THE LAST ITERATE ADVANTAGE: EMPIRICAL AUDITING AND PRINCIPLED HEURISTIC ANALYSIS OF DIFFERENTIALLY PRIVATE SGD

Thomas Steinke,* **Milad Nasr,*** **Arun Ganesh, Borja Balle, Christopher A. Choquette-Choo,**

Matthew Jagielski, Jamie Hayes, Abhradeep Guha Thakurta, Adam Smith, Andreas Terzis

ABSTRACT

We propose a simple heuristic privacy analysis of noisy clipped stochastic gradient descent (DP-SGD) in the setting where only the last iterate is released and the intermediate iterates remain hidden. Namely, our heuristic assumes a linear structure for the model.

We show experimentally that our heuristic is predictive of the outcome of privacy auditing applied to various training procedures. Thus it can be used prior to training as a rough estimate of the final privacy leakage. We also probe the limitations of our heuristic by providing some artificial counterexamples where it underestimates the privacy leakage.

The standard composition-based privacy analysis of DP-SGD effectively assumes that the adversary has access to all intermediate iterates, which is often unrealistic. However, this analysis remains the state of the art in practice. While our heuristic does not replace a rigorous privacy analysis, it illustrates the large gap between the best theoretical upper bounds and the privacy auditing lower bounds and sets a target for further work to improve the theoretical privacy analyses. We also empirically support our heuristic and show existing privacy auditing attacks are bounded by our heuristic analysis in both vision and language tasks.

1 INTRODUCTION

Differential privacy (DP) (Dwork et al., 2006) defines a measure of how much private information from the training data leaks through the output of an algorithm. The standard differentially private algorithm for deep learning is DP-SGD (Bassily et al., 2014; Abadi et al., 2016), which differs from ordinary stochastic gradient descent in two ways: the gradient of each example is clipped to bound its norm and then Gaussian noise is added at each iteration.

The standard privacy analysis of DP-SGD is based on composition (Bassily et al., 2014; Abadi et al., 2016; Mironov, 2017; Steinke, 2022; Koskela et al., 2020). In particular, it applies to the setting where the privacy adversary has access to all intermediate iterates of the training procedure. In this setting, the analysis is known to be tight (Nasr et al., 2021; 2023). However, in practice, potential adversaries rarely have access to the intermediate iterates of the training procedure, rather they only have access to the final model. Access to the final model can either be through queries to an API or via the raw model weights. The key question motivating our work is the following.

> Is it possible to obtain sharper privacy guarantees for DP-SGD when the adversary only has access to the final model, rather than all intermediate iterates?

1.1 BACKGROUND & RELATED WORK

The question above has been studied from two angles: Theoretical upper bounds, and privacy auditing lower bounds. Our goal is to shed light on this question from a third angle via principled heuristics.

*denotes equal contribution. Google DeepMind. {steinke,srxzr,arunganesh,bballe, cchoquette,jagielski,jamhay,athakurta,adamdsmith,aterzis}@google.com.
https://arxiv.org/abs/2410.06186

A handful of theoretical analyses (Feldman et al., 2018; Chourasia et al., 2021; Ye & Shokri, 2022; Altschuler & Talwar, 2022; Bok et al., 2024) have shown that asymptotically the privacy guarantee of the last iterate of DP-SGD can be far better than the standard composition-based analysis that applies to releasing all iterates. In particular, as the number of iterations increases, these analyses give a privacy guarantee that converges to a constant (depending on the loss function and the scale of the noise), whereas the standard composition-based analysis would give a privacy guarantee that increases forever. Unfortunately, these theoretical analyses are only applicable under strong assumptions on the loss function, such as (strong) convexity and smoothness. We lack an understanding of how well they reflect the "real" privacy leakage.

Privacy auditing (Jagielski et al., 2020; Ding et al., 2018; Bichsel et al., 2018; Nasr et al., 2021; 2023; Steinke et al., 2023; Tramer et al., 2022; Zanella-Béguelin et al., 2022) complements theoretical analysis by giving empirical lower bounds on the privacy leakage. Privacy auditing works by performing a membership inference attack (Shokri et al., 2017; Homer et al., 2008; Sankararaman et al., 2009; Dwork et al., 2015). That is, it constructs neighbouring inputs and demonstrates that the corresponding output distributions can be distinguished well enough to imply a lower bound on the differential privacy parameters. In practice, the theoretical privacy analysis may give uncomfortably large values for the privacy leakage (e.g., $\varepsilon > 10$); in this case, privacy auditing may be used as evidence that the "real" privacy leakage is lower. There are settings where the theoretical analysis is matched by auditing, such as when all intermediate results are released (Nasr et al., 2021; 2023). However, despite significant work on privacy auditing and membership inference (Carlini et al., 2022; Bertran et al., 2024; Wen et al., 2023; Leino & Fredrikson, 2020; Sablayrolles et al., 2019; Zarifzadeh et al., 2023), a large gap remains between the theoretical upper bounds and the auditing lower bounds (Andrew et al., 2023; Nasr et al., 2023; Cebere et al., 2024; Annamalai & De Cristofaro, 2024) when only the final parameters are released. This observed gap is the starting point for our work.

1.2 Our Contributions

We propose a *heuristic* privacy analysis of DP-SGD in the setting where only the final iterate is released. The heuristic ε is always lower than the standard theoretical analysis. Our experiments demonstrate that this heuristic analysis consistently provides an upper bound on the privacy leakage measured by privacy auditing tools in realistic deep learning settings.

Our heuristic analysis corresponds to a worst-case theoretical analysis under the assumption that the loss functions are linear. This case is simple enough to allow for an exact privacy analysis whose parameters can be computed numerically (Theorem 1). Our consideration of linear losses is built on the observation that current auditing techniques achieve the highest ε values when the gradients of the canaries – that is, the examples that are included or excluded to test the privacy leakage – are constant across iterations and independent from the gradients of the other examples. This is definitely the case for linear losses; the linear assumption thus allows us to capture the setting where current attacks are most effective. Linear loss functions are also known to be the worst case for the non-subsampled (i.e., full batch) case; see Appendix B. Assuming linearity is unnatural from an optimization perspective, as there is no minimizer. But, from a privacy perspective, we show that it captures the state of the art.

