Ji Gao, Sanjam Garg, Mohammad Mahmoody, and Prashant Nalini Vasudevan

Deletion inference, reconstruction, and compliance in machine (un)learning

Abstract: Privacy attacks on machine learning models aim to identify the data that is used to train such models. Such attacks, traditionally, are studied on *static* models that are trained once and are accessible by the adversary. Motivated to meet new legal requirements, many machine learning methods are recently extended to support machine *unlearning*, i.e., updating models as if certain examples are removed from their training sets, and meet new legal requirements. However, privacy attacks could potentially become more devastating in this new setting, since an attacker could now access both the original model before deletion and the new model after the deletion. In fact, the very act of deletion might make the deleted record *more* vulnerable to privacy attacks. Inspired by cryptographic definitions and the differential privacy framework, we *formally* study privacy implications of machine unlearning. We formalize (various forms of) *deletion inference* and *deletion reconstruction* attacks, in which the adversary aims to either identify which record is deleted or to reconstruct (perhaps part of) the deleted records. We then present successful deletion inference and reconstruction attacks for a variety of machine learning models and tasks such as classification, regression, and language models. Finally, we show that our attacks would provably be precluded if the schemes satisfy (variants of) deletion compliance (Garg, Goldwasser, and Vasudevan, Eurocrypt'20).

Keywords: data leakage, privacy, machine learning, machine unlearning

DOI 10.56553/popets-2022-0079
Received 2021-11-30; revised 2022-03-15; accepted 2022-03-16.

1 Introduction

Machine learning algorithms, in their most basic settings, focus on deriving predictive models with low error by using a collection of training examples $\mathcal{S} = \{e_1, \ldots, e_n\}$. However, a model $h_{\mathcal{S}}$ trained on set $\mathcal{S}$ might reveal (sensitive information about) the examples in $\mathcal{S}$, potentially violating the privacy of the individuals whose contributed those examples. Such exposure, particularly in certain (e.g., medical/political) contexts could be a major concern. In fact, the ever-increasing use of machine learning (ML) as a service [RGC15] for decision making further heightens such privacy concerns. Recent legal requirements (e.g., the European Union's GDPR [HvdSB19] or California's CCPA [dlT18]) aim to make such privacy considerations mandatory. At the same time, a recent line of work [VBE18, CN20, NBW$^+$17, GGV20] aims at (mathematically) formalizing such privacy considerations and their enforcement.

The work of Shokri et al. [SSSS17] demonstrated that natural and even commercialized ML models do, in fact, leak a lot about their training sets. In particular, their work initiated the *membership inference* framework for studying privacy attacks on ML models. In such attacks, an adversary with input example e and access to an ML model $h_{\mathcal{S}}$ aims to deduce if $e \in \mathcal{S}$ or not. In a bigger picture, membership inference of [SSSS17] and many follow-up attacks [LBG17, SZH$^+$19, LBW$^+$18, YZCL19, CTCP20, LZ20, ZBWT$^+$20, SBB$^+$20, JWEG20] as well as *model inversion* attacks [FLJ$^+$14, FJR15, WFJN16, VBE18] can all be seen as demonstrating ways to *infer* or *reconstruct* information about the data sets used in the ML pipeline based on publicly available auxiliary information about them [DN03, DSSU17, BBHM16, DSS$^+$15, SOJH09, HSR$^+$08]. A more recent line of work studies the related question of "memorization" in machine learning models set [SRS17, VBE18, CLE$^+$19, Fel20].

On the defense side, differential privacy [DN03, DMNS06, Dwo08] provides a framework to provably limit the information that would leak about the used training examples. This is done by guaranteeing that including or not including any individual example will

Ji Gao: University of Virginia, E-mail: jg6yd@virginia.edu.
Sanjam Garg: University of California, Berkeley and NTT Research, E-mail: sanjamg@berkeley.edu. .
Mohammad Mahmoody: University of Virginia, E-mail: mohammad@virginia.edu.
Prashant Nalini Vasudevan: National University of Singapore, E-mail: prashant@comp.nus.edu.sg.

have little statistical impact on the distribution of the produced ML model. Consequently, any form of interaction with the trained model h (e.g., even a full disclosure of it) will not reveal too much information about whether a particular example e was a member of the data set or not. Despite being a very powerful privacy guarantee, differential privacy imposes a challenge on the learning process [SCS13, DTTZ14, BST14, DBB18, She19, TTZ15] that usually leads to major utility loss when one uses the same amount of training data compared with non-private training [BST14, BBKN14]. Hence, it is important to understand the level of privacy that can be achieved by more efficient methods as well.

Privacy in the presence of data deletion. The above mentioned attacks are executed in a *static* setting, in which the model is trained once and then the adversary tries to extract information about the training set by interacting with the trained model afterwards. However, this setting is not realistic when models are dynamic and get updated. In particular, in light of the recent attention to the "right to erasure" or the "right to be forgotten," also stressed by legal requirements such as GDPR and CCPA, a new line of work has emerged with the goal of *unlearning* or simply *deleting* examples from machine learning models [CY15, GGVZ19, GAS19, GGV20, BCCC+20, ISCZ20, GGHvdM19, NRSM20]. In this setting, upon a deletion request for an example $e \in \mathcal{S}$, the trainer needs to update the model $h_{\mathcal{S}}$ to h_{-e} such that h_{-e} (ideally) has the same distribution as training a model from scratch using $\mathcal{S} \setminus \{e\}$. Clearly, if an ML model gets updated due to a deletion request, we are no longer dealing with a static ML model.

It might initially seem like a perfect deletion of a example e from a model $h_{\mathcal{S}}$ and releasing h_{-e} instead should *help* with preventing leakage about the particular deleted example e. After all, we are *removing* e from the learning process of the model accessible to the adversary. However, the adversary now could potentially access *both* models $h_{\mathcal{S}}$ and h_{-e}, and so it might be able to extract even *more* information about the deleted example e compared to the setting in which the adversary could only access $h_{\mathcal{S}}$ or h_{-e} alone. As a simplified contrived example, suppose the examples $\mathcal{S} = \{e_1, \dots, e_n\}$ are real-valued vectors, and suppose the ML model $h_{\mathcal{S}}$ (perhaps upon many queries) somehow reveals the summation $\sum_{i \in [n]} e_i$. In this case, if the set $\mathcal{S}$ is sampled from a distribution with sufficient entropy, the trained model $h_{\mathcal{S}}$ might potentially provide a certain degree of privacy for examples in $\mathcal{S}$. However, if one of the examples e_i is deleted from $h_{\mathcal{S}}$, then because the updated model h_{-e_i} also returns the updated summation $\sum_{j \neq i} e_j$, then an adversary who extracts both of these summations can reconstruct the deleted record e_i completely. In other words, the very task of deletion might in fact *harm* the privacy of the very deleted example e_i. Hence, in this work we ask: How vulnerable are ML algorithms to leak information about the deleted examples, if an adversary gets to interact with the models both before and after the deletion updates?

1.1 Our contribution

In this work, we formally study the privacy implications of machine unlearning. Our approach is inspired by cryptographic definitions, differential privacy, and deletion compliance framework of [GGV20]. More specifically, our contribution is two-fold. First, we initiate a formal study of various attack models in the two categories of *reconstruction* and *inference* attacks. Second, we present practical, simple, yet effective attacks on a *broad* class of machine learning algorithms for classification, regression, and text generation that extract information about the deleted example.

Below, we briefly go over new definitions, the relation between them, and the ideas behind our attacks. In what follows, $h_{\mathcal{S}}$ is the model trained on the set $\mathcal{S}$, and h_{-e} is the model after deletion of the example $e \in \mathcal{S}$. When the context is clear, we might simply use h to denote $h_{\mathcal{S}}$ and h_{del} to denote the model after deletion[1]. We assume that the deletion is *ideal*, in the sense that h_{-e} is obtained by a fresh retrain on $\mathcal{S} \setminus \{e\}$.[2] The adversary will have access to $h_{\mathcal{S}}$ followed by access to h_{-e}.

Deletion inference. Perhaps the most natural question about data leakage in the context of machine unlearning is whether deletion can be inferred. In *membership* inference attack, the job of the adversary is to infer whether an example e is a member of the used training set $\mathcal{S}$ or not by interacting with the produced model $h_{\mathcal{S}}$. In this work, we introduce *deletion inference* attacks which are, roughly speaking, analogous to *membership inference* but in the context where some deletion is happening. More specifically, our definition does not

[1] Using h_{del} is particularly useful when we want to refer to the model after deletion, without explicitly revealing the deleted example e.

[2] We suspect our attacks should have a good success rate on "approximate" deletion procedures (in which h_{-e} is just close to the ideal version) as well. We leave such studies for future work.

capture whether the deletion is happening or not, and our goal (in the main default definition) is only to hide *which* examples are being deleted. In particular, we formalize the goal of a deletion inference adversary to distinguish between a data example $e \in \mathcal{S}$ that was deleted from an ML model $h_{\mathcal{S}}$ and another example $e' \in \mathcal{S}$ (or $e' \notin \mathcal{S}$) that is not deleted from $\mathcal{S}$. We follow the cryptographic game-based style of security definitions. (See Definition 3.1 for the formal definition.)

Given examples e_0, e_1, with the promise that one of them is deleted and the other is not, one can always reduce the goal of a deletion inference adversary to *membership inference* by first inferring membership of e_0, e_1 in the two models h, h_{del}. However, given that the adversary has access to both of h, h_{del}, it is reasonable to suspect that much more can be done by a deletion inference adversary than what can be done through a reduction to membership inference. In fact, this is exactly what we show in Section 3.3. We show that when both models h, h_{del} can be accessed, relatively simple attacks can be designed to distinguish the deleted examples from the other examples by relying on the intuition that a useful model is usually more fit to the training data than to other data. In Section 3, we show the power of such attacks on a variety of models and real world data sets for both regression and classification. In each case, we both study deletion inference adversaries who know the full labeled examples e_0, e_1 (and infer which one of them are deleted) as well as stronger attackers who only know the (unlabeled) *instances* x_0, x_1.

Deletion reconstruction. The second category of our attacks focus on *reconstructing* part or all of the deleted example e. As anticipated, reconstruction attacks are *stronger* (and hence harder to achieve) attacks that can be used for obtaining deletion inference attacks as well (see Theorem 4.2). In all of our reconstruction attacks, the adversary is not given any explicit examples, and its goal is to extract information about the features of the deleted instance. We now describe some special cases of reconstruction attacks that we particularly study.

- **Deleted instance reconstruction.** Can an adversary fully or approximate find the features of a deleted instance x (where $e = (x, y)$ is the deleted example)? We show that for natural data distributions (both theoretical and real data) the 1-nearest neighbor classifier can completely reveal the deleted instance, even if the adversary has only black-box access to the models before and after deletion. In particular, we show that when the instances are uniformly distributed over $\{0,1\}^d$, and the model is the 1-nearest neighbor model, an adversary can extract virtually *all* of the features of the deleted instance (see Section 4.1). We also present attacks on real data for two major application settings: image classification and text generation.
  - **Deleted image reconstruction.** We show similar attacks on 1-nearest neighbor over the Omniglot dataset, where the job of the adversary is to extract *visually similar* pictures to the deleted ones (see Section 4.1.2).
  - **Deleted sentence reconstruction.** We then study deletion reconstruction attacks on language models. Here, a language model gets updated to remove an input (e.g., a sentence) e, and the job of the adversary is to find useful information about e. We show that for simple language models such as bigram or trigram models, the adversary can extract e completely.
- **Deleted label reconstruction.** Suppose we deal with a classification problem. For a deleted example $(x, y) = e$, can an adversary *who does not know the instance* x infer any information about the label of the deleted point? We show that this is indeed possible with a simple idea when the data set is not too large. In particular, the deletion of a point with label c reduces the probability that the new model outputs label c in general, and using this idea we give simple yet successful attacks. Now, suppose the adversary is somehow *aware* of the instance x of a deleted example $(x, y) = e$. Can the adversary leverage knowing the instance x to learn *more information* about the label y, than each of the models h, h_{del} alone provide? We show that doing so is possible for linear regression. In particular, we show an attack using which one can extrapolate a deleted point's label to a *higher* precision than what is provided through the original model h or the model after deletion h_{del}. (See Section C.1 for more details.)

