Reconstructing Training Data with Informed Adversaries

Borja Balle
DeepMind

Giovanni Cherubin
The Alan Turing Institute

Jamie Hayes
DeepMind

Abstract

Given access to a machine learning model, can an adversary reconstruct the model's training data? This work proposes a formal threat model to study this question, shows that reconstruction attacks are feasible in theory and in practice, and presents preliminary results assessing how different factors of standard machine learning pipelines affect the success of reconstruction. Finally, we empirically evaluate what levels of differential privacy suffice to prevent reconstruction attacks.

1 Introduction

Membership leakage is considered the gold standard for privacy in machine learning (ML), both from the point of view of empirical privacy evaluation (e.g. via membership inference attacks (Shokri et al., 2017)) as well as mitigation (e.g. differential privacy (Dwork et al., 2006)). Reconstruction of training data, on the other hand, represents a high risk scenario for privacy in ML models even when membership is not sensitive information. In the spectrum of privacy attacks, while membership inference recovers a single bit of information about the training points, a reconstruction attack uncovers the maximum amount of information and is a worst-case scenario for privacy leakage.

Despite the seriousness posed by reconstruction, there is little work investigating when such attacks are possible[1] and how to mitigate them. In this work, we show how a powerful informed adversary can mount effective reconstruction attacks. We propose a threat model to investigate reconstruction, prove that a broad family of convex models are vulnerable in our threat model, and empirically illustrate that neural networks are also susceptible to these attacks. We conclude by showing that even weak levels of differential privacy (DP) provide a successful mitigation against our attacks.

2 Threat model / Adversarial capabilities

Our threat model involves two players: model developer and adversary. The model developer owns a dataset $D = \{z_1, \ldots, z_n\}$ with $z_i \in \mathcal{Z}$ and uses an algorithm $\mathcal{A}$ to train a model $\theta^* = \mathcal{A}(D)$ which is revealed to the adversary. The adversary knows the training dataset D except for one point $z \in D$, and their goal is to use $D_0 = D \setminus \{z\}$ and θ^* to recover the missing point. We assume the adversary has full knowledge of the training algorithm $\mathcal{A}$; when the algorithm is randomized, we consider both the situations where the internal randomness might or might not be known to the adversary. In addition, the adversary may have side-knowledge about the missing point z in the form of a prior distribution – in practice this knowledge may come from additional samples from the distribution where D was sampled from. In some applications we might want to declare the adversary succeeded even if z is only recovered approximately; in these cases we introduce a distance on $\mathcal{Z}$ and declare success if the adversary produces a point $\hat{z}$ "close enough" to z.

The adversary in this threat model is extremely powerful; for example, they could enumerate (a fine discretization of) $\mathcal{Z}$ and pick the candidate $\hat{z}$ that produces the model $\hat{\theta} = \mathcal{A}(D_0 \cup \{\hat{z}\})$ closest to θ^*. However, for high-dimensional data this enumerative approach is infeasible, so we focus on attacks that can be executed in practice. In terms of side-knowledge, knowing all the training data except for one point mimics the intrinsic threat model used in DP, which ensures protection even against an adversary who knows

[1] Carlini et al. (2019, 2020) are notable exceptions in the context of generative language models.

Preprint. Under review.

the whole data set except for one point. In particular, our threat model can be seen as an instantiation of the model used in semantic privacy (Dwork et al., 2016, Definition A-4).

3 Reconstruction attacks on generalized linear models

We first discuss a general reconstruction attack strategy against a broad family of convex models when the empirical risk minimization problem has a unique minimum and is solved to optimality. Specifically, we show there exists a closed form solution to do reconstruction for Generalized Linear Models (GLMs). This attack applies to models such as linear regression, ridge regression, and logistic regression.

Consider fitting a GLM derived from a canonical exponential family with non-negative dispersion parameter ψ; without any loss of generality for our derivation, we let $\psi = 1$. Assume the training data D consists of pairs $z_i = (x_i, y_i)$ with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. We train the model via (regularized) MLE, by minimizing the objective $L(\theta) = -\sum_{i=1}^{n} \left(b(\langle x_i, \theta \rangle) - \langle x_i, \theta \rangle y_i\right) + \lambda \|\theta\|^2$, where function b is s.t. $b' = g^{-1}$, g is a canonical link function, and $\lambda \geq 0$ is a regularization parameter. For example, g^{-1} is the identify function for linear regression and the sigmoid function for logistic regression. This optimization admits a unique minimum when either $\lambda > 0$, b is strictly concave (as in the examples above) or the data is in general position (Wedderburn, 1976). We get the optimum by equating the gradient to zero:

$$-\sum_{i=1}^{n} x_i (g^{-1}(\langle x_i, \theta^* \rangle) - y_i) + \lambda \theta^* = 0 \ . \tag{1}$$