We also probe the limitations of our heuristic and give some artificial counterexamples where it underestimates empirical privacy leakage. One class of counterexamples exploits the presence of a regularizer. Roughly, the regularizer partially zeros out the noise that is added for privacy. However, the regularizer also partially zeros out the signal of the canary gradient. These two effects are almost balanced, which makes the counterexample very delicate. In a second class of counterexamples, the data is carefully engineered so that the final iterate effectively encodes the entire trajectory, in which case there is no difference between releasing the last iterate and all iterates.

Implications: Heuristics cannot replace rigorous theoretical analyses. However, our heuristic can serve as a target for future improvements to both privacy auditing as well as theoretical analysis. For privacy auditing, matching or exceeding our heuristic is a more reachable goal than matching the theoretical upper bounds, although our experimental results show that even this would require new attacks. When theoretical analyses fail to match our heuristic, we should identify why there is a gap, which builds intuition and could point towards further improvements.

Given that privacy auditing is computationally intensive and difficult to perform correctly (Aerni et al., 2024), we believe that our heuristic can also be valuable in practice. In particular, our heuristic

can be used prior to training (e.g., during hyperparameter selection) to predict the outcome of privacy auditing when applied to the final model. (This is a similar use case to scaling laws.)

2 LINEARIZED HEURISTIC PRIVACY ANALYSIS

Theorem 1 presents our heuristic differential privacy analysis of DP-SGD (which we present in Algorithm 1 for completeness; note that we include a regularizer r whose gradient is *not* clipped, because it does not depend on the private data $\mathbf{x}$).[1] We consider Poisson subsampled minibatches and add/remove neighbours, as is standard in the differential privacy literature.

Our analysis takes the form of a conditional privacy guarantee. Namely, under the assumption that the loss and regularizer are linear, we obtain a fully rigorous differential privacy guarantee. The heuristic is to apply this guarantee to loss functions that are not linear (such as those that arise in deep learning applications). Our thesis is that, in most cases, the conclusion of the theorem is still a good approximation, even when the assumption does not hold.

Recall that a function $\ell : \mathbb{R}^d \to \mathbb{R}$ is linear if there exist $\alpha \in \mathbb{R}^d$ and $\beta \in \mathbb{R}$ such that $\ell(\mathbf{m}) = \langle \alpha, \mathbf{m} \rangle + \beta$ for all $\mathbf{m}$.

Theorem 1 (Privacy of DP-SGD for linear losses)**.** *Let $\mathbf{x}, T, q, \eta, \sigma, \ell, r$ be as in Algorithm 1. Assume r and $\ell(\cdot, x)$, for every $x \in \mathcal{X}$, are linear.*

Algorithm 1 Noisy Clipped Stochastic Gradient Descent (DP-SGD) (Bassily et al., 2014; Abadi et al., 2016)

function DP-SGD($\mathbf{x} \in \mathcal{X}^n$, $T \in \mathbb{N}$, $q \in [0,1]$, $\eta \in (0,\infty)$, $\sigma \in (0,\infty)$, $\ell : \mathbb{R}^d \times \mathcal{X} \to \mathbb{R}$, $r : \mathbb{R}^d \to \mathbb{R}$)
  Initialize model $\mathbf{m}_0 \in \mathbb{R}^d$.
  for $t = 1 \cdots T$ **do**
    Sample minibatch $B_t \subseteq [n]$ including each element independently with probability q.
    Compute gradients of the loss $\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i)$ for all $i \in B_t$ and of the regularizer $\nabla_{\mathbf{m}_{t-1}} r(\mathbf{m}_{t-1})$.
    Clip loss gradients: $\mathsf{clip}\left(\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i)\right) := \frac{\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i)}{\max\{1, \|\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i)\|_2\}}$.
    Sample noise $\xi_t \leftarrow \mathcal{N}(0, \sigma^2 I_d)$.
    Update

$$\mathbf{m}_t = \mathbf{m}_{t-1} - \eta \cdot \left(\begin{array}{c} \sum_{i \in B_t} \mathsf{clip}\left(\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i)\right) \\ + \nabla_{\mathbf{m}_{t-1}} r(\mathbf{m}_{t-1}) + \xi_t \end{array} \right).$$

  end for
  if `last_iterate_only` **then**
    return $\mathbf{m}_T$
  else if `intermediate_iterates` **then**
    return $\mathbf{m}_0, \mathbf{m}_1, \cdots, \mathbf{m}_{T-1}, \mathbf{m}_T$
  end if
end function

Letting

$$P := \mathsf{Binomial}(T, q) + \mathcal{N}(0, \sigma^2 T) \quad \text{and} \quad Q := \mathcal{N}(0, \sigma^2 T), \tag{1}$$

DP-SGD with `last_iterate_only` *satisfies (ε, δ)-differential privacy with $\varepsilon \geq 0$ arbitrary and*

$$\delta = \delta_{T,q,\sigma}(\varepsilon) := \max\{H_{e^\varepsilon}(P, Q), H_{e^\varepsilon}(Q, P)\}. \tag{2}$$

Here, H_{e^ε} denotes the e^ε-hockey-stick-divergence $H_{e^\varepsilon}(P, Q) := \sup_S P(S) - e^\varepsilon Q(S)$.