Weak deletion compliance. The above results all deal with first defining *attack models* and then presenting attacks within those frameworks. Next, we ask if it is possible to realize machine learning algorithms with deletion mechanisms that offer meaningful notions of privacy for the deleted points. We approach this question through the lens of the recent work of Garg et al. [GGV20] in which they provide a general "deletion complience" framework that provides strong definitions of private data deletion. We first give a formal comparison between the framework of [GGV20] with our attack models and show that the deletion compliance framework of [GGV20] indeed captures all of the abovementioned attack models. Furthermore, we also present a *weakened* variant of the definition of [GGV20] that is adapted to a setting where the fact that deletion happened itself is allowed to leak. We believe this is a natural setting that needs special attention. For example, consider a text with redacted parts; this reveals the fact that deletion has happened, but not necessarily the redacted text. We further weaken the framework of [GGV20] by only revealing to the adversary what can be accessed through *black-box access* to the model and *not* the full state of the model. We show that even such weaker variants of deletion compliance still capture all of our attacks, and hence is sufficient for positive results. This means that, as shown by [GGV20], differential privacy (with strong parameters) can be used to prevent all attacks of our paper. However, note that enforcing differential privacy comes with costs in efficiency and sample complexity. Hence, it remains an interesting direction to find more efficient schemes (both in terms of running time and sample complexity) that satisfy our weaker notions of deletion compliance introduced in this work. See Section 5 for more discussions.

Motivation behind the attacks. At a high level, our work is relevant in any context in which (1) the users who provide the data examples care about their privacy and prefer not to reveal their participation in the data set $\mathcal{S}$ (2) the system aims to provide the deletion operation, perhaps due to legal requirements. Condition (1) essentially holds in any scenario in which membership inference constitutes a legitimate threat. In scenarios where conditions (1) and (2) hold, if the adversary maintains continuous access to the machine learning model (e.g., when the model is provided as public service) then all the attacks studied in this paper are relevant to practice and would model different adversarial power.

Our security games model attacks in which the adversary aims to infer (or reconstruct) deletion of a *random* example from a dataset. Real world adversaries are stronger in the sense that they could have a specific target in mind before making their queries to the online model. Moreover, real world adversaries usually have a lot of auxiliary information (e.g., as those exploited in the attacks on privacy on users in the Netflix challenge [NS06]) while our attackers have a minimal knowledge about the distribution from which the data is sampled.

Having a diverse set of security games and attacks is analogous to having many different security games and notions in cryptography (such as CPA and CCA security for encryption) to model different attack scenarios. Informally speaking, and at a very high level, one can also think of the very strong deletion compliance of [GGV20] as “UC security” [Can01], while our other security games/notions model weaker security criteria.

1.2 Related work

Chen et al [CZW$^+$21] study a setting similar to ours. They show attacks that, given access to two models – one trained on a dataset $\mathcal{S}$ and another on $\mathcal{S} \setminus \{e\}$ – determine whether a given input e' is equal to the deleted item e. This is close to our notion of deletion inference, though not quite the same. They show that their attacks perform much better than plain membership inference on the first model. Our work differs from that of [CZW$^+$21] in the following respects:

1. In addition to deletion inference, we also show various kinds of reconstruction attacks in a variety of models with different reconstruction goals.
2. Their attacks are constructed by running sophisticated learning algorithms on the posteriors corresponding to deleted and not deleted samples. While this results in attacks that work quite well, these attacks have little explanatory power – it is not clear what enables them, and it is hard to tell what the best way to prevent them is. Our attacks, on the other hand, make use of simple statistics of the outputs of the models.
3. They show that certain measures like publishing only the predicted label or using differential privacy can stop their attacks from working, but this is far from showing that such measures prevent all possible attacks. In order to prove security against all attacks, a formalization of what entails such security is necessary. We provide formal definitions of privacy and formally build a connection to the deletion compliance framework of [GGV20], which, as corollary, implies that differential privacy can provably prevent any possible deletion inference attack.

The work of Salem et al [SBB$^+$20] also studies a related setting. In their case, a model is updated by the addition of new samples, rather than by deletion, and they show attacks that partially reconstruct either the new sample itself or its label. These attacks are constructed by training generative models on posteriors of various samples from a shadow model. It is possible that their attacks can be used when data is deleted as well. In fact, our attacks can also potentially be adapted to be applied when the data is *added* rather than deleted (but the security game needs to change to formally allow this). They also present a cursory discussion of possible defences against their attacks, suggesting that adding noise to the posteriors or differential privacy might work. The distinction of our work from theirs is along the same lines as above – our attacks are simpler and more transparent, and our formalization allows us to identify strategies for *provable* security against arbitrary attacks by proving the relation of our attacks and the deletion compliance of [GGV20]. On the attack side, our work studies the attack landscape with much more granularity by studying very specific attacks that aim to only reconstruct (or infer) the instances, or their labels, or leverage the knowledge of the instance to better approximate the labels.

2 Preliminaries

Basic notation. $[n]$ denotes $\{1,\ldots,n\}$. $\mathcal{X}$ denotes the *instance* space, and $\mathcal{Y}$ denotes the *label* space. For regression tasks, $\mathcal{Y} = \mathbb{R}$ is the set of real numbers, and for classification tasks $\mathcal{Y}$ is a finite set where by default $\mathcal{Y} = \{0,1\}$. D denotes a distribution over $\mathcal{X}\times\mathcal{Y}$, and D^n denotes the n-fold product of D. A sample $e = (x,y) \leftarrow D$ is called a (labeled) *example*. By $D \equiv D'$ we denote that D, D' are identically distributed. When the data examples are not necessarily iid sampled, we use S_n to denote a *distribution* over data sets of size n (one special case is $S_n \equiv D^n$) and we use $\mathcal{S} \leftarrow S_n$ to denote sampling $\mathcal{S}$ from S_n. $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ denotes a set of *models* (aka *hypothesis class*) mapping $\mathcal{X}$ to $\mathcal{Y}$. For example, $\mathcal{H}$ could be the set of all neural nets with a specific architecture and size or the set of half spaces in dimension d when $\mathcal{X} = \mathbb{R}^d$.

Loss, risk, and learning. A loss function $\ell\colon \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ maps an input $(h,(x,y))$ to $\mathbb{R}$ and measures how bad the prediction of h on x is compared to the true label y. For classification, we use the 0-1 loss $\ell(h,e) = 1[h(x) \neq y]$, where 1 is the Boolean indicator random variable. Learn denotes a (perhaps randomized) learner that maps any (unordered) set of examples $\mathcal{S} = \{e_1,\ldots,e_n\}$ to a model $h \in \mathcal{H}$. $\mathsf{Risk}_D(h) = \mathbb{E}_{e\leftarrow D}\,\ell(h,e)$ denotes the *population risk* of h over a distribution D. $\mathsf{Risk}_{\mathcal{S}}(h) = \mathbb{E}_{e\leftarrow\mathcal{S}}\,\ell(h,e)$ denotes the *empirical risk* of h over a training set $\mathcal{S} \subset (\mathcal{X}\times\mathcal{Y})^*$. The Empirical Risk Minimization rule ERM is the learner that simply outputs a model that minimizes the empirical loss $\mathsf{ERM}(\mathcal{S}) = \operatorname{argmin}_{h\in\mathcal{H}} \mathsf{Risk}_{\mathcal{S}}(h)$.

Deletion. Fix a learner Learn, training set $\mathcal{S}$, and model $h \leftarrow \mathsf{Learn}(\mathcal{S})$. We use $h_{-e} \leftarrow \mathsf{Del}_{\mathcal{S}}(h,e)$ to denote the "ideal" data deletion procedure [GGVZ19] that outputs $h_{-e} \leftarrow \mathsf{Learn}(\mathcal{S} \setminus \{e\})$ using fresh randomness for Learn if needed. (Hence, if $e \notin \mathcal{S}$, then $\mathsf{Del}_{\mathcal{S}}(h,e)$ simply returns a fresh retraining on $\mathcal{S}$.) In general, Del needs to know the training set on which h is trained, or it needs a data structure that keeps some information about $\mathcal{S}$ in addition to h. Whenever $\mathcal{S}$ is clear from the context, we might simply write $h_{-e} \leftarrow \mathsf{Del}(h,e)$.

3 Deletion inference attacks

In this section, we describe a framework of attacks on machine unlearning (i.e., machine learning with deletion option) schemes that can *infer* the deleted examples. Such attacks are executed by adversaries who first access the model before deletion followed by having access to the model after deletion. In each case, we will first formally explain our threat model. We also provide theoretical intuition behind our attacks and report experimental findings by implementing those attacks.

Threat model. We define a security game that captures how well an adversary can tell which element is being deleted from the training set. Note that our (default) definition is not aiming to hide the fact that *something* is being deleted, and the only thing we try to hide is *which* element is being deleted. We use a definition that is inspired by how (CPA or CCA) security of encryption schemes are defined through indistinguishability-based security games [GM84, NY90].

Definition 3.1 (Deletion inference)**.** Let Learn be a learner, Del be a deletion mechanism for Learn, and S_n be a distribution on datasets of size n. The adversary Adv and the challenger Chal interact as follows.

1. **Sampling the data and revealing the challenges.** Chal picks a dataset $\{z_1,\ldots,z_n\} = \mathcal{S} \leftarrow S_n$ of size n. Chal picks two indices $i \neq j \in [n]$ at random and sends $e_0 = z_i, e_1 = z_j$ to Adv.
2. **Oracle access before deletion.** Chal trains $h \leftarrow \mathsf{Learn}(\mathcal{S})$. Adv is then given *oracle* access to h, and finally instructs moving to the next step.
3. **Random selection and deletion.** Chal picks $b \leftarrow \{0,1\}$ at random and lets $h_{\mathsf{del}} \leftarrow \mathsf{Del}(h,e_b)$.
4. **Oracle access after deletion.** The adversary Adv is now given oracle access (only) to h_{del}.
5. **Adversary's guess.** The adversary sends out a bit b' to Chal and wins if $b' = b$.

The scheme $(\mathsf{Learn},\mathsf{Del})$ is called ρ *insecure against deletion inference* for data distribution S_n, if there is a PPT adversary Adv whose success probability in the game above is at least ρ. (Note that achieving $\rho = 1/2$ is trivial.) Now, consider a modified game in which the adversary is given *only the instances* (x_0, x_1) where $e_0 = (x_0, y_0), e_1 = (x_1, y_1)$. We call this game the *instance* deletion inference. If an adversary has success probability at least ρ in the instance deletion inference game, then the scheme (Learn, Del) is called ρ insecure against *instance* deletion inference for distribution S_n. Similarly, we define *label* deletion inference, in which only the labels (y_0, y_1) are revealed to the adversary, and ρ-insecurity against such attacks accordingly. To contrast with instance and label deletion inference, we might use *example* deletion inference attack to refer to our default deletion inference attacks. ◇

Note that winning in an instance or label deletion inference game is potentially harder than winning the normal variant (with full examples revealed to the adversary) as the adversary can always ignore the full information given to it. Hence, showing successful instance deletion inference attacks is a stronger (negative) result. We empirically study the power of attacks in all these attack models.

Other variants of Definition 3.1. Definition 3.1 can be seen as a *weak* definition of privacy for deletion inference. The following list describe variants of Definition 3.1 that are either directly weaker, or our attacks can be adapted to in a rather straightforward way.

- **Two-challenges vs. one challenge.** Definition 3.1 includes two challenge examples and asks an adversary to find out which one is the actual deleted one. An alternative definition would only reveal one example to the adversary and asks it to tell if the example is deleted or not.[3]
- **Deletion-revealing vs. deletion-hiding.** Definition 3.1 does not aim to hide the fact that a deletion has happened. An alternative definition could even aim to capture hiding the deletion itself by sampling the non-deleted example *outside* the dataset.
- **Random vs. chosen challenges.** Definition 3.1 asks the adversary to distinguish between a *random* pair of challenge examples, one of which is deleted. In a stronger attack model, the adversary is allowed to *choose* the challenge examples.
- **Auxiliary information.** Definition 3.1 does not explicitly give any extra information about other examples $e_k, k \notin \{i, j\}$ to the adversary, while a real-word adversary might have such knowledge.
- **Multiple deletions vs. one deletion.** Definition 3.1 does not allow more than one deletion to happen, while in general users might request multiple deletions to happen over time. In fact, in Section 3.3, we use this variant of the attacks to test our attacks on large data sets and compare the result with deletion inference attacks that are obtained by reduction to membership inference.