Suppose for simplicity that the model is trained with an intercept parameter, i.e. the first coordinate of each feature vector is equal to 1. Then, given the optimum θ^* and all but one of the training points, the optimality condition (1) yields a system with d equations and d unknowns (the label plus the last $d-1$ features of the missing point). The following solution for this system gives an effective reconstruction attack.

Theorem 3.1 (Reconstruction attack against GLMs)**.** *Let θ^* be the unique optimum of $L(\theta)$ and D_0 the training data set except for one point (x, y). Suppose $\bar{X}$ contains as rows the features of all points in D_0 (its first column satisfies $\bar{X}_1 = \vec{1}$) and similarly for $\bar{Y}$ with the labels. Then we have:*

$$B_1 = \bar{X_1}^\top (g^{-1}(\bar{X}\theta^*) - \bar{Y}) \ , \quad x = \frac{\bar{X}^\top (g^{-1}(\bar{X}\theta^*) - \bar{Y}) + \lambda \theta^*}{B_1 + \lambda \theta_1^*} \ , \quad y = g^{-1}(\langle x, \theta^* \rangle) + \lambda B_1 \theta_1^* \ .$$

4 Reconstruction attacks on neural networks

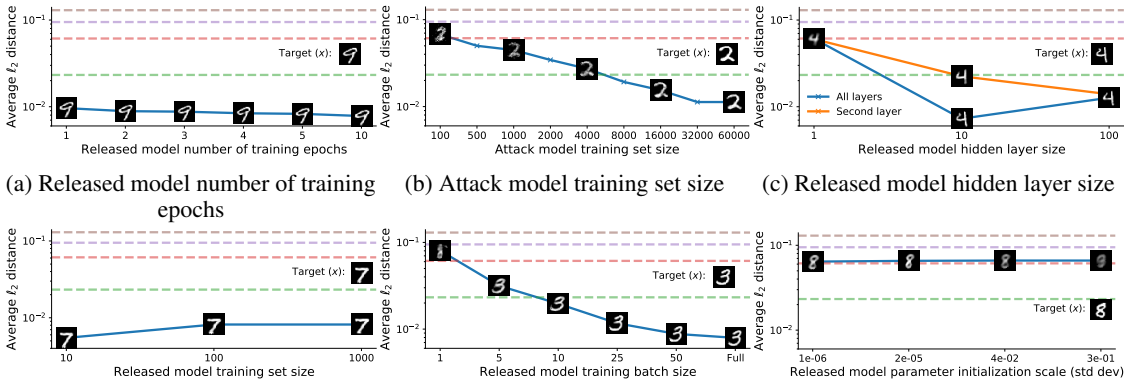
The closed form solution for reconstructing a point given above relies on the existence of a unique minimizer and the training procedure reaching this optimum. Unfortunately, this is no longer the case for non-convex models; there may exist more than one global optimum, and attaining any of these minima via training is not a foregone conclusion. For this setting, we design an adversary that *learns* a mapping from model parameters to missing training points, and use this to mount a reconstruction attack.

Attack workflow. The adversary is given θ^*, which we refer to as the *released model*, the training data with one point z omitted, D_0, and the learning algorithm with relevant hyper-parameters, $\mathcal{A}$. We also assume the adversary has access to k additional data points $\bar{D} = \{\bar{z}_1, \ldots, \bar{z}_k\}$ from the same distribution as D_0. The attack proceeds in three steps:

1. Obtain k models $\{\theta_1, \cdots, \theta_k\}$ where each θ_i is trained on data $\{\bar{z}_i\} \cup D_0$ with the training algorithm $\mathcal{A}$ used by the model developer.
2. Train an *attack model*, ϕ, that given model parameters θ_i learns to output the additional training point $\bar{z}_i$. The *attack dataset* is the set of k examples $\{(\theta_i, \bar{z}_i)\}_{i=1}^{k}$ of successful reconstructions.
3. Apply ϕ to the released model to obtain the candidate reconstruction $\hat{z} = \phi(\theta^*)$.