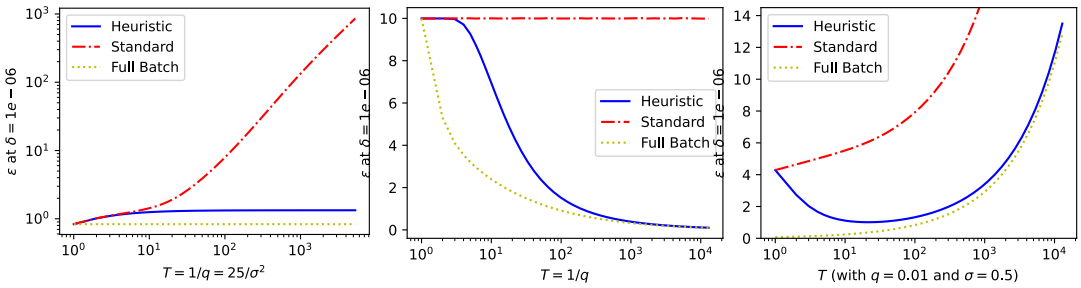
Equation 2 gives us a value of the privacy failure probability parameter δ. But it is more natural to work with the privacy loss bound parameter ε, which can be computed by inverting the formula:

$$\varepsilon_{T,q,\sigma}(\delta) := \min\{\varepsilon \geq 0 : \delta_{T,q,\sigma}(\varepsilon) \leq \delta\}. \tag{3}$$

Both $\delta_{T,q,\sigma}(\varepsilon)$ and $\varepsilon_{T,q,\sigma}(\delta)$ can be computed using existing open-source DP accounting libraries (Google, 2020). We also provide a self-contained & efficient method for computing them in Appendix A. The proof of Theorem 1 is deferred to Appendix A, but we sketch the main ideas: Under the linearity assumption, the output of DP-SGD is just a sum of the gradients and noises. We can reduce to dimension $d = 1$, since the only relevant direction is that of the gradient of the canary[2] (which is constant). We can also ignore the gradients of the other examples. Thus, by rescaling, the worst case pair of output distributions can be represented as in Equation 1. Namely, $Q = \sum_{t=1}^T \xi_t$ is simply the noise $\xi_t \leftarrow \mathcal{N}(0, \sigma^2)$ summed over T iterations; this corresponds to the case where the canary is

[1] Note that the regularizer is optional as it can be set to zero. We include it for convenience later.

[2] *The canary* refers to the individual datapoint that is added or removed between neighboring datasets. This terminology is used in the privacy auditing/attacks literature inspired on the expression "canary in a coalmine."



(a) Keep Tp and $T\sigma^2$ constant. (b) Keep Standard DP constant. (c) Keep p and σ constant.

Figure 1: Comparison of our heuristic to baselines in various parameter regimes. Horizontal axis is the number of iterations T and the vertical axis is ε such that we have $(\varepsilon, 10^{-6})$-DP.

excluded. When the canary is included, it is sampled with probability q in each iteration and thus the total number of times it is sampled over T iterations is $\mathsf{Binomial}(T, q)$. Thus P is the sum of the contributions of the canary and the noise. Finally the definition of differential privacy lets us compute ε and δ from this pair of distributions. Tightness follows from the fact that there exists a loss function and pair of inputs such that the corresponding outputs of DP-SGD matches the pair P and Q.

2.1 Baselines

In addition to privacy auditing, we compare our heuristic to two different baselines in Figure 1. The first is the standard, composition-based analysis. We use the open-source library from Google (Google, 2020), which computes a tight DP guarantee for DP-SGD with `intermediate_iterates`. Because DP-SGD with `intermediate_iterates` gives the adversary more information than with `last_iterate_only`, this will always give at least as large an estimate for ε as our heuristic.

We also consider approximating DP-SGD by full batch DP-GD. That is, set $q = 1$ and rescale the learning rate η and noise multiplier σ to keep the expected step and privacy noise variance constant:

$$\underbrace{\text{DP-SGD}(\mathbf{x}, T, q, \eta, \sigma, \ell, r)}_{\text{batch size} \approx nq,\ T \text{ iterations},\ Tq \text{ epochs}} \approx \underbrace{\text{DP-SGD}(\mathbf{x}, T, 1, \eta \cdot q, \sigma/q, \ell, r)}_{\text{batch size } n,\ T \text{ iterations},\ T \text{ epochs}}. \tag{4}$$

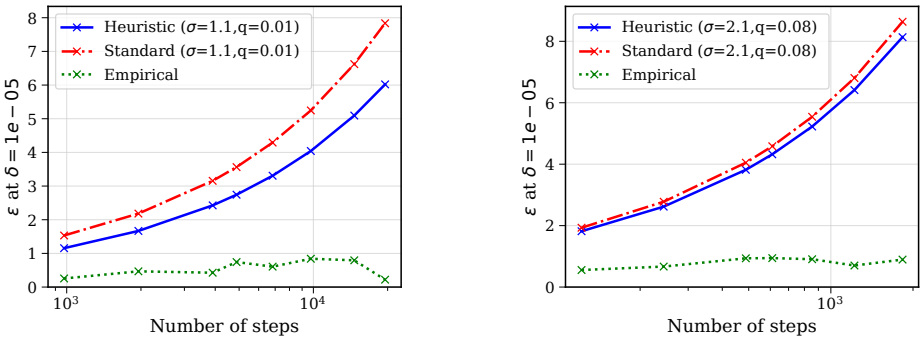
The latter algorithm is full batch DP-GD since at each step it includes each data point in the batch with probability 1. Since full batch DP-GD does not rely on privacy amplification by subsampling, it is much easier to analyze its privacy guarantees. Interestingly, there is no difference between full batch DP-GD with `last_iterate_only` and with `intermediate_iterates`; see Appendix B. Full batch DP-GD generally has better privacy guarantees than the corresponding minibatch DP-SGD and so this baseline usually (but not always) gives smaller values for the privacy leakage ε than our heuristic. In practice, full batch DP-GD is too computationally expensive to run. But we can use it as an idealized comparison point for the privacy analysis.

3 Empirical Evaluation via Privacy Auditing

We study how our heuristic compares to the empirical privacy auditing results in well studied settings. We focus on image classification (CIFAR-10), as this is a well-studied setting for privacy auditing of DP-SGD (Jagielski et al., 2020; Nasr et al., 2021; 2023; Tramer et al., 2022; Steinke et al., 2023; Cebere et al., 2024). In Section 3.2 we extend the experiments to language models and show similar results.