In Section 5, we discuss stronger security definitions that once satisfied would prevent the attack of Definition 3.1 and all the variants above as special cases of the Deletion Compliance framework of Gar et al. [GGV20]. In particular, the definitions of this section (including Definition 3.1) model *weaker* security guarantees than that of Deletion Compliance framework of [GGV20], which makes our attack results of this section stronger.

Learner's randomness. The random seeds of the learners manifest themselves in two ways:

1. Random initialization: Some parameters are generated from a random distribution.
2. Stochastic optimization: We train deep learning models using stochastic gradient descent, which runs with a randomly permutation of the training dataset in each epoch.

Following the definition of perfect deletion, as desired in Definitions 3.1 and 4.1, in our experiments we retrain the models from scratch *without fixing* the random seeds. However, We fix all the deterministic and algorithmic aspects, which include model architecture, the hyperparameters, and the initialization *distribution.*

Our deletion inference attacks. We propose two variants of attacks: (1) (example) deletion inference attack of Del-Inf-Exm which uses both instances and their true labels, and (2) instance inference attack of Del-Inf-Ins which only uses the instances, without knowing the true labels. (In the next subsection, we also show how to *find* the deleted label, which can be seen as a form of "label reconstruction"and is stronger than label inference attacks.)

Attack Del-Inf-Exm using labeled examples. Our example inference attack Del-Inf-Exm is parameterized by a loss function ℓ and proceeds by first computing the loss for both examples e_0, e_1 on both models h, h_{del}.

[3] If one can sample from the set $\mathcal{S}$ the two attack models can be shown to be equivalent using standard hybrid arguments when the adversary's success probability is negligible in security parameter. This is similar to how it is done for CPA/CCA security games in cryptography.

Then, this attack identifies the deleted example by picking the example that leads to a *larger increase* in its loss when we go from h to h_{del}. The intuition behind our attack is that the examples in the dataset are optimized (to a degree depending on the learning algorithm) to have small loss, while examples outside the dataset are not so. Therefore, once an example goes from inside the dataset to outside, it incurs a larger increase in loss. We now define the attack formally.

Algorithm 3.2 (Attack Del-Inf-Exm)**.** The attack is defined with respect to a loss function ℓ. For any example e, we define the *loss increase* of e as: $\delta(e, h, h_{\mathsf{del}}) = \ell(h_{\mathsf{del}}, e) - \ell(h, e)$. The adversary is given two labeled examples $e_0 = (x_0, y_0)$ and $e_1 = (x_1, y_1)$ and also has oracle access to h followed by access to h_{del}. The attack proceeds as follows.

1. Query h on both x_0, x_1.
2. After getting access to h_{del}, query h_{del} on both x_0, x_1.
3. Compute loss increases $\delta(e_0, h, h_{\mathsf{del}})$ and $\delta(e_1, h, h_{\mathsf{del}})$, and let $\alpha = \delta(e_0, h, h_{\mathsf{del}}) - \delta(e_1, h, h_{\mathsf{del}})$.
4. Output 0 if $\alpha > 0$, output 1 if $\alpha < 0$, and output a uniformly random bit $b' \in \{0, 1\}$ if $\alpha = 0$. ◇

Connection to label memorization [Fel20] . At a high level, Del-Inf-Exm can be seen as generalizing the notion of memorization by Feldman [Fel20] from the 0-1 loss to general loss functions. More formally, if we use the 0-1 loss, then for $e \in \mathcal{S}$, the *expected* value $\mathbb{E}_{h_{\mathsf{del}} \leftarrow \mathsf{Del}(h,e)}\, \delta(e, h, h_{\mathsf{del}})$ would become equal to $\mathsf{mem}(\mathsf{Learn}, \mathcal{S}, e)$ defined in [Fel20] to measure how much the learner Learn is memorizing the labels of its training set. , and Using this intuition, our adversary picks the example that is *most memorized* by the model.

The following lemma further formalizes the intuition behind our attack Del-Inf-Exm, so long as the the learning algorithm is the ERM rule.

Lemma 3.3. *Let* ERM *be the empirical risk minimization learning rule using a loss function ℓ. Let h =* $\mathsf{ERM}(\mathcal{S})$*, $h_{-e} \leftarrow$* $\mathsf{Del}(h, e)$ *for $e \in \mathcal{S}$, and $\mathcal{S}_{-e} = \mathcal{S} \setminus \{e\}$. Let $\delta_e = \delta(e, h, h_{-e})$, and let $\delta_{-e} = \mathbb{E}_{e' \leftarrow \mathcal{S}_{-e}}[\delta(e', h, h_{-e})]$ be the expected value of loss increase for examples that* remain *in the dataset. Then the following two hold.*

1. *$\delta_{-e} \leq 0$.*
2. *$\delta_e \geq -(n-1) \cdot \delta_{-e}$ where $n = |\mathcal{S}|$. (In particular, by Part 1, it also holds that $\delta_e \geq 0$.)*

Proof. The first item of the lemma holds simply because we are using the ERM rule. Namely, h_{-e} minimizes the empirical loss over $\mathcal{S}_{-e} = \mathcal{S} \setminus \{e\}$. Therefore:

$$\delta_{-e} = \mathsf{Risk}_{\mathcal{S}_{-e}}(h_{-e}) - \mathsf{Risk}_{\mathcal{S}_{-e}}(h) \leq 0.$$

Having proved the first part, the second part also follows due to using the ERM rule. In particular, suppose for sake of contradiction that $\delta_e < -(n-1) \cdot \delta_{-e}$, where $n = |\mathcal{S}|$. Then,

$$\ell(h, e) + (n-1) \cdot \mathsf{Risk}_{\mathcal{S}_{-e}}(h) > \ell(h_{-e}, e) + (n-1) \cdot \mathsf{Risk}_{\mathcal{S}_{-e}}(h_{-e}).$$

Then, this implies

$$\begin{aligned}\mathsf{Risk}_{\mathcal{S}}(h) &= \frac{\ell(h, e) + (n-1) \cdot \mathsf{Risk}_{\mathcal{S}_{-e}}(h)}{n} \\ &> \frac{\ell(h_{-e}, e) + (n-1) \cdot \mathsf{Risk}_{\mathcal{S}_{-e}}(h_{-e})}{n} = \mathsf{Risk}_{\mathcal{S}}(h_{-e}).\end{aligned}$$

However, the this contradicts that the ERM rule outputs h on training set $\mathcal{S}$. □

Proposition 3.3 shows that whenever (1) $\delta_{-e} = \mathbb{E}_{e' \leftarrow \mathcal{S}_{-e}}[\delta_{e'}] < 0$ and (2) $\delta(e', h, h_{-e})$ for $e' \in \mathcal{S}_{-e}$ is concentrated around its mean δ_{-e}, then for a random $e' \in \mathcal{S}_{-e}$, the attack Del-Inf-Exm of Algorithm 3.2 would likely identify the deleted example correctly. Even though, in general we are not able to prove when these two conditions hold, our experiments confirm that these conditions indeed hold in many natural scenarios, leading to the success of Del-Inf-Exm of Algorithm 3.2.

Attack Del-Inf-Ins using instances only. We now discuss our attack that does not rely on knowing the true labels y_0, y_1. The intuition is that, even if we do not know the true labels, when an example e is deleted from the dataset, the change in the predicted label for e is likely to be more than that of other examples that stay in the dataset. The reason is that for the remaining examples, the model is still trying to keep their prediction close to their correct value, but this optimization is not done for the deleted example e. Hence, our adversary would pick the candidate example that leads to *larger change* in the output *label* (not necessarily the loss). Hence, the attack is more natural to be used for regression tasks, even though it can also be used for classification if one uses the confidence parameters instead of the final labels.

Algorithm 3.4 (Attack Del-Inf-Ins)**.** The attack is parameterized by a distance metric dis over $\mathcal{Y}$ (e.g., $\mathcal{Y} = \mathbb{R}$ and $\mathsf{dis}(y_0, y_1) = |y_0 - y_1|$). The adversary is given two instances x_0, x_1, and it has oracle access to h followed by h_{del}. The attack then proceeds as follows.

1. Query the models (in the order of accessing them) to get $h(x_0)$, $h(x_1)$, $h_{\mathsf{del}}(x_0)$, $h_{\mathsf{del}}(x_1)$, and let $\beta = |h(x_0) - h_{\mathsf{del}}(x_0)| - |h(x_1) - h_{\mathsf{del}}(x_1)|$.

2. Return 0 if $\beta > 0$, return 1 if $\beta < 0$, and return a random answer in $\{0,1\}$ if $\beta = 0$. ◇

3.1 Experiments: Deletion inference attack on regression

Now we apply our attack Del-Inf-Exm (Algorithm 3.2) and attack Del-Inf-Ins (Algorithm 3.4) on multiple regression models including Linear Regression, Lasso regression, SVM Regressor, Decision Tree Regressor, and Neural Network Regressor[4]. D Appendix A includes the details of the attacked models.

Experiment details. Table 1 includes the details of all the datasets we used in the deletion inference experiments and also in other experiments later. We use two regression datasets Boston and Diabetes. For training the original model h, we use a random subset with 90% of the dataset. The experiment follows the security game of Definition 3.1. To ensure the perfect deletion, h_{del} is obtained by a full re-training with the dataset without the deleted example. For the attack Del-Inf-Exm, we use squared loss, which is defined as $\ell(h,(x,y)) = (h(x) - y)^2$. Finally, we repeat the security game of Definition 3.1 1000 times and take the average success probability of the adversaries.

Results. The result is shown in Table 2. In most cases, our adversary gets more than 90% success probability in the deletion inference.

3.2 Experiments: Deletion inference attacks on classification

In this experiment, we apply Del-Inf-Exm and Del-Inf-Ins on classification tasks. In our experiments, we use different models, including logistic regression, support vector machine (SVM), Decision tree, random forest, and multi-layer perceptron (MLP). Due to page limit, details of the models are included in Appendix A.

Experiment details. We use datasets Iris, Wine, Breast Cancer, and 1/12MNIST. (The details of the datasets are shown in Table 1.) Similarly to attacks on regression, We pick a random 90% fraction of the dataset to train the model, and we do a full retrain to obtain h_{del}. The difference compared to the case of regression is that the label space $\mathcal{Y}$ is now a finite set. In this experiment, we assume the output of any hypothesis function $h \in \mathcal{H}$ is a multinomial (confidence) distribution over $\mathcal{Y}$, and this probability is available to the adversary. This assumption is realistic as many machine learning applications have the confidence as part of the output [RGC15], and this is also the default setting of many adversarial machine learning researches [SSSS17, LBW$^+$18][5]. To formally fit the attack into the framework of Definition 3.1, we can extend the set $\mathcal{Y}$ to directly include any such multinomial distribution as the actual output "label".

For Del-Inf-Exm, we use the negative log likelihood loss function $\ell(h,(x,y)) = -\log(\Pr[h(x) = y])$. We then repeat the security game of Definition 3.1 1000 times to approximate the winning probability.

Results. We present the result of attacks Del-Inf-Exm and Del-Inf-Ins on three classification datasets in Table 3. As anticipated, the success rates Del-Inf-Exm are noticeably larger than those of Del-Inf-Ins.

3.3 Attacking large models and datasets

In this section, we aim to show that our deletion inference attacks can be scaled to work with large datasets and models. We first formally describe how *deletion* inference attacks can be obtained through black-box reductions to *membership* inference attacks. We then demonstrate the power of our attacks on datasets of the same size as those of [SSSS17] and compare the power of our direct deletion inference to doing reduction to the membership inference attack of [SSSS17]. We show that using our method can lead to *significantly* stronger results than making a *black-box* use of membership inference attacks.

