} \sim \mathcal{N}(0, c\epsilon'/\epsilon)^d$ with $c > 0$, then M is (ϵ, δ)-CR for A with $\delta = 1.5 \cdot e^{-c^2/2}$.

The distribution in (i) is equivalent to sampling a direction uniformly over the unit sphere and sampling a norm from the $\Gamma(d, \frac{\epsilon'}{\epsilon})$ distribution (Chaudhuri et al., 2011).

3.2. Practical Considerations

Least-squares and logistic regression. The certified removal mechanism described above can be used with least-squares and logistic regression, which are ubiquitous in real-world applications of machine learning.

Least-squares regression assumes $\forall i : y_i \in \mathbb{R}$ and uses the loss function $\ell(\mathbf{w}^\top \mathbf{x}_i, y_i) = \left(\mathbf{w}^\top \mathbf{x}_i - y_i\right)^2$. The Hessian of this loss function is $\nabla^2 \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) = \mathbf{x}_i \mathbf{x}_i^\top$, which is independent of $\mathbf{w}$. In particular, the gradient residual from Equation 4 is exactly zero, which makes the Newton update in Equation 3 an ϵ-certified removal mechanism with $\epsilon = 0$

[1]Our result can be modified to work with approximate loss minimizers by incurring a small additional error term.

(loss perturbation is not needed). This is not surprising since the Newton update assumes a local quadratic approximation of the loss, which is exact for least-squares regression.

Logistic regression assumes $\forall i : y_i \in \{-1, +1\}$ and uses the loss function $\ell(\mathbf{w}^\top \mathbf{x}_i, y_i) = -\log \sigma\left(y_i \mathbf{w}^\top \mathbf{x}_i\right)$, where $\sigma(\cdot)$ denotes the sigmoid function, $\sigma(x) = \frac{1}{1+\exp(-x)}$. The loss gradient and Hessian are given by:

$$\begin{aligned}\nabla \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) &= \left(\sigma(y_i \mathbf{w}^\top \mathbf{x}_i) - 1\right) y_i \mathbf{x}_i \\ \nabla^2 \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) &= \sigma(y_i \mathbf{w}^\top \mathbf{x}_i)\left(1 - \sigma(y_i \mathbf{w}^\top \mathbf{x}_i)\right) \mathbf{x}_i \mathbf{x}_i^\top.\end{aligned}$$

Under the assumption that $\|\mathbf{x}_i\|_2 \leq 1$ for all i, it is straightforward to show that $\|\nabla \ell(\mathbf{w}^\top \mathbf{x}_i, y_i)\|_2 \leq 1$ and that $\ell''(\mathbf{w}^\top \mathbf{x}_i, y_i)$ is γ-Lipschitz with $\gamma = 1/4$. This allows us to apply Theorem 1 to logistic regression.

Data-dependent bound on gradient norm. The bound in Theorem 1 contains a constant factor of $1/\lambda^2$ and may be too loose for practical applications on smaller datasets. Fortunately, we can derive a *data-dependent* bound on the gradient residual that can be efficiently computed and that is much tighter in practice. Recall that the Hessian of $L(\cdot; \mathcal{D}')$ has the form:

$$H_{\mathbf{w}} = (X^-)^\top D_{\mathbf{w}} X^- + \lambda(n-1) I_d,$$

where $X^- \in \mathbb{R}^{(n-1)\times d}$ is the data matrix corresponding to $\mathcal{D}'$, I_d is the identity matrix of size $d \times d$, and $D_{\mathbf{w}}$ is a diagonal matrix containing values:

$$(D_{\mathbf{w}})_{ii} = \ell''\left(\mathbf{w}^\top \mathbf{x}_i, y_i\right).$$

By Equation 4 we have that:

$$\begin{aligned}\|\nabla L(\mathbf{w}^-; \mathcal{D}')\|_2 &= \|(H_{\mathbf{w}_\eta} - H_{\mathbf{w}^*}) H_{\mathbf{w}^*}^{-1} \Delta\|_2 \\ &= \|(X^-)^\top (D_{\mathbf{w}_\eta} - D_{\mathbf{w}^*}) X^- H_{\mathbf{w}^*}^{-1} \Delta\|_2 \\ &\leq \|X^-\|_2 \|D_{\mathbf{w}_\eta} - D_{\mathbf{w}^*}\|_2 \|X^- H_{\mathbf{w}^*}^{-1} \Delta\|_2.\end{aligned}$$

The term $\|D_{\mathbf{w}_\eta} - D_{\mathbf{w}^*}\|_2$ corresponds to the maximum singular value of a diagonal matrix, which in turn is the maximum value among its diagonal entries. Given the Lipschitz constant γ of the second derivative ℓ'', we can thus bound it as:

$$\|D_{\mathbf{w}_\eta} - D_{\mathbf{w}^*}\|_2 \leq \gamma \|\mathbf{w}_\eta - \mathbf{w}^*\|_2 \leq \gamma \|H_{\mathbf{w}^*}^{-1} \Delta\|_2.$$

The following corollary summarizes this derivation.