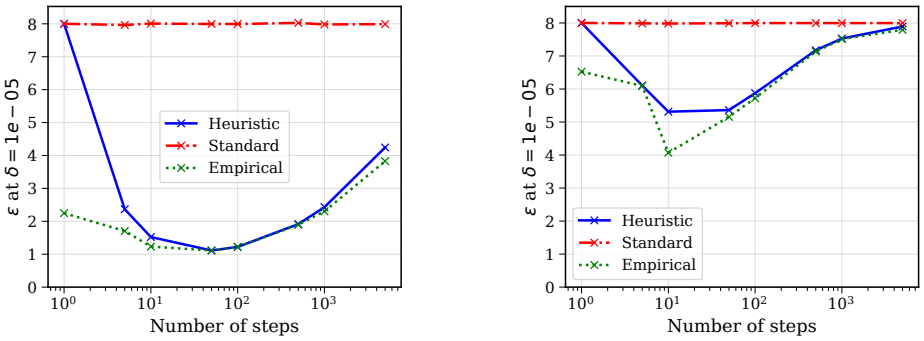
3.1 Image Classification Experiments

Setup: We follow the construction of Nasr et al. (2021) where we have 3 entities, adversarial crafter, model trainer, and distinguisher. In this paper, we assume the distinguisher only has access to the final iteration of the model parameters. We use the CIFAR10 dataset (Alex, 2009) with a WideResNet model (Zagoruyko & Komodakis, 2016) unless otherwise specified; in particular, we follow the training setup of De et al. (2022), where we train and audit a model with 79% test accuracy and,



(a) $q = 0.01$ (b) $q = 0.08$

Figure 2: Black-box gradient space attacks fail to achieve tight auditing when other data points are sampled from the data distribution. Heuristic and standard bounds diverge from empirical results, indicating the attack's ineffectiveness. This contrasts with previous work which tightly auditing with access to intermediate updates.



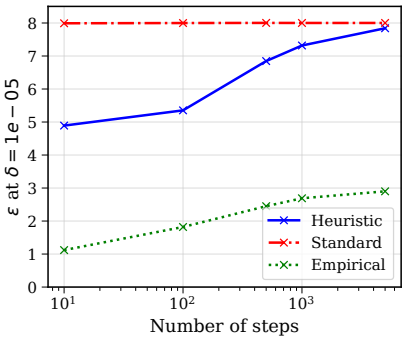
(a) $q = 0.01$ (b) $q = 0.1$

Figure 3: For gradient space attacks with adversarial datasets, the empirical epsilon (ε) closely tracks the final epsilon except for at small step counts, where distinguishing is more challenging. This is evident at both subsampling probability values we study ($q = 0.01$ and $q = 0.1$).

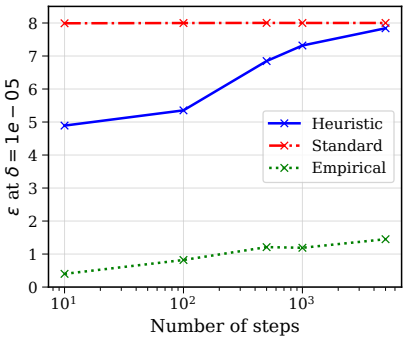
using the standard analysis, $(\varepsilon = 8, \delta = 10^{-5})$-DP. For each experiment we trained 512 CIFAR10 models with and without the canary (1024 total). To compute the empirical lower bounds we use the PLD approach with Clopper-Pearson confidence intervals used by Nasr et al. (2023). Here we assume the adversary knows the sampling rate and the number of iterations and is only estimating the noise multiplier used in DP-SGD, from which the reported privacy parameters (ε and δ) are derived.

We implement state-of-the-art attacks from prior work (Nasr et al., 2021; 2023). These attacks heavily rely on the intermediate steps and, as a result, do not achieve tight results. In the next section, we design specific attacks for our heuristic privacy analysis approach to further understand its limitations and potential vulnerabilities.

Gradient Space Attack: The most powerful attacks in prior work are gradient space attacks where the adversary injects a malicious gradient directly into the training process, rather than an example; prior work has shown that this attack can produce tight lower bounds, independent of the dataset and model used for training (Nasr et al., 2023). However, these previous attacks require access to all intermediate training steps to achieve tight results. Here, we use canary gradients in two settings: one where the other data points are non-adversarial and sampled from the real training data, and another where the other data points are designed to have very small gradients (≈ 0). This last setting was shown by (Nasr et al., 2021) to result in tighter auditing. In all attacks, we assume the distinguisher has access to all adversarial gradient vectors. For malicious gradients, we use Dirac gradient canaries, where gradient vectors consist of zeros in all but a single index. In both cases, the distinguishing test measures the dot product of the final model checkpoint and the gradient canary.



(a) All zero gradient inputs



(b) CIFAR10 Dataset

Figure 4: Input space attacks show promising results with both natural and blank image settings, although blank images have higher attack success. These input space attacks achieve tighter results than gradient space attacks in the natural data setting, in contrast to findings from prior work.

Figure 2 summarizes the results for the non-adversarial data setting, with other examples sampled from the true training data. In this experiment, we fix noise magnitude and subsampling probability, and run for various numbers of training steps. While prior work has shown tight auditing in this setting, we find an adversary without access to intermediate updates obtains much weaker attacks. Indeed, auditing with this strong attack results even in much lower values than the heuristic outputs.

Our other setting assumes the other data points are maliciously chosen. We construct an adversarial "dataset" of $m + 1$ gradients, m of which are zero, and one gradient is constant (with norm equal to the clipping norm), applying gradients directly rather than using any examples. As this experiment does not require computing gradients, it is very cheap to run more trials, so we run this procedure $N = 100,000$ times with the gradient canary, and N times without it, and compute an empirical estimate for ε with these values. We plot the results of this experiment in Figure 3 together with the ε output by the theoretical analysis and the heuristic, fixing the subsampling probability and varying the number of update steps. We adjust the noise parameter to ensure the standard theoretical analysis produces a fixed ε bound. The empirical measured ε is close to the heuristic ε except for when training with very small step counts: we expect this looseness to be the result of statistical effects, as lower step counts have higher relative variance at a fixed number of trials.

Input Space Attack: In practice, adversaries typically cannot insert malicious gradients freely in training steps. Therefore, we also study cases where the adversary is limited to inserting malicious inputs into the training set. Label flip attacks are one of the most successful approaches used to audit DP machine learning models in prior work (Nasr et al., 2023; Steinke et al., 2023). For input space attacks, we use the loss of the malicious input as a distinguisher. Similar to our gradient space attacks, we consider two settings for input space attacks: one where other data points are correctly sampled from the dataset, and another where the other data points are blank images.