3.3.1 How to reduce deletion inference to membership inference

One can always reduce the task of deletion inference to the task of membership inference. In particular, if we had a perfect membership inference oracle, we could use it to infer whether a given example is deleted or not

[4] Implementation of the methods are from the python library Scikit-learn.

[5] The model in this scenario is still considered as black-box in most machine learning adversarial literature, but someone may argue it is not fully black-box.

		No. Samples	No. Features	Label	Predict
Regression	Boston [HAR78]	506	14	Real	The median house price
	Diabetes [EHJ+04]	442	10	Real	Disease progression
Classification	Iris [Fis36]	150	4	3 types	The type of iris plants
	Wine [ACD94]	178	13	3 types	Wine cultivator
	Breast Cancer [SWM93]	569	30	Binary	Benign/malignant tumors
	1/12MNIST[LBBH98]	5000	784	10 types	Digit between 0 to 9
	CIFAR-10 [KH+09]	60000	3072	10 classes	Image classification
	CIFAR-100 [KH+09]	60000	3072	100 classes	Image classification

Table 1. Descriptions of the datasets used in deletion inference.

	Boston		Diabetes	
Learning Method	Del-Inf-Exm	Del-Inf-Ins	Del-Inf-Exm	Del-Inf-Ins
Linear regression	99.8%	99.1%	99.8%	99.3%
SVM	93.9%	89.1%	99.2%	100.0%
Lasso regression	98.8%	97.1%	99.3%	98.3%
Decision tree	100.0%	100.0%	100.0%	100.0%
MLP	80.4%	78.3%	72.2%	72.3%

Table 2. Success probabilities of various attacks on regressors for different datasets.

by calling the membership inference oracle on the two models h, h_{del}.

Algorithm 3.5 below shows an intuitive way to reduce deletion inference (DI) to *imperfect* membership inference (MI) in a black-box way. Specifically, suppose the membership inference adversary $M(e,h) \to \{0,1\}$ returns 1 if (it thinks) e is a member of the dataset that is used to obtain the model h. Then, if a deletion inference adversary wants to find out whether e is deleted from the model h to reach the model h_{del}, it can simply run $M(e, h_{\mathsf{del}})$ and output what it outputs. Note that there is no need to run $M(e,h)$, as the adversary of Definition 3.1 is given the promise that both e_0, e_1 *are* members of the initial dataset $\mathcal{S}$. Then the only question is how to combine the answers $M(e_0, h_{\mathsf{del}}), M(e_1, h_{\mathsf{del}})$, which Algorithm 3.5 decides in a natural way.

Algorithm 3.5 (From membership to deletion inference). Given examples $e_0 = (x_0, y_0)$, $e_1 = (x_1, y_1)$ and models h_{del}, the reduction from deletion inference to membership inference proceeds as follows:

1. Perform two membership inferences to obtain $b_0 = M(e_0, h_{\mathsf{del}})$ and $b_1 = M(e_1, h_{\mathsf{del}})$.
2. Return 0 if $b_0 = 0, b_1 = 1$, return 1 if $b_1 = 1, b_0 = 0$, and return a random bit if $b_0 = b_1$. ◇

Using confidence probabilities. An alternative reduction to Algorithm 3.5 can use the *confidence* probabilities of $M(e_0, h_{\mathsf{del}})$ and $M(e_1, h_{\mathsf{del}})$ instead of their final (rounded) values. In this variant, the reduction returns 0 if the confidence difference of $M(e_0, h_{\mathsf{del}}) - M(e_0, h)$ to output zero is more than the confidence difference of $M(e_1, h_{\mathsf{del}}) - M(e_1, h)$ to output zero.

3.3.2 Experiments with large data, and comparison with reduction to membership inference

We now evaluate our deletion inference attacks Del-Inf-Exm and Del-Inf-Ins on large dataset and large neural networks. In our experiment, we use CIFAR-10 and CIFAR-100 datasets [KH+09] as the training dataset, which are standard datasets for the evaluation of image classifiers, especially for deep learning models.

To better compare the success of our attacks with [SSSS17] we use a variant attack of Definition 3.1 in which *multiple* deletions happen (as explained in one of the variants following Definition 3.1). One advantage of this experiment setting is that the attack of [SSSS17] needs to train "attack models" for each victim model, and hence having multiple different deletions lead to multiple full training of attack models for [SSSS17] which is very expensive to run. However, in the multiple-deletion attack setting, one needs to only train the attack models of [SSSS17] twice to compare each execution of our attack with a reduction to [SSSS17].

Setting of our attack. The success probability is then calculated by taking the average over 20 rounds of full experiment. In each round of experiment, we first train a deep model with n examples, where n varies from $15,000, 20,000, 25,000$, and $29,540$ ($29,540$ is picked to match the scenario of [SSSS17]). We then randomly remove a batch of 100 examples in the training dataset, and train a new model without those 100 examples. As a reference, we pick another 100 random examples that remains in the dataset. The success probability is calculated over every pairs (in total, $10,000$ pairs) of the

Datasets →	**Iris**		**Wine**		**Breast Cancer**		**1/12 MNIST**	
Learning Method ↓	**Del-Inf-Exm**	**Del-Inf-Ins**	**Del-Inf-Exm**	**Del-Inf-Ins**	**Del-Inf-Exm**	**Del-Inf-Ins**	**Del-Inf-Exm**	**Del-Inf-Ins**
Logistic Regression	88.3%	86.8%	80.8%	76.1%	69.1%	60.6%	72.9%	56.6%
Decision Tree	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
SVM	70.5%	60.3%	76.9%	66.7%	73.8%	57.3%	72.3%	62.0%
Random Forest	89.2%	89.1%	83.3%	78.1%	89.2%	85.7%	89.9%	84.5%
MLP	92.9%	55.5%	54.2%	51.1%	83.5%	67.7%	62.5%	59.0%

Table 3. Success probabilities of the attacks Del-Inf-Exm and Del-Inf-Ins on classifiers.

deleted and reference examples, i.e., one deleted examples and one remaining example is given to the deletion inference adversaries Del-Inf-Exm and Del-Inf-Ins. We then measure the fraction of all pairs in which our adversary correctly predicts the deleted example. We evaluate our results on two deep neural network models: 1. A convolutional neural network that includes two convolutional layers (called smallCNN below), similar to the network used in [SSSS17]. 2. VGG-19 network (called VGG below) that has 19 layers in total, which is well-known for its power for image classification tasks.

Baseline settings for comparison. We compare our attacks with reductions to the membership inference attack in [SSSS17][6], i.e., reduction with label only and reduction with confidence probabilities.

Results. In Figure 1 and 2, we analyze the success probabilities of our deletion inference adversaries Del-Inf-Exm and Del-Inf-Ins on smallCNN model and VGG model. Our attack is able to correctly predict most of the deletions in the deep learning models, even when a batch of examples is deleted at the same time. Furthermore, note that for the membership inference attack of [SSSS17] to work, the adversary needs to have the label of the target instance and also make many queries to the target model for training an attack model (or many auxiliary data examples to train a similar model). On the other hand, our attack is extremely simple, and Del-Inf-Ins even does not require the label of the example.

Remark 3.6 (About using reduction to MI as baseline)**.** Here we comment on the limitations of membership inference as a baseline attack, as membership inference is not tuned to distinguishing between two points (one of which is guaranteed to be in the training set). Indeed, membership inference attackers only get only one instance as input, while our formalization of deletion inference gets two inputs. However, please note that we compare our deletion inference attackers to *reductions* to membership inference adversaries. The reduction is allowed to call the MI adversary *multiple* times. Indeed our reduction of the previous subsection calls the MI adversary twice, and this change makes the *reduction* to MI (which is a DI adversary itself) powerful enough to be able to win the DI inference game with probability close to 1, so long as its (regular) MI oracle wins its own game with probability close to 1.

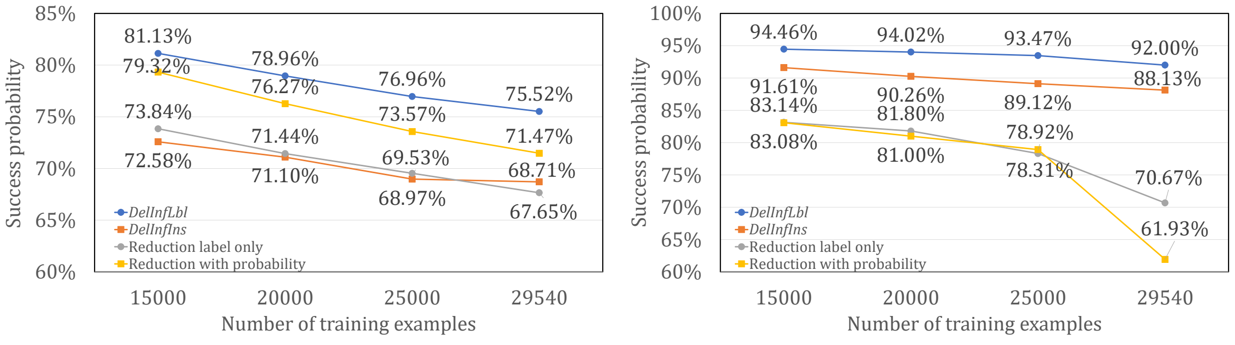
4 Deletion reconstruction

Section 3 focused on attacks that infer which of the two given examples is the deleted one. A more devastating form of attack aims to *reconstruct* the deleted example by querying the two models (before and after deletion). In this section, we show how to design such stronger attacks. We propose two types of reconstruction attacks on the deleted example. The first one focuses on reconstructing the deleted instance, while the second one focuses on reconstructing the deleted label. Both types of attacks follow the same security game which is explained in the definition below.

Definition 4.1 (Deletion reconstruction attacks)**.** Let Learn be a learning algorithm, Del be a deletion mechanism for Learn, and S_n be a distribution over $(\mathcal{X} \times \mathcal{Y})^n$. Consider the following game played between the adversary Adv and challenger Chal.

1. **Sampling the data and random selection.** Chal picks a dataset $\{e_1 \ldots e_n\} = \mathcal{S} \leftarrow S_n$ of size n. It also chooses $i \leftarrow [n]$ at random.
2. **Oracle access before deletion.** The challenger Chal trains $h \leftarrow \mathsf{Learn}(\mathcal{S})$. The adversary Adv is then

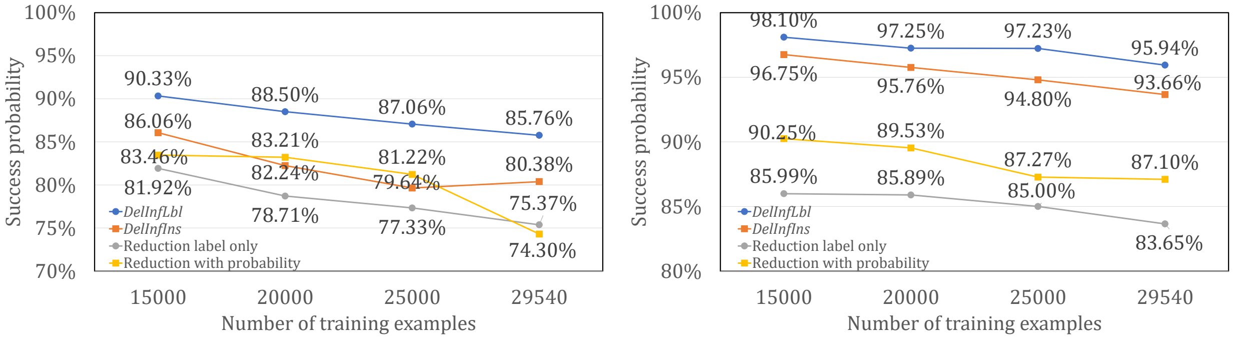
[6] We implemented [SSSS17] attack. [SSSS17] reports their membership inference attack achieves 71% success rate on a CNN model with two convolutional layers that is trained with CIFAR-10 dataset with 15,000 random examples. Our implementation of membership inference attack achieves 74% success rate on smallCNN model (which also has two convolutional layers) and 88% success rate on VGG model, which are trained on a subset of CIFAR-10 dataset with 15,000 random examples. The success rate matches the number reported in their work.