Corollary 1. *The Newton update* $\mathbf{w}^- = \mathbf{w}^* + H_{\mathbf{w}^*}^{-1} \Delta$ *satisfies:*

$$\|\nabla L(\mathbf{w}^-; \mathcal{D}')\|_2 \leq \gamma \|X^-\|_2 \|H_{\mathbf{w}^*}^{-1} \Delta\|_2 \|X^- H_{\mathbf{w}^*}^{-1} \Delta\|_2,$$

where γ *is the Lipschitz constant of* ℓ''.

The bound in Corollary 1 can be easily computed from the Newton update $H_{\mathbf{w}^*}^{-1} \Delta$ and the spectral norm of X^-, the latter admitting efficient algorithms such as power iterations.

Multiple removals. The worst-case gradient residual norm after T removals can be shown to scale linearly in T. We can prove this using induction on T. The base case, $T = 1$, is proven above. Suppose that the gradient residual after $T \geq 1$ removals is $\mathbf{u}_T$ with $\|\mathbf{u}_T\|_2 \leq T\epsilon'$, where ϵ' is the gradient residual norm bound for a single removal. Let $\mathcal{D}^{(T)}$ be the training dataset with T data points removed. Consider the modified loss function $L_{\mathbf{b}}^{(T)}(\mathbf{w}; \mathcal{D}^{(T)}) = L_{\mathbf{b}}(\mathbf{w}; \mathcal{D}^{(T)}) - \mathbf{u}_T^\top \mathbf{w}$ and let $\mathbf{w}_T$ be the approximate solution after T removals. Then $\mathbf{w}_T$ is an exact solution of $L_{\mathbf{b}}^{(T)}(\mathbf{w}; \mathcal{D}^{(T)})$, hence, the argument above can be applied to $L_{\mathbf{b}}^{(T)}(\mathbf{w}; \mathcal{D}^{(T)})$ to show that the Newton update approximation $\mathbf{w}_{T+1}$ has gradient residual $\mathbf{u}'$ with norm at most ϵ'. Then:

$$\begin{aligned}\mathbf{u}' = \nabla_{\mathbf{w}} L_{\mathbf{b}}^{(T)}(\mathbf{w}; \mathcal{D}^{(T)}) &= \nabla_{\mathbf{w}} L_{\mathbf{b}}(\mathbf{w}; \mathcal{D}^{(T)}) - \mathbf{u}_T \\ \Rightarrow 0 &= \nabla_{\mathbf{w}} L_{\mathbf{b}}(\mathbf{w}) - \mathbf{u}_T - \mathbf{u}'.\end{aligned}$$

Thus the gradient residual for $L_{\mathbf{b}}(\mathbf{w}; \mathcal{D}^{(T)})$ after $T+1$ removals is $\mathbf{u}_{T+1} := \mathbf{u}_T + \mathbf{u}'$ and its norm is at most $(T+1)\epsilon'$ by the triangle inequality.

Batch removal. In certain scenarios, data removal may not need to occur immediately after the data's owner requests removal. This potentially allows for batch removals in which multiple training samples are removed at once for improved efficiency. The Newton update removal mechanism naturally supports this extension. Assume without loss of generality that the batch of training data to be removed is $\mathcal{D}_m = \{(\mathbf{x}_{n-m+1}, y_{n-m+1}), \ldots, (\mathbf{x}_n, y_n)\}$. Define:

$$\begin{aligned}\Delta^{(m)} &= m\lambda \mathbf{w}^* + \sum_{j=n-m+1}^{n} \nabla \ell((\mathbf{w}^*)^\top \mathbf{x}_j, y_j) \\ H_{\mathbf{w}^*}^{(m)} &= \nabla^2 L(\mathbf{w}^*; \mathcal{D} \setminus \mathcal{D}_m).\end{aligned}$$

The batch removal update is:

$$\mathbf{w}^{(-m)} = \mathbf{w}^* + \left[H_{\mathbf{w}^*}^{(m)}\right]^{-1} \Delta^{(m)}. \tag{6}$$

We derive bounds on the gradient residual norm for batch removal that are similar Theorem 1 and Corollary 1.