In all our experiments we consider standard classification tasks where $z = (x, y) \in \mathcal{X} \times \mathcal{Y}$ with a finite set of labels $\mathcal{Y}$. We thus make the simplifying assumption that if x can be reconstructed, then the label y can be accurately inferred from x. Accordingly, we sometimes abuse notation and write $\bar{D}$ to denote only the set of features for the additional data points.

Reconstruction metric. We measure the reconstruction quality as the distance between target and reconstruction, $\|x - \hat{x}\|_2$. As a baseline, we use the smallest distance between the target and any example from the attack training data $\min_{\bar{x} \in \bar{D}} \|x - \bar{x}\|_2$. For a good reconstruction, the distance to the target should be much smaller than the baseline; i.e., the reconstruction should be more similar to the target than to any other point that is available to the adversary to carry out the attack.



(a) Released model number of training epochs (b) Attack model training set size (c) Released model hidden layer size

(d) Released training set size (e) Released model training batch size (f) Released model random initialization

Figure 1: Average ℓ_2 distance over all test set targets and reconstructions. We also display non-cherry-picked examples along with the average ℓ_2 distance, and reference baseline distances introduced in Figure 4.

4.1 Empirical evaluation

We now empirically show that this attack is effective on a simple MNIST classification task using a small neural network as the released model, and survey factors that affect the fidelity of reconstruction.

Experimental setup. We split the MNIST dataset into three sets: a *fixed* dataset of size $\sim$1K from where we will draw D_0, a *training* dataset of size $\sim$60K from which we draw $\bar{D}$, and a *test* dataset of size $\sim$1K from which we draw the target points z. The released model is an MLP with one hidden layer, and the attack model is a feed-forward network with two hidden layers mapping (flattened) model parameters to 28x28 gray-scale images. Further details about the setup are in Appendix A.

We first investigate the influence of each hyper-parameter on reconstruction success. We evaluate both deterministic (i.e. gradient descent from fixed starting point) and randomized training algorithms (i.e. random initialization and/or random mini-batches) for the released model. Results are in Figure 1, where the x-axis corresponds to a hyper-parameter and the y-axis represents average reconstruction error (ℓ_2 norm) across the test set. Additionally, for each task we present candidate reconstructions for a single representative target, and compare the average reconstruction error against error percentiles between the target and other images available to the adversary (see Appendix A.1 for details).

4.1.1 Released model with deterministic training

Does reconstruction improve with more released model training epochs? Figure 1a shows that although reconstructions do improve if the released model is trained for longer, they are already highly similar to their targets after the first epoch of training the released model. In other words, it is possible to reconstruct the target input even if it was processed only once by the released model during training.

Size of attack training set. Figure 1b shows that increasing the size of the attack dataset available to the adversary to train the attack model significantly improves the quality of reconstruction.

Size of target model hidden layer. Figure 1c shows how the size of the released model affects reconstructions. Intuitively, a model with a larger hidden layer can "memorize" more of its training data, making reconstruction easier. We observe this when going from a hidden layer size of 1 to 10. However, the attack model's input is the flattened vector of the released model's parameters; consequently, when moving from a hidden layer of size 10 to 100, the input dimensionality to the attack model increases tenfold, which hinders the ability for the attack model to learn on this data. We ameliorate this curse of dimensionality by supplying only the second layer of the released model parameters as input to the attack model; this improves the reconstruction quality as the released model's hidden layer size increases. Finally, it is worth noting that, although reconstruction fidelity is poor for a released model with hidden layer size 1, it still enables one to infer the target's label.

Size of released training dataset. One may conjecture that increasing the size of the fixed dataset will results in worse reconstruction, because the released model parameters will be less dependent on the single input unknown to the adversary. In Figure 1d, we show results for different sizes of released model training set, made up of the fixed dataset and a single sample from either the training or test set. Although the

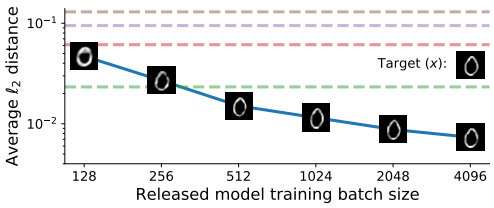


Figure 2: "Realistic" released model training size (10K+1) and batch size.