(a)

(b)

Fig. 1. Trend of success probabilities of attacks Del-Inf-Exm and Del-Inf-Ins on **smallCNN** models trained with different number of examples are shown; (a) uses dataset CIFAR-10 and (b) uses dataset CIFAR-100 dataset. The success probabilities are also compared with two baseline attacks that are obtained by reductions to the membership inference attack of [SSSS17].



(a)

(b)

Fig. 2. Trend of success probabilities of attacks Del-Inf-Exm and Del-Inf-Ins on **VGG** models trained with different number of examples are shown; (a) uses dataset CIFAR-10 and (b) uses dataset CIFAR-100 dataset. The success probabilities are also compared with two baseline attacks that are obtained by reductions to the membership inference attack of [SSSS17].

given *oracle* access to h. At the end of this step, the adversary instructs moving to the next step.

3. **Deletion.** The challenger obtains $h_{-e_i} \leftarrow \mathsf{Del}(h, e_i)$.
4. **Oracle access after deletion.** The adversary Adv is now given (only) *oracle* access to h_{-e_i}.
5. **Adversary's guess.** Adversary outputs a guess e.

For a similarity metric dis defined on $(\mathcal{X} \times \mathcal{Y})$, the adversary Adv is called a (ρ, ε)*-successful deletion reconstruction* attack if it holds that $\Pr[\mathsf{dis}(e, e_i) \leq \varepsilon] \geq \rho$. For bounded $\mathsf{dis}(\cdot, \cdot) \in [0, 1]$ and an adversary Adv, we define the *expected accuracy* of Adv as $1 - \mathbb{E}[\mathsf{dis}(e, e_i)]$. ◇

Deleted instance/label reconstruction attacks. One can use Definition 4.1 to capture attacks in which the goal of the adversary is to only (perhaps partially) reconstruct the instance x or the label y. In case of approximating x, we can use a metric distance dis that is only defined over $\mathcal{X}$ and ignores the labels of e and e_i. We refer to such attacks as *deleted instance reconstruction* attacks. Similarly, by using a proper metric distance defined only over $\mathcal{Y}$, we can use Definition 4.1 to obtain *deleted label reconstruction* attacks. Finally, to *completely* find e (resp. x or y) we use the 0-1 metric $\mathsf{dis}(e, e') = 1[e \neq e']$ (resp. $1[x \neq x']$ or $1[y \neq y']$).

Theorem 4.2 (From reconstruction to inference)**.** *Let* Learn *be a learning algorithm,* Del *be a deletion mechanism for* Learn*,* dis *be a distance metric over* $(\mathcal{X} \times \mathcal{Y})$*, and* S_n *be a distribution over* $(\mathcal{X} \times \mathcal{Y})^n$*. Suppose there is a* (ρ, ε)*-successful PPT reconstruction adversary against the scheme* (Learn, Del)*, and* $\Pr[\mathsf{dis}(e_0, e_1) > 2\varepsilon] \geq 1 - \delta$ *where the probability is over sampling* e_0, e_1 *from the* *sampled dataset* $\mathcal{S} \leftarrow S_n$.[7] *Then,* $(\mathsf{Learn}, \mathsf{Del})$ *is* $(\rho - \delta)$*-insecure against deletion inference over distribution* S_n.

Proof of Theorem 4.2. We give a polynomial time reduction. In particular, suppose B is a (black-box) adversary that shows the (ρ, ε) insecurity of the scheme $(\mathsf{Learn}, \mathsf{Del})$ against deletion reconstruction attacks. We design an adversary Adv against deletion inference (as in Definition 3.1) as follows. Given (e_0, e_1) as challenges, first ignore (e_0, e_1) and using oracle access to models h, h_{del}, run B to obtain e as approximation of the deleted example. Output 0 if $\mathsf{dis}(e_0, e) \leq \varepsilon$, else output 1 if $\mathsf{dis}(e_1, e) \leq \varepsilon$, otherwise output uniformly in $\{0, 1\}$.

We now analyze the reduction above. With probability at least ρ over the execution of the attack B, it holds that $\mathsf{dis}(e, e_b) \leq \varepsilon$, where e_b is the deleted example. Also, with probability $1 - \delta$ it holds that $\mathsf{dis}(e_0, e_1) > 2\varepsilon$. By a union bound, we have that with probability at least $\rho - \delta$ both of the conditions above happen at the same time, in which case the adversary Adv outputs the correct answer b. □

Due to the theorem above, all the reconstruction attacks below can be seen as strengthening of deletion inference attacks.

4.1 Experiments: Deletion reconstruction of instances for nearest neighbor

In this experiment, we consider a classification clustering task in high dimension. The previous work [Fel20, BBF+21] studied the same setting and showed that machine learning models sometimes need to memorize their training set in order to learn with high accuracy. In this setting, we extend the attacks of [Fel20, BBF+21] into two directions to obtain deletion reconstruction attacks: (1) we obtain polynomial time attacks that extract instances rather than proving mutual information between the model and the examples, (2) we show a setting where the extraction is enabled after the *deletion.*

Roadmap and the leakage of the deletion. We develop polynomial-time reconstruction attacks that crucially leverage the deletion operation. However, in order to analyze our attacks, we first *limit* ourselves to the so-called *singleton* setting in which each label appears at most once for an example in the dataset (Section 4.1.1). Focusing on this case allows us to provide theoretical ideas that support our attacks. However, our attacks in the singleton case are also able to extract instances *even* without deletion. Hence, in the singleton case, our attacks can be seen as leakage of the model h itself, *even without deletion.* Note that such attacks can still be used for deletion reconstruction, they do not reflect the *extra leakage* of the deletion operation. Nevertheless, we next experimentally show (Section 4.1.2) that virtually the same polynomial-time attacks succeed even when the labels are not unique on the real world dataset Omniglot. In particular, when we have many repeated labels (perhaps even as neighbor cells), then our simpole attacks do *not* extract the instances from access to either of h, h_{del}, and it is needed to have access to *both* models to find the "vanished" Voronoi cell before extracting the center of the cell.

We now our polynomial-time deletion reconstruction attack for the case of 1-nearest neighbor models. We work with instance space $\mathcal{X} = \{0, 1\}^d$.[8] We also assume the learner Learn runs a 1-nearest neighbor algorithm. Namely for $h = \mathsf{Learn}(\mathcal{S})$ where $\mathcal{S} = \{(x_1, y_1) \ldots (x_n, y_n)\}$, we have $h(x) = y_j$ where $j = \operatorname{argmin}_i \mathsf{dis}(x, x_i)$.

We propose the following attack $\mathsf{Del\text{-}Ins\text{-}Rec}$ that aims to reconstruct the deleted instance x_i.

Algorithm 4.3 (Attack $\mathsf{Del\text{-}Ins\text{-}Rec}$)**.** Suppose the adversary is given oracle access to h followed by oracle access to h_{del}, along with an auxiliary set of instances $\mathcal{T}$, $|\mathcal{T}| = m$. (For example, $\mathcal{T}$ could simply be m independent samples different from the original training set $\mathcal{S}$.) The attack then proceeds as follows:

- For all $x \in \mathcal{T}$ query the model h.
- Then for all $x \in \mathcal{T}$, query the model h_{del}.
- Create the set of points in the "deleted region": $\mathcal{T}' = \{x \mid h(x) \neq h_{\mathsf{del}}(x), x \in \mathcal{T}\}$.
- Return the majority for each coordinate; namely, return $x = (b'_1, \ldots, b'_d)$, where $\forall i \in [d]$,

$$b'_i = \operatorname*{argmax}_{b \in \{0,1\}} \sum_{(b_1, \ldots, b_d) \in \mathcal{T}'} 1[b_i = b]. \quad \diamond$$

Intuition behind the attack. The intuition behind the attack of Algorithm 4.3 is that instances like x whose prediction label changes during the deletion process should belong to the Voronoi cell centered at x_i, where (x_i, y_i) is the deleted example. Then the algorithm

[7] For example, when S_n consists of n i.i.d. samples from D, e_0, e_1 are simply two independent samples from D.

[8] We use binary features because it is more general and that other features can also be represented in the form of binary strings.

heuristically assumes that when we pick x at random *conditioned* on changed labels, then they give a pseudo-random distribution inside the Voronoi cell of x_i. In the next section we show that for a natural case called singletons, in which the labels are unique, this intuition carries over formally. We then experimentally verify our attack for the general case (when labels can repeat) on a real data set.

4.1.1 Theoretical analysis for uniform singletons

In this section, we focus on a theoretically natural case to analyze the attack of Algorithm 4.3. We refer to this case as the *uniform singletons* which is also studied in [Fel20, BBF+21] and is as follows. First, we assume that instances are uniformly distributed in $\{0,1\}^d$, and secondly, we assume that the labels are unique (i.e., without loss of generality, the labels $y_1,\ldots,y_n$ are just $1,\ldots,n$). The following lemma shows that in this case, the attack of Algorithm 4.3 never converges to wrong answers for *any* coordinate of the instances.

Lemma 4.4 (Non-negative correlations)**.** *Let* $\mathcal{S} = \{x_1,\ldots,x_n\}$ *where* $\forall i, x_i \in \{0,1\}^d$*, and suppose* $h(x) = \operatorname{argmin}_i \mathsf{dis}(x,x_i)$*, and we break ties by outputting the smallest index* i*, if multiple nearest neighbors exist. Suppose* $C_i = \{x \mid h(x) = i\}$ *be the Voronoi cell centered at* x_i*. Let* $x[j]$ *be the* j*'th bit of* x*. Then, for every* $i \in [n]$ *and every* $j \in [d]$*, we have*

$$\Pr_{x \leftarrow C_i}[x[j] = x_i[j]] \geq \frac{1}{2}.$$

Proof of Lemma 4.4. Let $C_i^{j,b} = \{x \in C_i \mid x[j] = b\}$ be the subset of C_i that has b in its j'th coordinate.

We claim that by flipping the j'th bit of every $x \in C^{j,1-x_i[j]}$, we obtain a vector $x' \in C^{j,x_i[j]}$. The reason is as follows. (1) By definition, the j'th bit of x' is indeed $x_i[j]$. (2) It holds that $h(x') = i$, which means $x' \in C_i$. The reason for (2) is that, by flipping the j'th bit of x, x' gets one step *closer* to x_i compared to how far x was from x_i. Therefore, if x_i was the nearest neighbor of x, it would also be the nearest neighbor of x' as well. A boundary case occurs if multiple points are the nearest points of x, but the same tie breaking rule still assigns x_i as the nearest neighbor of x'. Since the mapping from x to x' is injective, it also gives an injective mapping from $\left|C_i^{j,1-x_i[j]}\right|$ to $\left|C_i^{j,x_i[j]}\right|$. This proves that

$$\left|C_i^{j,x_i[j]}\right| \geq \left|C_i^{j,1-x_i[j]}\right|,$$

which is equivalent to $\Pr_{x \leftarrow C_i}[x[j] = x_i[j]] \geq 1/2$. □

4.1.2 Deleted image reconstruction for 1-NN

We now show that the simple attack of Algorithm 4.3 can be used to reconstruct visually recognizable images even when the distribution is not normal and labels are not unique. Hence, we conclude that the actual power of this attack goes beyond the theoretical analysis of the previous section. We use the Omniglot [LST15] dataset, a symbol classification dataset specialized for few-shot learning. The dataset includes handwritten symbols from multiple languages.

Experiment details. We binarize each pixel of the dataset to remove the noise in gray-scale. The input space is $\mathcal{X} = \{0,1\}^d$, where $d = 11025$ is the number of pixels. We assume the Omniglot dataset is divided into two parts: (1) a training subset which contains 140 symbols from 30 different languages. The languages serve as the class label in the dataset in our experiments,[9] and (2) a fixed test set with another 140 examples from each language which is provided to the adversary as auxiliary information. The learning algorithm Learn is the 1-nearest neighbor predictor, which for a dataset $\mathcal{S}$ always returns the label (i.e., the language) of the nearest example in the dataset $h(x') = \operatorname{argmin}_y\{\mathsf{dis}(x,x') \mid (x,y) \in \mathcal{S}\}$. We use Algorithm 4.3 as the attack, which simply takes majority on each pixel over the instances that fall into the disagreement region of the two models (before and after deletion). We run the security game of Definition 4.1 with 100 random images from the dataset as the deleted image.