Theorem 4. *Under the same regularity conditions of Theorem 1, we have that:*

$$\begin{aligned}\|\nabla L(\mathbf{w}^{(-m)}; \mathcal{D} \setminus \mathcal{D}_m)\|_2 &\leq \gamma(n-m) \left\| \left[H_{\mathbf{w}^*}^{(m)}\right]^{-1} \Delta^{(m)} \right\|_2^2 \\ &\leq \frac{4\gamma m^2 C^2}{\lambda^2 (n-m)}.\end{aligned}$$

Algorithm 1 Training of a certified removal-enabled model.

```
1: Input: Dataset $\mathcal{D}$, loss $\ell$, parameters $\sigma, \lambda > 0$.
2: Sample $\mathbf{b} \sim \mathcal{N}(0, \sigma)^d$.
3: Return: $\text{argmin}_{\mathbf{w} \in \mathbb{R}^d} \sum_{i=1}^n \ell(\mathbf{w}^\top \mathbf{x}_i, y_i) + \lambda n \|\mathbf{w}\|_2^2 + \mathbf{b}^\top \mathbf{w}$.
```

Algorithm 2 Repeated certified removal of data batches.

```
1: Input: Dataset $\mathcal{D}$, loss $\ell$, parameters $\epsilon, \delta, \sigma, \lambda > 0$.
2:        Lipschitz constant $\gamma$ of $\ell''$.
3:        Solution $\mathbf{w}$ computed by Algorithm 1.
4:        Sequence of batches of training sample
5:        indices to be removed: $B_1, B_2, \ldots$
6: Gradient residual bound $\beta \leftarrow 0$.
7: $c \leftarrow \sqrt{2 \log(1.5/\delta)}$.
8: $K \leftarrow \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top$.
9: $X \leftarrow [\mathbf{x}_1 | \mathbf{x}_2 | \cdots | \mathbf{x}_n]^\top$.
10: for $j = 1, 2, \ldots$ do
11:   $\Delta \leftarrow |B_j| \lambda \mathbf{w} + \sum_{i \in B_j} \nabla \ell(\mathbf{w}^\top \mathbf{x}_i, y_i)$.
12:   $H \leftarrow \sum_{i: i \notin B_1, B_2, \ldots, B_j} \nabla^2 \ell(\mathbf{w}^\top \mathbf{x}_i, y_i)$.
13:   $X \leftarrow \texttt{remove\_rows}(X, B_j)$.
14:   $K \leftarrow K - \sum_{i \in B_j} \mathbf{x}_i \mathbf{x}_i^\top$.
15:   $\beta \leftarrow \beta + \gamma \sqrt{\|K\|_2} \cdot \|H^{-1}\Delta\|_2 \cdot \|XH^{-1}\Delta\|_2$.
16:   if $\beta > \sigma\epsilon/c$ then
17:     Re-train from scratch using Algorithm 1.
18:   else
19:     $\mathbf{w} \leftarrow \mathbf{w} + H^{-1}\Delta$.
20:   end if
21: end for
```

Corollary 2. *The Newton update* $\mathbf{w}^{(-m)} = \mathbf{w}^* + \left[H_{\mathbf{w}^*}^{(m)}\right]^{-1} \Delta^{(m)}$ *satisfies:*

$$\|L(\mathbf{w}^{(-m)}; \mathcal{D} \setminus \mathcal{D}_m)\|_2 \leq$$

$$\gamma \|X^{(-m)}\|_2 \left\| \left[H_{\mathbf{w}^*}^{(m)}\right]^{-1} \Delta^{(m)} \right\|_2 \left\| X^{(-m)} \left[H_{\mathbf{w}^*}^{(m)}\right]^{-1} \Delta^{(m)} \right\|_2,$$

where $X^{(-m)}$ *is the data matrix for* $\mathcal{D} \setminus \mathcal{D}_m$ *and* γ *is the Lipschitz constant of* ℓ''.

Interestingly, the gradient residual bound in Theorem 4 scales quadratically with the number of removals, as opposed to linearly when removing examples one-by-one. This increase in error is due to a more crude approximation of the Hessian, that is, we compute the Hessian only once at the current solution $\mathbf{w}^*$ rather than once per removed data.