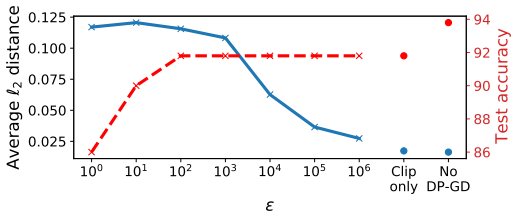


Figure 3: Average ℓ_2 distance of reconstructions and test accuracy of released model using (ε, δ)-DP.

reconstruction fidelity becomes poorer with larger training sets, the change is relatively small in comparison to our baselines, and reconstructions still resemble the targets.

4.1.2 Released model with randomized training

Batch size. In Figure 1e, we measure reconstruction quality when there is some stochasticity in training with random mini-batches. Clearly, it is not possible to reconstruct with a batch size of 1, however at a batch size of only 5 it is possible to reconstruct the target with almost no perceptible difference. As the batch size increases, the amount of randomness in training decreases, our attack exploits this resulting in an improvement in reconstruction quality.

Random weight initialization. So far we have assumed the adversary knows the initialization of the released model. This knowledge may come, e.g., from a leaked random seed, or in a fine-tuning scenario, where the model developer starts training from a publicly available model. We now consider an adversary that no longer knows the initial weights. We initialize the weights using a truncated normal distribution with zero mean and $1/\sqrt{\text{Input size to layer}}$ standard deviation, the default in the Haiku library (Hennigan et al., 2020). In Figure 1f, we vary the standard deviation of the initialization. For small values, the initial parameters of all the models are similar, resulting in reasonably good reconstructions, while large values completely distort reconstructions. We conclude that the randomization of the initial model parameters is a serious challenge to reconstruction success for the proposed attack.

4.1.3 Towards realistic released models

Our previous set of experiments reviewed factors in isolation. We combine all of these into a single experiment, demonstrating the practicality of our attack. We increase the fixed set size used for training released models to 10,000, no longer use full batch gradient descent, and train released models for 100 epochs; all released models achieve 93-95% accuracy on the MNIST test set. Even in this more practical scenario, our attack generates highly effective reconstructions (Figure 2). The interested reader can find non-cherry-picked examples of reconstructions in Appendix A.2. It is also important to note that our findings are not confined to fully connected architectures; we have experimented with small convolutional neural networks and achieve equivalent results.

4.2 Mitigating reconstruction with differentially private gradient descent (DP-GD)

In Figure 3 we show how even a large ε in (ε, δ)-DP can result in a successful mitigation against reconstruction attacks. The released model training set-up is identical to Section 4.1.3, except we train with full batch differentially private gradient descent with clipped gradients (Abadi et al., 2016). Gradients are clipped to have a maximum ℓ_2 norm of 1, and Gaussian noise is added such that the model is $(\varepsilon, \delta = 10^{-5})$-DP. Interestingly, for high levels of privacy, the reconstruction attack generates realistic but wildly incorrect reconstructions. The interested reader can find non-cherry-picked examples of reconstructions in Appendix A.2.

5 Conclusion

Our work is a first step towards a systematic study on the risks of training data reconstruction from ML models and potential mitigations. In future work we will: study how to improve reconstruction attacks when model initialization is not known, how to improve our attacks' data efficiency, and how to bound the success of reconstruction attacks in terms of DP parameters.

References

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

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. (2019). The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 267–284.

Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T. B., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. (2020). Extracting training data from large language models. *CoRR*, abs/2012.07805.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. D. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, pages 265–284.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. D. (2016). Calibrating noise to sensitivity in private data analysis. *J. Priv. Confidentiality*, 7(3):17–51.

Hennigan, T., Cai, T., Norman, T., and Babuschkin, I. (2020). Haiku: Sonnet for JAX.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 3–18.

Wedderburn, R. W. (1976). On the existence and uniqueness of the maximum likelihood estimates for certain generalized linear models. *Biometrika*, 63(1):27–32.