Comparison with reconstruction attacks without deletion. As a comparison to further highlight the leakage that happens due to the deletion, we also run a similar reconstruction attack *without* deletion. Suppose for a moment that labels were unique. Then, to reconstruct instance x, the attacker aims to extract the image x from the data set with label y, where y is the label of x. To do that, the reconstruction attacker can run the same exact attack as our deletion reconstruction, as follows: it tests all the images in the test dataset on the model and records every image with label y. The attacker then generate a reconstruction image by taking the majority of the images with label y on every pixel.

[9] Note that in the original dataset, the labels reflect the character, but to demonstrate the leakage of *deletion* rather than the mere leakage of datasets alone, we use the labels that represent the languages to increase the frequency of the labels.

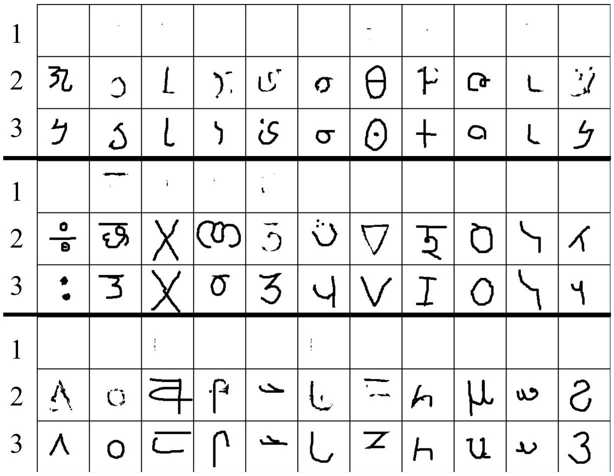


Fig. 3. 33 reconstruction examples on Omniglot dataset. In the figure, Row 1 is the result from the attack without deletion, Row 2 is the result from the deletion reconstruction attack, and Row 3 is the deleted example, which is the target of the attack.

When the labels are unique, this reconstruction attack can reconstruct the instances used by a 1-NN just like how our deletion reconstruction attack does and succeeds. However, in our case labels are not unique. Hence, we use this attack as the baseline to show how much our deletion inference attack is in fact extracting information that is the result of the deletion operation.

The result of our deletion reconstruction and the baseline (non-deletion) reconstruction attacks are shown in Figure 3. Our deletion reconstruction algorithm reconstructs 40 out of 100 images, due to page limit, 33 of them is shown in Figure 3. As is clear from the pictures, the non-deletion reconstruction attack gives no meaningful result in our setting. More concretely, for 35 of the 40 images the label of the deletion reconstruction attack obtains the the correct label when fed back into the nearest neighbor classifier, while only 1 of the images generated by the attack without deletion obtains the correct label.

5 Weak deletion compliance

In Sections 3 and 4, we studied *attacks* on data privacy under data deletion. The definitions of those sections provide *weak* guarantees on what adversary cannot do, hence they are suitable for stronger *negative* results. In this section, we investigate the other side; namely, positive results that can prevent attacks of Sections 3 and 4 and provide strong guarantees about what adversary can(not) learn about the data that is being updated through deletion requests. In particular, we observe that the deletion compliance definition of Garg, Goldwasser and Vasudevan [GGV20] would prevent attacks of Sections 3 and 4. More precisely, we show that even a *weaker* variant of the [GGV20] definition would prevent the attacks of Sections 3 and 4.

Components of deletion compliance definition. The "deletion compliance" framework of Garg, Goldwasser and Vasudevan [GGV20] provides an intuitive way of capturing data deletion guarantees in general systems that collect and process data. This framework models the world by three interacting parties – the data collector DatCol, the deletion-requester DelReq, and the environment Env. All components are the same as those of [GGV20], however, we will work with a modified DelReq and a different indistinguishability guarantee.

- *Data collector (learner)* DatCol represents the algorithm that collects the records (training examples) and processes data according to a (learning) mechanism. For example DelReq might accept up to n data storage requests and up to k data deletion requests.
- *Deletion requester (user)* DelReq is a special honest user who only stores two particular examples e_0, e_1 and will delete one of them later. The timing of such requests are stated below. In the original [GGV20], the deletion requester just stores *one* record e and delete it, or that it might never store e in the first place. At a high level, their DelReq is designed so that one can define privacy that even hides the deletion itself, while our variant is designed for a weaker definition that does not hide the deletion itself.
- *Environment (adversary)* Env models the "rest of users" who might not be honest and who are interested in finding out what DelReq is deleting. The interaction between Env and DatCol, DelReq is defined by the interfaces of DatCol, DelReq.

Interaction of the components. We let $\mathcal{U}$ model a *universe of records*. For example, $\mathcal{U} = \mathrm{Supp}(D)$ for a distribution over labeled examples $\mathcal{X} \times \mathcal{Y}$. We now describe the restrictions on how the components interact with each other. Other than the below-mentioned restrictions, the parties run in PPT.

- DatCol accepts instructions $\mathsf{Add}(e), \mathsf{Del}(e), \mathsf{Eval}(x)$. The interpretation of these instructions are as follows. $\mathsf{Add}(e)$ adds the record $e \in \mathcal{U}$ to the set of records stored at DatCol. $\mathsf{Del}(e)$ removes e from the set stored by the data collector, and $\mathsf{Eval}(x)$ returns the evaluation of the "current model" stored by DatCol (which is the result of learning over the set stored at DatCol) on x and returns the answer.

- As in [GGV20], we also require that only Env can send messages to DelReq. At some point in the execution of the system Env sends DelReq the following messages, which is followed by messages from DelReq to DatCol as described below.
  1. (Add, e_0, e_1): DelReq sends $\mathsf{Add}(e_0), \mathsf{Add}(e_1)$ to DatCol.
  2. Del: DelReq will send $\mathsf{Del}(e)$ to DatCol where $e \in \{e_0, e_1\}$. By DelReq_b we refer to the instantiation of DelReq that sends $\mathsf{Del}(e_b)$ to DatCol.

Weak deletion compliance. For our purposes, we consider a different weaker definition (compared to that of [GGV20]) that still captures all attacks of Section 3 and 4. To start, we define two worlds, World 0 and World 1, corresponding to the instantiation of DelReq by DelReq_0 and DelReq_1.

Definition 5.1 (Weak deletion compliance)**.** Let the interactive algorithms DatCol, Env, DelReq be, in order, the data collector, the environment, and the deletion requester (interactive) algorithms limited to interact as described above. We call DatCol ε *deletion compliant*, if no PPT Env can detect whether it is in World 0 (with DelReq_0) or World 1 (with DelReq_1) with advantage more than ε. If this holds under the restriction that Env makes at most $(k-1)$ deletion requests during the execution, then DatCol is said to be ε-weak deletion-compliant for up to k deletions ◇

Comparison with [GGV20]. The key differences between our Definition 5.1 and that of [GGV20] are as follows. In each case, we state the property of our definition in contrast to that of [GGV20].

- **Hiding the state of DatCol from adversary.** The definition of [GGV20] focuses on scenarios where the data collector's state might be revealed at some point in the future (e.g., due to a subpoena). However, in this work we focus on hiding the information that is *leaked* from the data collector (about deleted record) through interaction with the adversary.
- **Not aiming to hide the deletion itself.** Whereas plain deletion-compliance asks that deletion make the world look as though the deleted data were never present in the first place, here we only ask that it not be revealed *which* record was deleted. For instance, a data collector that is weak deletion-compliant might still reveal the number of deletions it has processed, as long as the data that is deleted is not revealed. While weaker than deletion-compliance definition of [GGV20], our notion is fit for hiding the deleted record among the records in the training set, and still giving a more general and stronger definition than Definition 3.1.

We now formally discuss why Definition 5.1 captures the attacks of Section 3 and 4. Recall that Definition 3.1 was already shown in Theorem 4.2 to be a stronger notion than instance and label reconstruction attacks (Definition 4.1). Hence, we just need to show that Definition 5.1 is stronger than Definition 3.1.

Theorem 5.2 (Deletion inference from compliance)**.** *Let* Learn *be a learner,* Del *be a deletion mechanism for* Learn*, D be a distribution over labeled examples, and $\mathcal{U} = \mathrm{Supp}(D)$ be the universe of records. The data collector* DatCol *answers queries as follows.*

1. DatCol *does not respond any* Del *or* Eval *queries till receiving n* $\mathsf{Add}(\cdot)$ *queries, which we refer to as $\mathcal{S}$.*
2. DatCol *permutes $\mathcal{S}$ and gets $h \leftarrow \mathsf{Learn}(\mathcal{S})$.*
3. *Then it answers $\mathsf{Eval}(e) = h(e)$ queries arbitrarily.*
4. *Then it accepts* one $\mathsf{Del}(e)$*, and lets $h_{-e} = \mathsf{Del}(h, e)$.*
5. *Then it continues answering $\mathsf{Eval}(e) = h(e)$ queries.*

If DatCol *is $(2\varepsilon - 1)$-deletion compliant (as in Definition 5.1) against PPT adversaries with oracle access to D, then the scheme* (Learn, Del) *is ε-secure against deletion inference (as in Definition 3.1).*

Proof of Theorem 5.2. We give a proof by reduction. Suppose Adv breaks the membership inference security game of Definition 3.1 with probability $(1+\varepsilon)/2$. We construct an environment Env that ε-distinguishes DelReq_0 from DelReq_0 with advantage ε that proceeds as follows:

1. Env plays the role of the challenger from Definition 3.1 and picks a data set $\{e_1, \dots, e_n\} = \mathcal{S} \leftarrow S_n$ of size n. Env passes this to the DatCol and picks $i \neq j \in [n]$ at random as the challenge records.
2. Next, Env instantiates Adv and provides it with the records e_i, e_j and *oracle* access to h (through DatCol). At the end of this step, the adversary instructs moving to the next step.
3. Env passes (e_i, e_j) to DelReq (which will then request the deletion of one of the two records).
4. Env actives the Adv again and it is again provided *oracle* access to h (through DatCol). At the end of this step, the adversary's output is included in the output of the environment.