Reducing online computation. The Newton update requires forming and inverting the Hessian. Although the $O(d^3)$ cost of inversion is relatively limited for small d and inversion can be done efficiently on GPUs, the cost of forming the Hessian is $O(d^2 n)$, which may be problematic for large datasets. However, the Hessian can be formed at training time, *i.e.*, before the data to be removed is presented, and only the inverse needs to be computed at removal time.

When computing the data-dependent bound, a similar technique can be used for calculating the term $\|X^- H_{\mathbf{w}^*}^{-1} \Delta\|_2$ – which involves the product of the $(n-1) \times d$ data matrix X^- with a d-dimensional vector. We can reduce the online component of this computation to $O(d^3)$ by forming the SVD of X offline and applying online down-dates (Gu and Eisenstat, 1995) to form the SVD of X^- by solving an eigen-decomposition problem on a $d \times d$ matrix. It can be shown that this technique reduces the computation of $\|X^- H_{\mathbf{w}^*}^{-1} \Delta\|_2$ to involve only $d \times d$ matrices and d-dimensional vectors, which enables the online computation cost to be independent of n.

Pseudo-code. We present pseudo-code for training removal-enabled models and for the (ϵ, δ)-CR Newton update mechanism. During training (Algorithm 1), we add a random linear term to the training loss by sampling a Gaussian noise vector $\mathbf{b}$. The choice of σ determines a "removal budget" according to Theorem 3: the maximum gradient residual norm that can be incurred is $\sigma\epsilon/c$. When optimizing the training loss, any optimizer with convergence guarantee for strongly convex loss functions can be used to find the minimizer in Algorithm 1. We use L-BFGS (Liu and Nocedal, 1989) in our experiments as it was the most efficient of the optimizers we tried.

During removal (line 19 in Algorithm 2), we apply the batch Newton update (Equation 6) and compute the gradient residual norm bound using Corollary 2 (line 15 in Algorithm 2). The variable β accumulates the gradient residual norm over all removals. If the pre-determined budget of $\sigma\epsilon/c$ is exceeded, we train a new removal-enabled model from scratch using Algorithm 1 on the remaining data points.

3.3. Non-Linear Models

Deep learning models often apply a linear model to features extracted by a network pre-trained on a public dataset like ImageNet (Ren et al., 2015; He et al., 2017; Zhao et al., 2017; Carreira and Zisserman, 2017) for vision tasks, or from language model trained on public text corpora (Devlin et al., 2019; Dai et al., 2019; Yang et al., 2019; Liu et al., 2019) for natural language tasks. In such setups, we only need to worry about data removal from the linear model that is applied to the output of the feature extractor.

When feature extractors are trained on private data as well, we can use our certified-removal mechanism on linear models that are applied to the output of a differentially-private feature extraction network (Abadi et al., 2016).

Theorem 5. *Suppose* Φ *is a randomized learning algorithm that is* $(\epsilon_{DP}, \delta_{DP})$*-differentially private, and the out-*

Dataset	**MNIST (§4.1)**	**LSUN (§4.2)**	**SST (§4.2)**	**SVHN (§4.3)**
Removal setting	CR Linear	Public Extractor + CR Linear	Public Extractor + CR Linear	DP Extractor + CR Linear
Removal time	0.04s	0.48s	0.07s	0.27s
Training time	15.6s	124s	61.5s	1.5h

Table 1. **Summary of removal and training times observed in our experiments.** For LSUN and SST, the public extractor is trained on a public dataset and hence removal is only applied to the linear model. For SVHN, removal is applied to a linear model that operates on top of a differentially private feature extractor. In all cases, using the Newton update to (certifiably) remove data is several orders of magnitude faster than re-training the model from scratch.