Figure 4 summarizes the results for this setting. Compared to Figure 2, input space attacks achieve tighter results than gradient space attacks. This finding is in stark contrast to prior work. The reason is that input space attacks do not rely on intermediate iterates, so they transfer well to our setting.

In all the cases discussed so far, the empirical results for both gradient and input attacks fall below the heuristic analysis and do not violate the upper bounds based on the underlying assumptions. This suggests that the heuristic might serve as a good indicator for assessing potential vulnerabilities. However, in the next section, we delve into specific attack scenarios that exploit the assumptions used in the heuristic analysis to create edge cases where the heuristic bounds are indeed violated.

3.2 Language Model Experiments

While our main results focus on image classification models, our findings extend to other modalities. We also explore fine-tuning language models with differential privacy, specifically Gemma 2 (Team et al., 2024) on the PersonaChat dataset (Zhang et al., 2018). Given several works on auditing differential private image classification works, previous research on images investigated designing optimal canaries to improve empirical auditing in image classification tasks. However, it is unclear how to design such examples for modern language models.

Table 1: Evaluation of empirical auditing for DP finetuning of Gemma 2 on PersonaChat varying batch size

Batch size	Heuristic ε	Standard ε	Perplexity	Empirical ε
2048	6.0	8	24.3	2.1
1024	6.1	14	24.2	2.1
512	6.0	26	24.8	2.0

Interestingly, we observed that Gemma 2 tokenizer contains many tokens that are not present in the PersonaChat dataset. In this work, we utilize these tokens as differing examples, leaving the design of optimal canaries for language models to future research. Table 1 summarizes the results for the language model setting (cf. Table 2 for the image classification setting). We can see that there is not much difference between different modality and our heuristic approach can be a good predictive of the empirical privacy bounds. Additional details about the experiments can be found at Appendix D.

4 Counterexamples

We now test the limits of our heuristic by constructing some artificial counterexamples. That is, we construct inputs to DP-SGD with `last_iterate_only` such that the true privacy loss exceeds the bound given by our heuristic. While we do not expect the contrived structures of these examples to manifest in realistic learning settings, they highlight the difficulties of formalizing settings where the heuristic gives a provable upper bound on the privacy loss.

4.1 Warmup: Zeroing Out The Model Weights

We begin by noting the counterintuitive fact that our heuristic $\varepsilon_{T,q,\sigma}(\delta)$ is *not* always monotone in the number of steps T when the other parameters σ, q, δ are kept constant. This is shown in Figure 1c. More steps means there is both more noise and more signal from the gradients; these effects partially cancel out, but the net effect can be non-monotone.

We can use a regularizer $r(\mathbf{m}) = \|\mathbf{m}\|_2^2/2\eta$ so that $\eta \cdot \nabla_{\mathbf{m}} r(\mathbf{m}) = \mathbf{m}$. This regularizer zeros out the model from the previous step, i.e., the update of DP-SGD (Algorithm 1) becomes

$$\mathbf{m}_t = \mathbf{m}_{t-1} - \eta \cdot \left(\sum_{i \in B_t} \mathsf{clip}\left(\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i)\right) + \nabla_{\mathbf{m}_{t-1}} r(\mathbf{m}_{t-1}) + \xi_t \right) \tag{5}$$

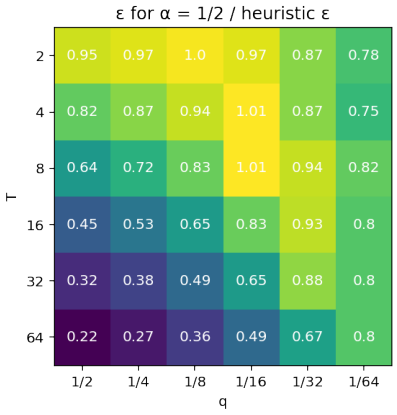
$$= \eta \cdot \sum_{i \in B_t} \mathsf{clip}\left(\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i)\right) + \xi_t. \tag{6}$$

This means that the last iterate $\mathbf{m}_T$ is effectively the result of only a single iteration of DP-SGD. In particular, it will have a privacy guarantee corresponding to one iteration. Combining this regularizer with a linear loss and a setting of the parameters T, q, σ, δ such that the privacy loss is non-monotone – i.e., $\varepsilon_{T,q,\sigma}(\delta) < \varepsilon_{1,q,\sigma}(\delta)$ – yields a counterexample.

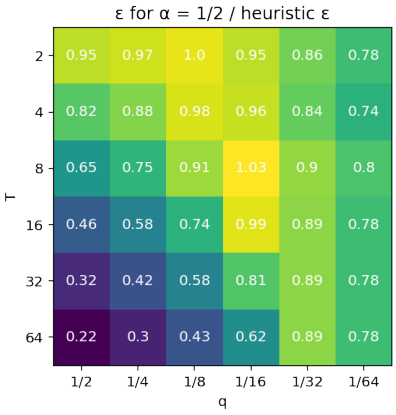
In light of this counterexample, in the next subsection, we benchmark our counterexample against sweeping over smaller values of T. I.e., we consider $\max_{t \le T} \varepsilon_{t,q,\sigma}(\delta)$ instead of simply $\varepsilon_{T,q,\sigma}(\delta)$.

4.2 Linear Loss + Quadratic Regularizer

Consider running DP-SGD in one dimension (i.e., $d = 1$) with a linear loss $\ell(\mathbf{m}, x) = \mathbf{m}x$ for the canary and a quadratic regularizer $r(\mathbf{m}) = \frac{1}{2}\alpha\mathbf{m}^2$, where $\alpha \in [0, 1]$ and $x \in [-1, 1]$ and we use learning rate $\eta = 1$. With sampling probability q, after T iterations the privacy guarantee is equivalent to distinguishing $Q := \mathcal{N}(0, \widehat{\sigma}^2)$ and $P := \mathcal{N}(\sum_{i \in [T]} (1-\alpha)^{i-1} \mathsf{Bernoulli}(q), \widehat{\sigma}^2)$, where $\widehat{\sigma}^2 := \sigma^2 \sum_{i \in [T]} (1-\alpha)^{2(i-1)}$. When $\alpha = 0$, this retrieves linear losses. When $\alpha = 1$, this corresponds to distinguishing $\mathcal{N}(0, \widehat{\sigma}^2)$ and $\mathcal{N}(\mathsf{Bernoulli}(q), \widehat{\sigma}^2)$ or, equivalently, to distinguishing linear losses after $T = 1$ iteration. If we maximize our heuristic over the number of iterations $\le T$, then our heuristic is tight for the extremes $\alpha \in \{0, 1\}$.