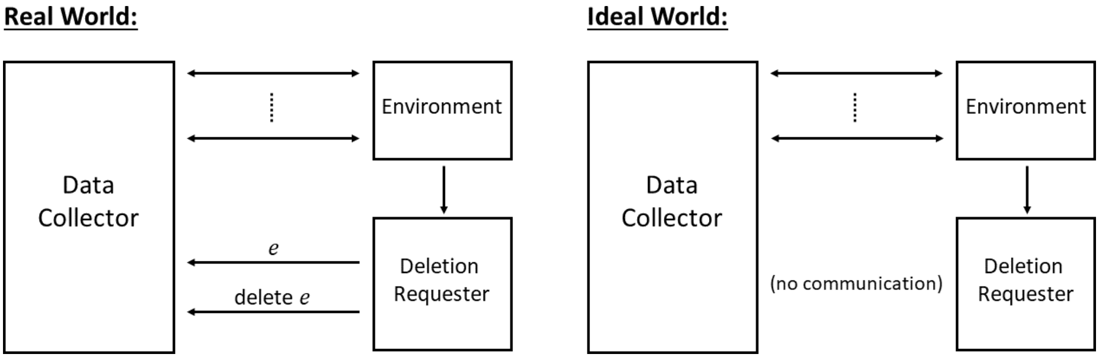


Fig. 4. The real and ideal worlds for (strong) deletion compliance

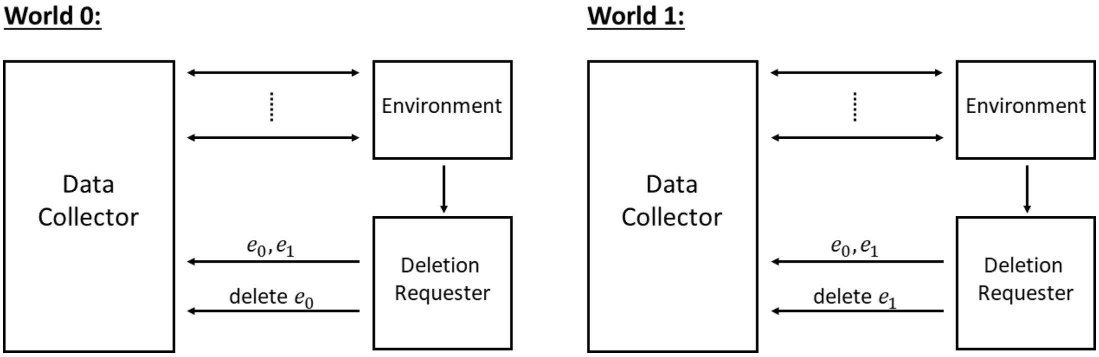


Fig. 5. The worlds for weak deletion-compliance

The view of the adversary Adv in the above experiment is identical to its view as part of Definition 3.1. Thus, the output of Adv will correctly (with probability greater than ϵ) identify whether DelReq requests the deletion of record e_i or record e_j. This allows us to conclude that the view of the Env changes depending of whether DelReq requests deletion of e_i or e_j. □

Using the same three components described in Section 5 (with a different DelReq), [GGV20] defines the notion of *deletion-compliance*. Here the ideal world is the same as the real world in all respects except that DelReq is not allowed to communicate with DatCol as represented in Fig. 4. (The restriction of DelReq not being able to send messages to Env was imposed in order for this ideal world to be well-defined, by excluding cases where Env sends to DatCol messages that depend non-trivially on DelReq's records.) [GGV20] calls DatCol to be ε-deletion-compliant if, for any Env and DelReq, the joint distributions of the state of DatCol and view of Env in the real and ideal world are ε-close in the statistical distance, denoted by notation $\approx_\varepsilon$. That is,

$$(state_D^{\mathsf{real}}, view_E^{\mathsf{real}}) \approx_\varepsilon (state_D^{\mathsf{ideal}}, view_E^{\mathsf{ideal}}).$$

The above (strong) definition from [GGV20] captures the intuition that a system is deletion-compliant if the state of the world after its deleting a record is similar to what it would have been if the record had never been part of the system in the first place. Note that this requirement of ε-closeness in statistical distance is more relaxed than the kind of closeness of distributions required by differential privacy, and so DP can be used to satisfy these requirements. [GGV20] showed how to obtain their strong deletion compliance based on differentially private mechanisms.

To contrast with Figure 4, in Figure 5 we have depicted the more symmetric worlds that are behind our Definition 5.1. In particular, Definition 5.1 requires that no PPT Env can distinguish between World 0 and World 1 of Figure 5 by more than advantage ε.

Acknowledgements

Ji Gao and Mohammad Mahmoody were supported by NSF grants CCF-1910681and CNS-1936799. Sanjam Garg was supported in part by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. Prashant Nalini Vasudevan was supported by funds from an NUS Presidential Young Professorship. Part of this work was done while he was a postdoctoral researcher at UC Berkeley supported by Sanjam Garg's mentioned funds and the UC Berkeley Center for Long-Term Cybersecurity.

References

[ACD94] Stefan Aeberhard, Danny Coomans, and Olivier De Vel. Comparative analysis of statistical pattern recognition methods in high dimensional settings. *Pattern Recognition*, 27(8):1065–1077, 1994.

[BBF+21] Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 123–132, 2021.

[BBHM16] Michael Backes, Pascal Berrang, Mathias Humbert, and Praveen Manoharan. Membership privacy in microrna-based studies. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 319–330, Vienna, 2016. ACM.

[BBKN14] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. *Machine learning*, 94(3):401–437, 2014.

[BCCC+20] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2020.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473, Philadelphia, 2014. IEEE, IEEE.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CLE+19] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, Santa Clara, 2019. USENIX Symposium.

[CN20] Aloni Cohen and Kobbi Nissim. Towards formalizing the gdpr's notion of singling out. *Proceedings of the National Academy of Sciences*, 117(15):8344–8352, 2020.

[Con09] International Warfarin Pharmacogenetics Consortium. Estimation of the warfarin dose with clinical and pharmacogenetic data. *New England Journal of Medicine*, 360(8):753–764, 2009.

[CTCP20] Christopher A Choquette Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-only membership inference attacks, 2020.

[CY15] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, San Jose, 2015. IEEE, IEEE.

[CZW+21] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 896–911, 2021.

[DBB18] Ashish Dandekar, Debabrota Basu, and Stéphane Bressan. Differential privacy for regularised linear regression. In *International Conference on Database and Expert Systems Applications*, pages 483–491, Regensburg, 2018. Springer, Springer.

[dlT18] Lydia de la Torre. A guide to the California consumer privacy act of 2018. *Available at SSRN 3275571*, 2018.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 202–210, 2003.

[DSS+15] Cynthia Dwork, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. Robust traceability from trace amounts. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 650–669. IEEE, IEEE, 2015.

[DSSU17] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application*, 4:61–84, 2017.

[DTTZ14] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20, 2014.

[Dwo08] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[EHJ+04] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[Fel20] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[Fis36] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[FLJ+14] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014.

[GAS19] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep neural networks. *arXiv preprint arXiv:1911.04933*, 2019.

[GGHvdM19] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens van der Maaten. Certified data removal from machine learning models. *arXiv preprint arXiv:1911.03030*, 2019.

[GGV20] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. Formalizing data deletion in the context of the right to be forgotten. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 373–402. Springer, 2020.

[GGVZ19] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. Making ai forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems*, pages 3513–3526, 2019.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[HAR78] DAVID HARRISON. Hedonic housing prices and the demand for clean air. *JOURNAL OF ENVIRONMENTAL ECONOMICS AND MANAGEMENT*, 5:81–102, 1978.

[HSR+08] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS Genet*, 4(8):e1000167, 2008.

[HvdSB19] Chris Jay Hoofnagle, Bart van der Sloot, and Frederik Zuiderveen Borgesius. The european union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, 2019.

[ISCZ20] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models: Algorithms and evaluations. *arXiv preprint arXiv:2002.10077*, 2020.

[JWEG20] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Revisiting membership inference under realistic assumptions. *arXiv preprint arXiv:2005.10881*, 2020.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBG17] Yunhui Long, Vincent Bindschaedler, and Carl A Gunter. Towards measuring membership privacy. *arXiv preprint arXiv:1712.09136*, 2017.

[LBW+18] Yunhui Long, Vincent Bindschaedler, Lei Wang, Diyue Bu, Xiaofeng Wang, Haixu Tang, Carl A Gunter, and Kai Chen. Understanding membership inferences on well-generalized learning models. *arXiv preprint arXiv:1802.04889*, 2018.

[LST15] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[LZ20] Zheng Li and Yang Zhang. Label-leaks: Membership inference attack with label, 2020.

[MSM93] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[NBW+17] Kobbi Nissim, Aaron Bembenek, Alexandra Wood, Mark Bun, Marco Gaboardi, Urs Gasser, David R O'Brien, Thomas Steinke, and Salil Vadhan. Bridging the gap between computer science and legal approaches to privacy. *Harv. JL & Tech.*, 31:687, 2017.

[NRSM20] Seth Neel, Aaron Roth, and Saeed Sharifi-Malvajerdi. Descent-to-delete: Gradient-based methods for machine unlearning. *arXiv preprint arXiv:2007.02923*, 2020.

[NS06] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 427–437, 1990.

[RGC15] Mauro Ribeiro, Katarina Grolinger, and Miriam A M Capretz. Mlaas Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 896–902. IEEE, 2015.

[SBB+20] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 1291–1308, 2020.

[SCS13] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[She19] Or Sheffet. Old techniques in differentially private linear regression. In *Algorithmic Learning Theory*, pages 789–827, 2019.

[SKS+06] Todd E Scheetz, Kwang-Youn A Kim, Ruth E Swiderski, Alisdair R Philp, Terry A Braun, Kevin L Knudtson, Anne M Dorrance, Gerald F DiBona, Jian Huang, Thomas L Casavant, et al. Regulation of gene expression in the mammalian eye and its relevance to eye disease. *Proceedings of the National Academy of Sciences*, 103(39):14429–14434, 2006.

[SOJH09] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965–967, 2009.

[SRS17] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 587–601, Dallas, 2017. ACM.

[SSSS17] 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, San Jose, 2017. IEEE, IEEE.

[SWM93] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. International Society for Optics and Photonics, 1993.

[SZH+19] Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Network and Distributed Systems Security Symposium 2019*. Internet Society, 2019.

[TTZ15] Kunal Talwar, Abhradeep Guha Thakurta, and Li Zhang. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems*, pages 3025–3033, 2015.

[VBE18] Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: Model inversion attacks and data protection law. *CoRR*, abs/1807.04644, 2018.

[WFJN16] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370, Lisbon, 2016. IEEE, IEEE.

[YGFJ18] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, Oxford, 2018. IEEE, IEEE.

[YZCL19] Ziqi Yang, Jiyi Zhang, Ee-Chien Chang, and Zhenkai Liang. Neural network inversion in adversarial setting via background knowledge alignment. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 225–240, London, 2019. ACM.

[ZBWT+20] Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olge Ohrimenko, Boris Köpf, and Marc Brockschmidt. Analyzing information leakage of updates to natural language models. In *ACM Conference on Computer and Communication Security (CCS)*. ACM, ACM, November 2020.

A Hyperparameters of models

Here we describe the hyperparameters of the models used in the experiments of our paper.

- **MLP:** We use multiple layer perceptron with two hidden layers. For regression, we set the size of hidden layers as $(20, 2)$, and for classification, we set the size of hidden layers as $(20, 10)$. The reason behind is that the output layer of classification tasks have more neurons. We use LBFGS as the optimization algorithm to train the model, and we train 200 epochs on each model.
- **SVM:** We use the default SVMClassifier and SVMRegressor in Scikit-learn. Specifically, we use the RBF kernel with $C = 1.0$.
- **Decision tree:** For the decision tree model, we use the default DecisionTreeClassifier and DecisionTreeRegressor in Scikit-learn. Specifically, we use Gini impurity to split the leafs and do not set a limit on the tree size.
- **Random forest:** We use the default RandomForestClassifier in Scikit-learn, which generates 10 trees in the forest. For each tree, its hyperparameter is the same with the decision tree classifier above.
- **Logistic regression and Linear regression:** We use the default LinearRegression and LogsticRegression from Scikit-learn.
- **Lasso regression:** We set $\alpha = 0.1$ in lasso regressor.

B Comparison with reduction to the MI attack of [YGFJ18]

Additionally, here we compare our direct deletion inference attack Del-Inf-Exm with a reduction (as stated in Algorithm 3.5) to membership inference attacks of Yeom et al. [YGFJ18]. The membership inference adversary of Yeom et al. [YGFJ18] works with a threshold value that is chosen to be the average of losses over all the training examples as an auxiliary information. This threshold can be estimated by the adversary from extra examples sampled from the distribution. The adversary predicts an example to be a member of the model if the loss of that example is smaller than the average training loss. We use the original code, models, and data released by [YGFJ18]. Specifically, we use two datasets IWPC (International Warfarin Pharmacogenetics Consortium) [Con09] and Eyedata [SKS+06] from [YGFJ18]. The statistics of the datasets are summarized in Table 1. Regarding the hyperparameters, we report the success probability of both attacks on the model with best performance without attack, which is selected via cross validation. The experiment's results are shown in Table 4.

Our simple adversary algorithms achieve around 90% success probability in almost all cases, in both settings of comparing with samples in the training set (inside $\mathcal{S}$) or comparing with samples outside the training dataset (outside $\mathcal{S}$). In most cases, the success in fact close to 100%. Clearly, with the access of both versions of models, there exists an obvious privacy leak that can be easily exploited by naive adversaries.

	Reduction to Mem. Inf.	Del-Inf-Exm
Eyedata	**64.80%**	**97.54%**
IWPC	**68.83%**	**95.98%**

Table 4. Comparing the success probabilities of our attack Del-Inf-Exm and an alternative attack based on reduction to the membership inference attack of Yeom et al. [YGFJ18].