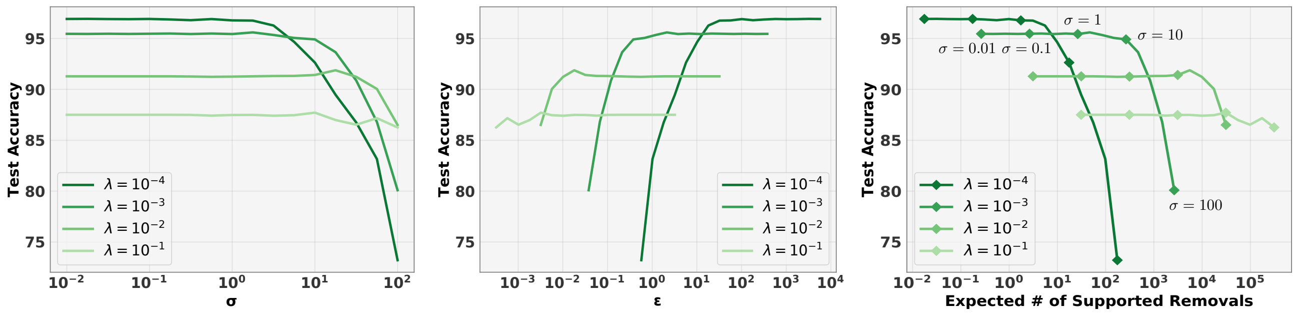

Figure 1. **Linear logistic regression on MNIST. Left**: Effect of L_2-regularization parameter, λ, and standard deviation of the objective perturbation, σ, on test accuracy. **Middle**: Effect of ϵ on test accuracy when supporting 100 removals. **Right**: Trade-off between accuracy and supported number of removals at $\epsilon = 1$. At a given ϵ, higher λ and σ values reduce test accuracy but allow for many more removals.

puts of Φ are used in a linear model by minimizing $L_{\mathbf{b}}$ and using a removal mechanism that guarantees $(\epsilon_{CR}, \delta_{CR})$-certified removal. Then the entire procedure guarantees $(\epsilon_{DP}+\epsilon_{CR}, \delta_{DP}+\delta_{CR})$-certified removal.

The advantage of this approach over training the entire network in a differentially private manner (Abadi et al., 2016) is that the (removal-enabled) linear model can be trained using a much smaller perturbation, which may greatly boost the accuracy of the final model (see Section 4.3).

4. Experiments

We test our certified removal mechanism in three settings: (1) removal from a standard linear logistic regressor, (2) removal from a linear logistic regressor that uses a feature extractor pre-trained on public data, and (3) removal from a non-linear logistic regressor by using a differentially private feature extractor. Code reproducing the results of our experiments is publicly available from `https://github.com/facebookresearch/certified-removal`. Table 1 summarizes the training and removal times measured in our experiments.

4.1. Linear Logistic Regression

We first experiment on the MNIST digit classification dataset. For simplicity, we restrict to the binary classification problem of distinguishing between digits 3 and 8, and train a regularized logistic regressor using Algorithm 1. Removal is performed using Algorithm 2 with $\delta = \texttt{1e-4}$.

Effects of λ and σ. Training a removal-enabled model using Algorithm 1 requires selecting two hyperparameters: the L_2-regularization parameter, λ, and the standard deviation, σ, of the sampled perturbation vector $\mathbf{b}$. Figure 1 shows the effect of λ and σ on test accuracy and the expected number of removals supported before re-training. When fixing the supported number of removals at 100 (middle plot), the value of σ is inversely related to ϵ (*cf.* line 16 of Algorithm 2), hence higher ϵ results in smaller σ and improved accuracy. Increasing λ enables more removals before re-training (left and right plots) because it reduces the gradient residual norm, but very high values of λ negatively affect test accuracy because the regularization term dominates the loss.

Tightness of the gradient residual norm bounds. In Algorithm 2 , we use the data-dependent bound from Corollaries 1 and 2 to compute a *per-data* or *per-batch* estimate of the removal error, as opposed to the *worst-case* bound in Theorems 1 and 4. Figure 2 shows the value of different bounds as a function of the number of removed points. We consider two removal scenarios: single point removal and batch removal with batch size $m = 10$. We observe three phenomena: (1) The worst-case bounds (light blue and light green) are *several orders of magnitude* higher than the data-dependent bounds (dark blue and dark green), which means that the number of supported removals is *several orders of magnitude* higher when using the data-dependent bounds. (2) The cumulative sum of the gradient residual

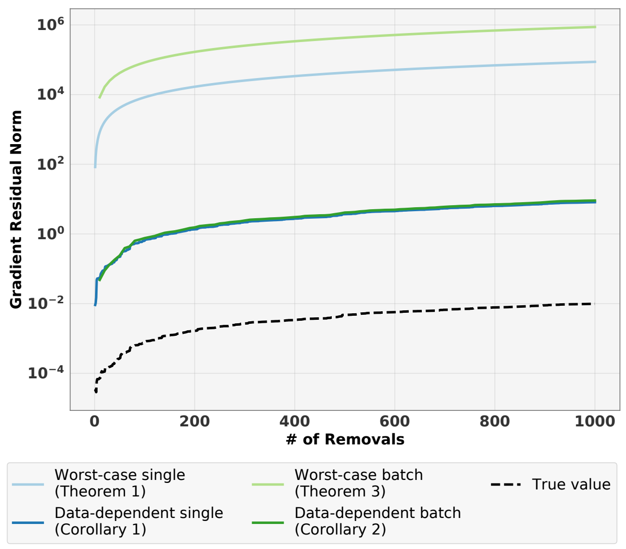

Figure 2. **Linear logistic regression on MNIST.** Gradient residual norm (on log scale) as a function of the number of removals.