(a) One iteration of DP-SGD satisfies $(1, 10^{-6})$-DP.



(b) One iteration of DP-SGD satisfies $(2, 10^{-6})$-DP.

Figure 5: Ratio of upper bound on ε for quadratic loss with $\alpha = 0.5$ divided by maximum ε of i iterations on a linear loss. In Figure 5a (resp. Figure 5b), for each choice of q, σ is set so 1 iteration of DP-SGD satisfies $(1, 10^{-6})$-DP (resp $(2, 10^{-6})$-DP).

A natural question is whether the worst-case privacy guarantee on this quadratic is always given by $\alpha \in \{0, 1\}$. Perhaps surprisingly, the answer is no: we found that for $T = 3, q = 0.1, \sigma = 1, \alpha = 0$, DP-SGD is $(2.222, 10^{-6})$-DP. For $\alpha = 1$ instead DP-SGD is $(2.182, 10^{-6})$-DP. However, for $\alpha = 0.5$ instead the quadratic loss does not satisfy $(\varepsilon, 10^{-6})$-DP for $\varepsilon < 2.274$.

However, this violation is small, which suggests our heuristic is still a reasonable for this class of examples. To validate this, we consider a set of values for the tuple (T, q, σ). For each setting of T, q, σ, we compute $\max_{t \leq T} \varepsilon_{t,q,\sigma}(\delta)$ at $\delta = 10^{-6}$. We then compute ε for the linear loss with quadratic regularizer example with $\alpha = 1/2$ in the same setting. Since the support of the random variable $\sum_{i \in [T]} (1-\alpha)^{i-1} \mathsf{Bernoulli}(q)$ has size 2^T for $\alpha = 1/2$, computing exact ε for even moderate T is computationally intensive. Instead, let X be the random variable equal to $\sum_{i \in [T]} (1-\alpha)^{i-1} \mathsf{Bernoulli}(q)$, except we round up values in the support which are less than .0005 up to .0005, and then round each value in the support up to the nearest integer power of 1.05. We then compute an exact ε for distingushing $\mathcal{N}(0, \widehat{\sigma}^2)$ vs $\mathcal{N}(X, \widehat{\sigma}^2)$. By Lemma 4.5 of Choquette-Choo et al. (2024), we know that distinguishing $\mathcal{N}(0, \widehat{\sigma}^2)$ vs. $\mathcal{N}(\sum_{i \in [T]} (1-\alpha)^{i-1} \mathsf{Bernoulli}(q), \widehat{\sigma}^2)$ is no harder than distingushing $\mathcal{N}(0, \widehat{\sigma}^2)$ vs $\mathcal{N}(X, \widehat{\sigma}^2)$, and since we increase the values in the support by no more than 1.05 multiplicatively, we expect that our rounding does not increase ε by more than 1.05 multiplicatively.

In Figure 5, we plot the ratio of ε at $\delta = 10^{-6}$ for distingushing between $\mathcal{N}(0, \widehat{\sigma}^2)$ and $\mathcal{N}(X, \widehat{\sigma}^2)$ divided by the maximum over $i \in [T]$ of ε at $\delta = 10^{-6}$ for distinguishing between $\mathcal{N}(0, i\sigma^2)$ and $\mathcal{N}(\mathsf{Binomial}(i, q), i\sigma^2)$. We sweep over T and q, and for each q In Figure 5a (resp. Figure 5b) we set σ such that distinguishing $\mathcal{N}(0, \sigma^2)$ from $\mathcal{N}(\mathsf{Bernoulli}(q), \sigma^2)$ satisfies $(1, 10^{-6})$-DP (resp. $(2, 10^{-6})$-DP). In the majority of settings, the linear loss heuristic provides a larger ε than the quadratic with $\alpha = 1/2$, and even when the quadratic provides a larger ε, the violation is small ($\leq 3\%$). This is evidence that our heuristic is still a good approximation for many convex losses.

4.3 Pathological Example

If we allow the regularizer r to be arbitrary – in particular, not even requiring continuity – then the gradient can also be arbitrary. This flexibility allows us to construct a counterexample such that the standard composition-based analysis of DP-SGD with `intermediate_iterates` is close to tight. Specifically, choose the regularizer so that the update $\mathbf{m}' = \mathbf{m} - \eta \nabla_{\mathbf{m}} r(\mathbf{m})$ does the following: $\mathbf{m}'_1 = 0$ and, for $i \in [d-1]$, $\mathbf{m}'_{i+1} = v \cdot \mathbf{m}_i$. Here $v > 1$ is a large constant. We chose the loss so that, for our canary x_1, we have $\nabla_{\mathbf{m}} \ell(\mathbf{m}, x_1) = (1, 0, 0, \cdots, 0)$ and, for all other examples x_i ($i \in \{2, 3, \cdots, n\}$), we have $\nabla_{\mathbf{m}} \ell(\mathbf{m}, x_i) = \mathbf{0}$. Then the last iterate is

$$\mathbf{m}_T = (A_T + \xi_{T,1}, vA_{T-1} + v\xi_{T-1,1} + \xi_{T,2}, v^2 A_{T-2} + v^2 \xi_{T-2,1} + v\xi_{T-1,2} + \xi_{T,3}, \cdots), \quad (7)$$

where $A_t \leftarrow \text{Bernoulli}(p)$ indicates whether or not the canary was sampled in the t-th iteration and $\xi_{t,i}$ denotes the i-th coordinate of the noise ξ_t added in the t-th step. Essentially, the last iterate $\mathbf{m}_T$ contains the history of all the iterates in its coordinates. Namely, the i-th coordinate of $\mathbf{m}_T$ gives a scaled noisy approximation to A_{T-i}:

$$v^{1-i}\mathbf{m}_{T,i} = A_{T-i} + \sum_{j=0}^{i-1} v^{j+1-i}\xi_{T-j,i-j} \sim \mathcal{N}\left(A_{T-i}, \sigma^2 \frac{1-v^{-2i}}{1-v^{-2}}\right). \tag{8}$$

As $v \to \infty$, the variance converges to σ^2. In other words, if v is large, from the final iterate, we can obtain $\mathcal{N}(A_i, \sigma^2)$ for all i. This makes the standard composition-based analysis of DP-SGD tight.