A Experimental details

The MNIST dataset is split into three groups which we refer to as the *fixed*, *training*, and *test* sets. The fixed dataset is small ($\sim$10-1000) and represents the fixed part of the training set the adversary is assumed to know, D_0. The training dataset is larger ($\sim$60K); each point will represent the target input the adversary aims to reconstruct, and can train with; these points will make up the targets in the *attack training set*, $\bar{D} = \bigcup_{i=1}^{k=60,000} \bar{x}_i$. This means the adversary will generate $\sim$60K models, each of which will be trained on the fixed dataset, D_0, and one input, $\bar{x}_i$, sampled from $\bar{D}$. These models will make up the input features in the *attack training set*, $\bigcup_{i=1}^{k=60,000}(\theta_i, \bar{x}_i)$, and the *attack model*, ϕ, is trained on this data. The test dataset is smaller ($\sim$1K); released models will again be trained on the fixed dataset, D_0, and one sampled example, $\bar{x}_j$, from the test set. The *attack test dataset* we use to measure reconstruction success is then given by, $\bigcup_{j=1}^{k=1,000}(\theta_j, \bar{x}_j)$. We report the average distance between reconstructions, $\hat{x}_j$, and targets $\bar{x}_j$, for each example in the test set: $\frac{1}{1,000}\sum_{j=1}^{k=1,000} \|\bar{x}_j - \hat{x}_j\|$. This will be used to measure how well the *attack model* can learn to reconstruct unknown target inputs from *released models*.

We survey how training data reconstruction is affected by various hyperparameters of the released model and experimental set-up. Unless stated otherwise, the hyperparameter settings are as follows:

Released model hyperparameters. We use an MLP with one hidden layer. The fixed dataset has size[2] 100, the batch size is the same size as the released model training set size (100 + 1). The number of training epochs and the size of the hidden layer are both set to 10. The hidden layer's activation function is set to ReLU. Each released model is initialized with the same random seed. The learning rate is set to 0.01 and we use standard (S)GD as our optimizer[3]. The loss is set to cross-entropy used to solve the original MNIST prediction task.

Attack model hyperparameters. The attack model is trained with a learning rate of 0.001 using RMSProp. The input to the attack model is a flattened vector of the model parameters of an instance of a trained released

[2]Our experiments show that reconstruction is still feasible when this number gets larger by at least two orders of magnitude. We choose 100 for most of our experiments to reduce the computational cost of generating training data for the attack model.

[3]We also investigated the use of optimizers with momentum for the released model and obtained similar results as the ones presented here.

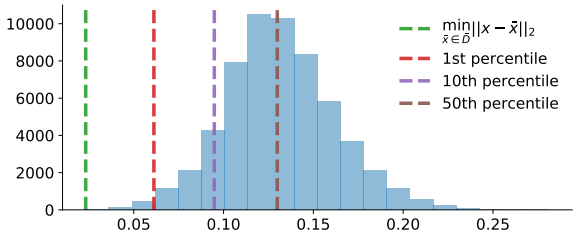


Figure 4: Averaged histogram of ℓ_2 distances between targets from the test set ($\sim$1K) and all examples from the training set ($\sim$60K).

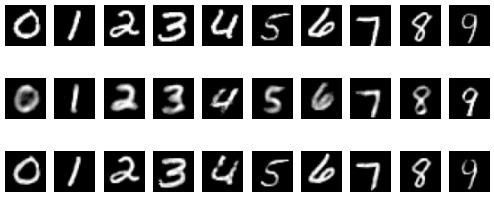


Figure 5: Example of reconstructions. Top: Targets, Middle: Reconstruction for released batch size of 128, Bottom: Reconstruction for released batch size of 4096.

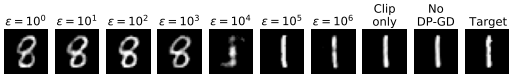


Figure 6: Example of reconstructions for DP-GD experiment in Section 4.2.

model. The attack model is trained for 30 epochs and has two hidden layer's of size 1000 with ReLU activations. The batch size is set to 128. The loss is a linear combination of ℓ_1 and ℓ_2 distance errors.

A.1 Reconstruction quality baselines

In Figure 4, we show the average histogram of ℓ_2 distances between targets that we aim to reconstruct from the test set, and all examples from the training set. Along with this, we insert markers for the average minimum distance, 1st, 10th, and 50th percentile. These markers are represented against reconstruction accuracy in Figure 1 as a benchmark for the fidelity of our reconstructions against knowledge readily available to the attacker.

A.2 Examples of reconstructed MNIST digits

In Figure 5 and Figure 6, we display non-cherry-picked examples of reconstructions under the experimental settings described in Section 4.1.3 and Section 4.2, respectively. In Figure 5, although reconstructions are of better quality for a batch size of 4096, reconstructions at a batch size of 128 still closely match their corresponding targets. In Figure 6, we see that reconstructions look realistic but are entirely incorrect up until $\epsilon > 10^3$.