norm bounds is approximately linear for both the single and batch removal data-dependent bounds. (3) There remains a large gap between the data-dependent norm bounds and the true value of the gradient residual norm (dashed line), which suggests that the utility of our removal mechanism may be further improved via tighter analysis.

Gradient residual norm and removal difficulty. The data-dependent bound is governed by the norm of the update $H_{\mathbf{w}^*}^{-1}\Delta$, which measures the influence of the removed point on the parameters and varies greatly depending on the training sample being removed. Figure 3 shows the training samples corresponding to the 10 largest and smallest values of $\|H_{\mathbf{w}^*}^{-1}\Delta\|_2$. There are large visual differences between these samples: large values correspond to oddly-shaped 3s and 8s, while small values correspond to "prototypical" digits. This suggests that removing outliers is harder, because the model tends to memorize their details and their impact on the model is easy to distinguish from other samples.

4.2. Non-Linear Logistic Regression using Public, Pre-Trained Feature Extractors

We consider the common scenario in which a feature extractor is trained on public data (*i.e.*, does not require removal), and a linear classifier is trained on these features using non-public data. We study two tasks: (1) scene classification on the LSUN dataset and (2) sentiment classification on the Stanford Sentiment Treebank (SST) dataset. We subsample the LSUN dataset to 100K images per class (*i.e.*, $n = 1$M).

For LSUN, we extract features using a ResNeXt-101 model (Xie et al., 2017) trained on 1B Instagram images (Mahajan et al., 2018) and fine-tuned on ImageNet (Deng et al., 2009). For SST, we extract features using a pre-trained RoBERTa (Liu et al., 2019) language model. At removal time, we use Algorithm 2 with $\epsilon = 1$ and $\delta = \texttt{1e-4}$ in both experiments.

Result on LSUN. We reduce the 10-way LSUN classification task to 10 one-versus-all tasks and randomly subsample the negative examples to ensure the positive and negative classes are balanced in all binary classification problems. Subsampling benefits removal since a training sample does not always need to be removed from all 10 classifiers.

Figure 4 (left) shows the relationship between test accuracy and the expected number of removals on LSUN. The value of (λ, σ) is shown next to each point, with the left-most point corresponding to training a regular model that supports no removal. At the cost of a small drop in accuracy (from 88.6% to 83.3%), the model supports over $10,000$ removals before re-training is needed. As shown in Table 1, the computational cost for removal is more than $250\times$ smaller than re-training the model on the remaining data points.

Result on SST. SST is a sentiment classification dataset commonly used for benchmarking language models (Wang et al., 2019). We use SST in the binary classification task of predicting whether or not a movie review is positive. Figure 4 (right) shows the trade-off between accuracy and supported number of removals. The regular model (left-most point) attains a test accuracy of 89.0%, which matches the performance of competitive prior work (Tai et al., 2015; Wieting et al., 2016; Looks et al., 2017). As before, a large number of removals is supported at a small loss in test accuracy; the computational costs for removal are $870\times$ lower than for re-training the model.

4.3. Non-linear Logistic Regression using Differentially Private Feature Extractors

When public data is not available for training a feature extractor, we can train a differentially private feature extractor on private data (Abadi et al., 2016) and apply Theorem 5 to remove data from the final (removal-enabled) linear layer. This approach has a major advantage over training the entire model using the approach of (Abadi et al., 2016) because the final linear layer can partly correct for the noisy features produced by the private feature extractor.

We evaluate this approach on the Street View House Numbers (SVHN) digit classification dataset. We compare it to a differentially private CNN[2] trained using the technique of Abadi et al. (2016). Since the CNN is differentially private, certified removal is achieved trivially without applying any removal. For a fair comparison, we fix $\delta = \texttt{1e-4}$ and train $(\epsilon_{\text{DP}}/10, \delta)$-private CNNs for a range of values of ϵ_{DP}.

[2]We use a simple CNN with two convolutional layers with 64 filters of size 3×3 and 2×2 max-pooling.