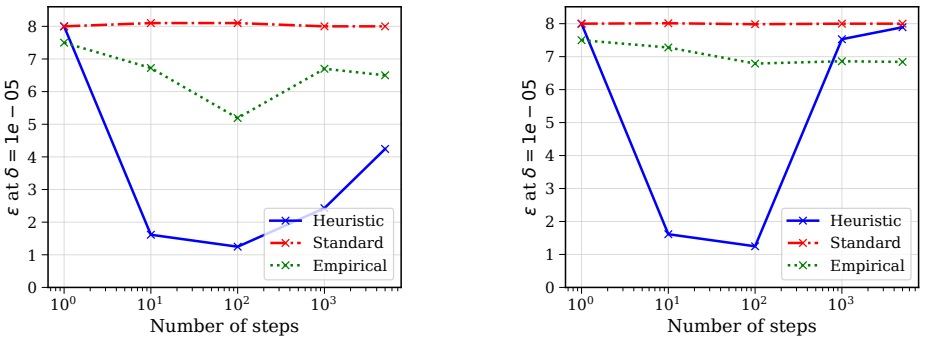
Concurrent work (Annamalai, 2024) presents a counterexample that yields a similar conclusion, although their construction is quite different.

4.4 Malicious Dataset Attack

The examples above rely on the regularizer having large unclipped gradients. We now construct a counterexample without a regularizer, instead using other examples to amplify the canary signal.

Our heuristic assumes the adversary does not have access to the intermediate iterations and that the model is linear. However, we can design a nonlinear model and specific training data to directly challenge this assumption. The attack strategy is to use the model's parameters as a sort of noisy storage, saving all iterations within them. Then with access only to the final model, an adversary can still examine the parameters, extract the intermediate steps, and break the assumption. Our construction introduces a data point that changes its gradient based on the number of past iterations, making it easy to identify if the point was present at a given iteration of training. The rest of the data points are maliciously selected to ensure the noise added during training doesn't impact the information stored in the model's parameters. We defer the full details of the attack to Appendix C.

Figure 6 summarizes the results. As illustrated in the figure, this attack achieves an auditing lower bound matching the standard DP-SGD analysis even in the `last_iterate_only` setting. As a result, the attack exceeds our heuristic. However, this is a highly artificial example and it is unlikely to reflect real-world scenarios.



(a) $q = 0.01$ (b) $q = 0.1$

Figure 6: In this adversarial example, the attack encodes all training steps within the final model parameters, thereby violating the specific assumptions used to justify our heuristic analysis.

5 Discussion & Implications

Both theoretical analysis and privacy auditing are valuable for understanding privacy leakage in machine learning, but both have limitations. Theoretical analysis is inherently conservative, while auditing procedures evaluate only specific attacks, and may thus underrepresent the privacy leakage.

Table 2: Previous works showed that large batch sizes achieve high performing models (De et al., 2022). Using our heuristic analysis it is possible to achieve similar performance for smaller batch sizes.

Batch size	Heuristic ε	Standard ε	Accuracy	Empirical ε
4096	6.34	8	79.5%	1.7
512	7.0	12	79.9%	1.8
256	6.7	14	78.2%	1.6

Our heuristic approach empirically upper bounds the privacy leakage in practical scenarios (Section 3). Although counter-examples with higher leakage exist (Section 4), these are highly contrived and not representative of real-world models.

The current best practice for differentially private training is to use high sub-sampling rates (i.e., large batch sizes) (De et al., 2022; Ponomareva et al., 2023). Our experimental results (Tables 1 and 2) challenge this best practice. We find that batch size does not significantly affect either the model performance or the privacy as measured by the heuristic epsilon or privacy auditing, but it does affect the standard privacy analysis significantly. This indicates that the larger batch sizes may not be necessary for better privacy.

The current trend in training large models is to gather very large datasets and use a small number of epochs. Moreover, as models get larger using large batch sizes becomes impractical as it slows down in the training pipeline. This is the setting where the gap between the standard analysis and our heuristic is large (see Figure 1b). Furthermore, training large models is too costly to perform proper empirical privacy evaluations.

Another trend is to report increasingly large differential privacy epsilon values based on the standard privacy analysis. It is hard to justify such values. Our heuristic analysis, along with empirical privacy evaluation, allows for more grounded claims in such settings. Of course, our heuristic analysis relies on linearity assumptions, but to exceed the bounds of our analysis, an adversary would need to break these linearity assumptions in a practical setting. This is currently very difficult with existing attacks on deep learning models.

6 Conclusion

A major drawback of differentially private machine learning is the gap between theoretical and practical threat models. Theoretical analyses often assume the adversary has access to every training iteration, while in practice, training typically occurs in closed environments, limiting adversaries to the final model. This disconnect, in combination with our limited understanding of deep learning loss landscapes, slowed down progress in theoretical privacy analysis of private training of deep learning models. As a result many studies supplement theoretical privacy bounds with empirical leakage evaluations to justify their choice of epsilon, this entails high computational costs and requires careful attack optimizations, making them both expensive and error-prone.

Our work introduces a novel heuristic analysis for DP-SGD that focuses on the privacy implications of releasing only the final model iterate. This approach is based on the empirical observation that linear loss functions accurately model the effectiveness of state of the art membership inference attacks. Our heuristic offers a practical and computationally efficient way to estimate privacy leakage to complement privacy auditing and the standard composition-based analysis.