C Deleted label reconstruction

We now show that when the dataset is small, the label y for the deleted example $e = (x, y)$ might completely leak through a black-box access to the models before and after the deletion. Note that when y is binary, there is little difference between label *inference* and *reconstruction*, but our attacks work even when the labels are not binary, hence it is suitable to call them label reconstruction attacks as defined in Definition 4.1.

We propose the following attack Del-Lbl-Rec to reconstruct the deleted label.

Algorithm C.1 (Attack Del-Lbl-Rec)**.** Given models h and h_{del}, a number $n \in \mathbb{N}$, the label inference attacker Del-Lbl-Rec proceeds as follows:

```
1. Randomly pick $m$ random samples $\mathcal{T} = \{x_1, x_2, \ldots, x_m\}$ in the data range $\mathcal{X}$.
2. For all $i \in [m]$, query $h$ to obtain $\hat{y}_i = h(x_i)$.
3. For all $i \in [m]$, query $h_{\mathsf{del}}$ to obtain $\hat{y}'_i = h_{\mathsf{del}}(x_i)$.
4. Return $\mathrm{argmin}_{c \in \mathcal{Y}} \sum_{i=1}^{m} (\Pr[\hat{y}'_i = c] - \Pr[\hat{y}_i = c])$. ◇
```

The intuition is that for natural models (e.g., ERM rule), removing one example of a specific class will tend to move the prediction towards other classes, i.e., the expectation of predictions to that specific label is likely to decrease. In other words, if the attack above fails, it means that adding this deleted sample back to the training set will let the model tend to predict other classes, which is an unlikely scenario. Our experiments confirm that this attack intuition succeeds.

Experiment details. We test the attack on three classic classification datasets, including the Iris Dataset [Fis36], the Wine Recognition dataset [ACD94], and the Breast Cancer Wisconsin Diagnosis dataset [SWM93]. The label is among a discrete set $\mathcal{Y}$. The learning algorithms are the logistic regression model and K-Nearest Neighbor model. The experiment result is presented in Table 5. The success probability of the attack is higher than 90% on the Iris and Wine datasets, and is higher than 75% on Breast Cancer dataset.

	Iris	Wine	Breast Cancer
Logistic Regression	**92.90%**	**97.30%**	**86.60%**
K Nearest Neighbor	**93.70%**	**90.10%**	**77.80%**

Table 5. Results of our deleted label reconstruction attacks

C.1 Known-instance label reconstruction

In this section, we study attacks in which the adversary knows the instance x of the deleted record $e = (x, y)$ and wishes to approximate the true label y by querying the models h and h_{del}. The goal is to beat the correctness of both models for true label y. This means that, in case the two models were supposed to hide the label (perhaps if it was a sensitive information to know very precisely) the data removal process, in this case, clearly goes against the goal of hiding y in its exact form.

Definition C.2 (Known-instance label reconstruction)**.** This definition is identical to Definition 4.1 with the only difference that the adversary is now given x_i (but not y_i) in Step 2 of the attack. ◇

Even though one can define the success criteria of the attackers of Definition C.2 the same way as those of Definition 4.1, such attacks are only interesting if they can beat the precision of the answers provided by the two models h, h_{del}, as anyone (including the adversary) could query those models on the point x_i, once x_i is revealed. Our experiments show that such "accuracy boosting" attacks are indeed sometimes possible in the presence of deletion operations.

We propose a simple attack Ins-Rev-Lbl-Rec in Construction C.3 below. Ins-Rev-Lbl-Rec makes an estimation on y based on the output of the two models.

Algorithm C.3 (Attacker Ins-Rev-Lbl-Rec)**.** This attack is parameterized by $\lambda > 0$. Given sample x, models h and h_{del}, and a constant λ, the label reconstruction adversary Ins-Rev-Lbl-Rec proceeds as follows:

1. Query to obtain $\hat{y} = h(x)$ and $\hat{y}' = h_{\mathsf{del}}(x)$.
2. Return $\tilde{y} = \hat{y} + \lambda \cdot (\hat{y} - \hat{y}')$. ◇

Intuition behind the attack. Similar to the attacks of Section 3 (see Proposition 3.3), the loss of the deleted sample will increase after the deletion. For simplicity, suppose the loss is mean squared error. In this case, when the learner follows the ERM rule, we have $|\hat{y}' - y|_2 \geq |\hat{y} - y|_2$. Therefore, moving from $\hat{y}'$ towards $\hat{y}$ makes the prediction closer to the actual label y. Consequently, using a small positive λ could lead to less loss. The best value of λ in each different scenario could be empirically estimated by a similar size dataset that is individually sampled by the attacker.

Experiment details. We perform the attack on linear regression models. We test the attack on two classic regression datasets, the Boston Housing Price Dataset [HAR78] and the diabetes dataset [EHJ+04]. For each dataset, we train the model h with the whole dataset. The adversary returns an approximation $\tilde{y}$. $|\tilde{y} - y|_2$ will denote the distance of the prediction by the adversary, and we use $\min(|h(x) - y|_2, |h_{\mathsf{del}}(x) - y|_2)$ as the baseline value to compare the quality of adversary's prediction.

Results. We calculate the average distance of $\tilde{y}_i$ and y_i with different λ values. Our results (in Table 6) show that there exists a λ value for each dataset, such that can reduce the the estimated loss by around 70%.

	Best λ	Models	Adversary	%
Boston	17.5	21.897	7.149	30%
Diabetes	30	2859.7	829.8	28%

Table 6. Result of the label reconstruction Attack on Logistic Regression. The column Models lists the average of the minimum distance of the predictions of the two models h, h_{del}. The column Adversary lists the average distance of the prediction of the adversary and the real prediction, and the percentage shows the percentage of the improvement in the prediction compared with the better of the predictions of the two models h, h_{del}.

D Deleted sentence reconstruction

In this experiment we perform reconstruction attacks on sequential text data. Namely, we show how to extract the deleted sentence by querying a language model according to the security game of Definition 4.1.

We start by giving formal definitions. We define a text sequence as $\mathbf{x} = (x_1, x_2, \ldots, x_t) \in \mathcal{X}^t$, where each $x_i \in \mathcal{D}$ is a word, $\mathcal{D}$ is a set of words that shapes a predefined dictionary. A (next-step) language model is a generative model which models the probability $\Pr[\mathbf{x}]$ by applying the chain rule $\Pr[\mathbf{x}] = \Pi_1^t \Pr[x_i \mid x_1, x_2, \ldots, x_{i-1}]$. Specifically, a next-step language model f takes a prefix of the text sequence $x_1, \ldots, x_i$ as input, and with the parameter θ it returns the likelihood $f_\theta(x_i \mid x_1, \ldots x_{i-1})$ that ideally equals $\Pr[x_i \mid x_1, x_2, \ldots, x_{i-1}]$. As an example, an N-gram language model models the mentioned probabilities with a Markov chain, and it approximates $\Pr[x_i \mid x_1, \ldots, x_{i-1}]$ with the estimated probability of $N-1$ previous words, i.e., $\Pr[x_i \mid x_{i-N+1}, \ldots, x_{i-1}]$ (N contiguous words $x_{i-N+1}, \ldots, x_i$ is called an N-gram). Specifically a bigram language model ($N = 2$) follows $f_{\mathbf{bi}}[x_i \mid x_1, \ldots, x_{i-1}] \approx \Pr[x_i \mid x_{i-1}]$ and a trigram language model ($N = 3$) follows $f_{\mathbf{tri}}[x_i \mid x_1, \ldots, x_{i-1}] \approx \Pr[x_i \mid x_{i-2}, x_{i-1}]$. In the training of the language model, a training dataset $\mathcal{S}$ with multiple sequences is given. The language model parameter θ is optimized to maximize the overall likelihood of sequences in the training dataset, that is, the probability of returning the dataset given such N-gram probability.

Threat model. In general, we follow the security game described in Definition 4.1. Namely, we first train the language model with a dataset $\mathcal{S}$. In the deletion step, we delete a random sequence $\mathbf{x}' = x'_1, \ldots, x'_{i-1}$ from the dataset and retrain the model. Finally, the adversary aims to reconstruct the example $\mathbf{x}'$. Note that black-box access by the adversary means that it can send an text sequence $x_1, x_2, \ldots, x_t$ to the language model and gets the probability of the text sequence $f_\theta(x_1, x_2, \ldots, x_t)$.

Our deleted sentence reconstruction attack. We now define a simple adversary that can accurately reconstruct the deleted sentence. It first simply queries every possible N-grams in the dictionary to the model h and records their probabilities. Then after deletion, it again sends every possible N-grams queries to the model h_{del}. Now suppose $N = 2$, i.e. bigram. According to the definition of language models, for a word pair (x'_{i-1}, x'_i), if $h(x'_{i-1}, x'_i) > h_{\mathsf{del}}(x'_{i-1}, x'_i)$, then the number of occurrence of the bigram x'_{i-1}, x'_i is decreased in the updated dataset, which further indicates the bigram is included in the deleted example. Therefore, for one particular suffix x'_{i-1}, the adversary can guess a word x'_i which satisfies that $h(x'_{i-1}, x'_i) > h_{\mathsf{del}}(x'_{i-1}, x'_i)$.

We then propose a heuristic approach to reconstruct the deleted text sequence. First, we abstract the problem into a search problem defined on a graph, where each node is a N-gram that satisfies $h(x'_{i-N+1}, \ldots, x'_i) > h_{\mathsf{del}}(x'_{i-N+1}, \ldots, x'_i)$. We draw a directed edge from an N-gram node v_i to v_j if and only if the last $N-1$ words of v_i is the first $N-1$ words of v_j with the same order. Then each path in the graph represents a sentence. We then search to find a Hamiltonian path in the generated graph. Note that it is possible that the deleted sentence includes a specific N-gram with multiplicity more than one. We then allow the "Hamiltonian" path to tolerate a limited number of repetitive visits to a node. Finally, we return the shortest traverse path found, i.e., with the fewest number of repetitions. To implement this attack we use a recursive algorithm to traverse the nodes of the graph while we maintain the number of times that the current path has visited each node.

Experiment details. We perform our attacks on unigram, bigram, and trigram language models. We train the language models on the Penn Treebank Corpus [MSM93]. After regular preprocessing, the dataset includes 42068 text sequences, which includes 971657 words and 10001 unique words. We use two metrics to evaluate our attacks.

- Success rate: Probability that the adversary reconstructs a sequence x completely, when x is chosen at random from $\mathcal{S}$, it is deleted, and then the adversary is able to extract x by first interacting with h and then with h_{-x}.
- F1 score of the reconstruction: Let the reconstructed sequence of the adversary Adv be x' and the deleted sequence be x. Let's treat both of them as unordered multisets. Then the F1 score of the reconstruction measures the quality of the reconstruction by balancing the precision and recall of the prediction, namely,

$$F1 = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2|x \cap x'|}{(|x| + |x'|)}.$$

which is equal to 1 if and only if $x = x'$ (as multisets).

We then repeat the security game for 1000 times (i.e., each time a random sentence is deleted), and measure the two metrics on three language models.

Results. We present the experimental result on the three language models in Table 7. Note that the unigram language model does not store anything on order, so it is impossible to reconstruct the full sequence in the correct order. Our defined reconstruction attack gets 99% on the bigram and trigram models on the F1 score, and also successfully reconstruct 97% of the sequence with correct words and correct order on the trigram model.

	Success rate	F1 score
unigram	\	93.76%
bigram	62.00%	99.72%
trigram	97.30%	99.90%

Table 7. Reconstruction attack on language models.

Leakage of deletion. Note that without deletion, even if the adversary can fully reconstruct the N-gram model, the adversary only has the probability of N-grams, which is an aggregation over all the N-grams in the dataset. Although the adversary has those N-grams, it is still hard to get specific private information when the dataset is large. However, we show that when deletion happens, by tracing the *changes* in the probabilities during the deletion, an adversary can extract the *full deleted sequence* (of length longer than N) with high probability, completely revealing the deleted sequence.