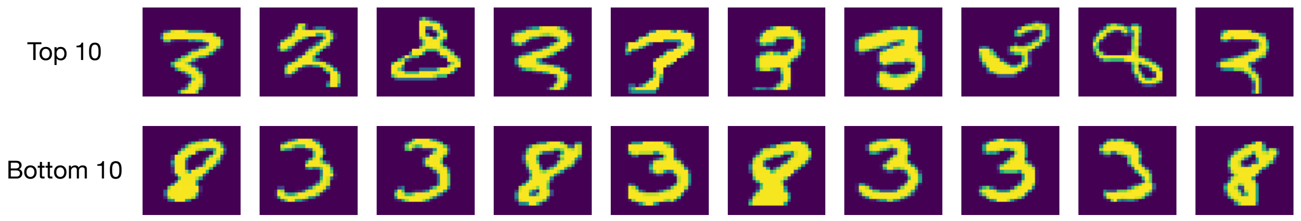

Figure 3. **MNIST training digits sorted by norm of the removal update $\|\mathbf{H}_{\mathbf{w}^*}^{-1}\mathbf{\Delta}\|_2$.** The samples with the highest norm (**top**) appear to be atypical, making it harder to undo their effect on the model. The samples with the lowest norm (**bottom**) are prototypical 3s and 8s, and hence are much easier to remove.

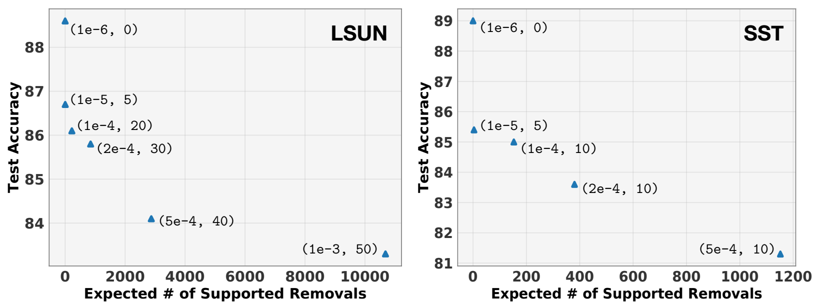

Figure 4. **Linear models trained on public feature extractors.** Trade-off between test accuracy and the expected number of supported removals (at $\epsilon = 1$) on LSUN (**left**) and SST (**right**). The setting of (λ, σ) is shown next to each point. The number of supported removals rapidly increases when accuracy is slightly sacrificed.

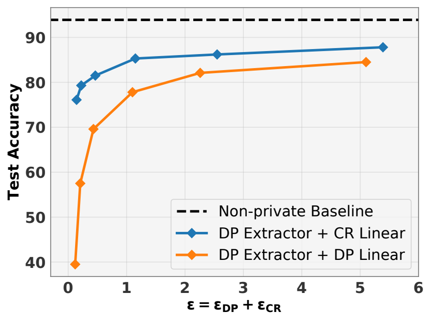

Figure 5. **Using ϵ-DP features.** Trade-off between ϵ and test accuracy on SVHN of models that support 10 removals. Dashed line shows non-private model accuracy.

By the union bound for group privacy (Dwork, 2011), the resulting models support up to then $(\epsilon_{\text{DP}}, \delta)$-CR removals.

To measure the effectiveness of Theorem 5 for certified data removal, we also train an $(\epsilon_{\text{DP}}/10, \delta/10)$-differentially private CNN and extract features from its penultimate linear. We use these features in Algorithm 1 to train 10 one-versus-all classifiers with total failure probability of at most $\frac{9}{10}\delta$. Akin to the experiments on LSUN, we subsample the negative examples in each of the binary classifiers to speed up removal. The expected contribution to ϵ from the updates is set to $\epsilon_{\text{CR}} \approx \epsilon_{\text{DP}}/10$, hence achieving (ϵ, δ)-CR with $\epsilon = \epsilon_{\text{DP}} + \epsilon_{\text{CR}} \approx \epsilon_{\text{DP}} + \epsilon_{\text{DP}}/10$ after 10 removals.

Figure 5 shows the relationship between test accuracy and ϵ for both the fully private and the Newton update removal methods. For reference, the dashed line shows the accuracy obtained by a non-private CNN that does not support removal. For smaller values or ϵ, training a private feature extractor (blue) and training the linear layer using Algorithm 1 attains much higher test accuracy than training a fully differentially private model (orange). In particular, at $\epsilon \approx 0.1$, the fully differentially private baseline's accuracy is only 22.7%, whereas our approach attains a test accuracy of 71.2%. Removal from the linear model trained on top of the private extractor only takes 0.27s, compared to more than 1.5 hour when re-training the CNN from scratch.

5. Related Work

Removal of specific training samples from models has been studied in prior work on decremental learning (Cauwenberghs and Poggio, 2000; Karasuyama and Takeuchi, 2009; Tsai et al., 2014) and machine unlearning (Cao and Yang, 2015). Ginart et al. (2019) studied the problem of removing data from k-means clusterings. These studies aim at *exact removal* of one or more training samples from a trained model: their success measure is closeness to the optimal parameter or objective value. This suffices for purposes such as quickly evaluating the leave-one-out error or correcting mislabeled data, but it does not provide a formal guarantee of statistical indistinguishability. Our work leverages differential privacy to develop a more rigorous definition of data removal. Concurrent work (Bourtoule et al., 2019) presents an approach that allows certified removal with $\epsilon = 0$.

Our definition of certified removal uses the same notion of indistinguishability as that of differential privacy. Many classical machine learning algorithms have been shown to support differentially private versions, including PCA (Chaudhuri et al., 2012), matrix factorization (Liu et al., 2015), linear models (Chaudhuri et al., 2011), and neural networks (Abadi et al., 2016). Our removal mechanism can be viewed on a spectrum of noise addition techniques for preserving data privacy, balancing between computation time for the removal mechanism and model utility. We hope to further explore the connections between differential privacy and certified removal in follow-up work to design certified-removal algorithms with better guarantees and computational efficiency.

6. Conclusion

We have studied a mechanism that quickly "removes" data from a machine-learning model up to a differentially private guarantee: the model after removal is indistinguishable from a model that never saw the removed data to begin with. While we demonstrate that this mechanism is practical in some settings, at least four challenges for future work remain. (1) The Newton update removal mechanism requires inverting the Hessian matrix, which may be problematic. Methods that approximate the Hessian with near-diagonal matrices may address this problem. (2) Removal from models with non-convex losses is unsupported; it may require local analysis of the loss surface to show that data points do not move the model out of a local optimum. (3) There remains a large gap between our data-dependent bound and the true gradient residual norm, necessitating a tighter analysis. (4) Some applications may require the development of alternative, less constraining notions of data removal.

References

Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 308–318.

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. pages 1467–1474.

Bourtoule, L., Chandrasekaran, V., Choquette-Choo, C., Jia, H., Travers, A., Zhang, B., Lie, D., and Papernot, N. (2019). Machine unlearning. In *arXiv 1912.03817*.

Cao, Y. and Yang, J. (2015). Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 463–480.

Carlini, N., Liu, C., Kos, J., Erlingsson, U., and Song, D. (2019). The secret sharer: Measuring unintended neural network memorization & extracting secrets. In *USENIX Security Symposium*, pages 267–284.

Carreira, J. and Zisserman, A. (2017). Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750.

Cauwenberghs, G. and Poggio, T. A. (2000). Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 409–415.

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109.

Chaudhuri, K., Sarwate, A. D., and Sinha, K. (2012). Near-optimal differentially private principal components. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 998–1006.

Cook, R. D. and Weisberg, S. (1982). *Residuals and influence in regression*. New York: Chapman and Hall.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dwork, C. (2011). Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340.

Ginart, A., Guan, M. Y., Valiant, G., and Zou, J. (2019). Making AI forget you: Data deletion in machine learning. *CoRR*, abs/1907.05012.

Gu, M. and Eisenstat, S. C. (1995). Downdating the singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 16(3):793–810.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988.

Karasuyama, M. and Takeuchi, I. (2009). Multiple incremental decremental learning of support vector machines. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 907–915.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1885–1894.

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(1-3):503–528.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Liu, Z., Wang, Y.-X., and Smola, A. (2015). Fast differentially private matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 171–178. ACM.

Looks, M., Herreshoff, M., Hutchins, D., and Norvig, P. (2017). Deep learning with dynamic computation graphs. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Mahajan, D., Girshick, R. B., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part II*, pages 185–201.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566.

Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. (2016). Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73.

Tsai, C., Lin, C., and Lin, C. (2014). Incremental and decremental training for linear classification. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 343–352.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2019). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. (2016). Towards universal paraphrastic sentence embeddings. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Xie, S., Girshick, R. B., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision* *and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5987–5995.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Yeom, S., Giacomelli, I., Fredrikson, M., and Jha, S. (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *CSF*.

Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6230–6239.