Our heuristic avoids the cost of empirical privacy evaluations, which could be useful, e.g., for selecting hyperparameters a priori. More importantly, the scientific value of our heuristic is that it sets a target for further progress. Privacy auditing work should aim to exceed our heuristic, while theoretical work should aim to match it.

We acknowledge the limitations of our heuristic with specific counterexamples that demonstrate the heuristic underestimates the true leakage in some contrived examples. However, we show that in practical scenarios, our heuristic provides an upper bound on all existing empirical attacks.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016. URL https://arxiv.org/abs/1607.00133.

Michael Aerni, Jie Zhang, and Florian Tramèr. Evaluations of machine learning privacy defenses are misleading. *arXiv preprint arXiv:2404.17399*, 2024. URL https://arxiv.org/abs/2404.17399.

Krizhevsky Alex. Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/kriz/learning-features-2009-TR.pdf*, 2009.

Jason Altschuler and Kunal Talwar. Privacy of noisy stochastic gradient descent: More iterations without more privacy loss. *Advances in Neural Information Processing Systems*, 35:3788–3800, 2022. URL https://arxiv.org/abs/2205.13710.

Galen Andrew, Peter Kairouz, Sewoong Oh, Alina Oprea, H Brendan McMahan, and Vinith Suriyakumar. One-shot empirical privacy estimation for federated learning. *arXiv preprint arXiv:2302.03098*, 2023. URL https://arxiv.org/abs/2302.03098.

Meenatchi Sundaram Muthu Selva Annamalai. It's our loss: No privacy amplification for hidden state dp-sgd with non-convex loss. *arXiv preprint arXiv:2407.06496*, 2024. URL https://arxiv.org/abs/2407.06496.

Meenatchi Sundaram Muthu Selva Annamalai and Emiliano De Cristofaro. Nearly tight black-box auditing of differentially private machine learning. *arXiv preprint arXiv:2405.14106*, 2024.

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*, pp. 464–473. IEEE, 2014. URL https://arxiv.org/abs/1405.7085.

Martin Bertran, Shuai Tang, Aaron Roth, Michael Kearns, Jamie H Morgenstern, and Steven Z Wu. Scalable membership inference attacks via quantile regression. *Advances in Neural Information Processing Systems*, 36, 2024. URL https://arxiv.org/abs/2307.03694.

Benjamin Bichsel, Timon Gehr, Dana Drachsler-Cohen, Petar Tsankov, and Martin Vechev. Dp-finder: Finding differential privacy violations by sampling and optimization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 508–524, 2018.

Jinho Bok, Weijie Su, and Jason M Altschuler. Shifted interpolation for differential privacy. *arXiv preprint arXiv:2403.00278*, 2024. URL https://arxiv.org/abs/2403.00278.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 1897–1914. IEEE, 2022. URL https://arxiv.org/abs/2112.03570.

Tudor Cebere, Aurélien Bellet, and Nicolas Papernot. Tighter privacy auditing of dp-sgd in the hidden state threat model. *arXiv preprint arXiv:2405.14457*, 2024. URL https://arxiv.org/abs/2405.14457.

Christopher A. Choquette-Choo, Arun Ganesh, Thomas Steinke, and Abhradeep Thakurta. Privacy amplification for matrix mechanisms, 2024.

Rishav Chourasia, Jiayuan Ye, and Reza Shokri. Differential privacy dynamics of langevin diffusion and noisy gradient descent. *Advances in Neural Information Processing Systems*, 34:14771–14781, 2021. URL https://arxiv.org/abs/2102.05855.

Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 475–489, 2018.

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 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 3*, pp. 265–284. Springer, 2006. URL https://www.iacr.org/archive/tcc2006/38760266/38760266.pdf.

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*, pp. 650–669. IEEE, 2015.

Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 521–532. IEEE, 2018. URL https://arxiv.org/abs/1808.06651.

Google. Differential privacy accounting, 2020. https://github.com/google/differential-privacy/tree/main/python/dp_accounting.

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 genetics*, 4(8):e1000167, 2008.

Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33: 22205–22216, 2020.

Antti Koskela, Joonas Jälkö, and Antti Honkela. Computing tight differential privacy guarantees using fft. In *International Conference on Artificial Intelligence and Statistics*, pp. 2560–2569. PMLR, 2020. URL https://arxiv.org/abs/1906.03049.

Klas Leino and Matt Fredrikson. Stolen memories: Leveraging model memorization for calibrated {White-Box} membership inference. In *29th USENIX security symposium (USENIX Security 20)*, pp. 1605–1622, 2020. URL https://arxiv.org/abs/1906.11798.

Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pp. 263–275. IEEE, 2017. URL https://arxiv.org/abs/1702.07476.

Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on security and privacy (SP)*, pp. 866–882. IEEE, 2021. URL https://arxiv.org/abs/2101.04535.

Milad Nasr, Jamie Hayes, Thomas Steinke, Borja Balle, Florian Tramèr, Matthew Jagielski, Nicholas Carlini, and Andreas Terzis. Tight auditing of differentially private machine learning. *arXiv preprint arXiv:2302.07956*, 2023. URL https://arxiv.org/abs/2302.07956.

Natalia Ponomareva, Hussein Hazimeh, Alex Kurakin, Zheng Xu, Carson Denison, H Brendan McMahan, Sergei Vassilvitskii, Steve Chien, and Abhradeep Guha Thakurta. How to dp-fy ml: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research*, 77:1113–1201, 2023. URL https://arxiv.org/abs/2303.00654.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pp. 5558–5567. PMLR, 2019. URL https://arxiv.org/abs/1908.11229.

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.

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)*, pp. 3–18. IEEE, 2017.

Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *arXiv preprint arXiv:2210.00597*, 2022. URL https://arxiv.org/abs/2210.00597.

Thomas Steinke, Milad Nasr, and Matthew Jagielski. Privacy auditing with one (1) training run. *Advances in Neural Information Processing Systems*, 36, 2023. URL https://arxiv.org/abs/2305.08846.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Florian Tramer, Andreas Terzis, Thomas Steinke, Shuang Song, Matthew Jagielski, and Nicholas Carlini. Debugging differential privacy: A case study for privacy auditing. *arXiv preprint arXiv:2202.12219*, 2022. URL https://arxiv.org/abs/2202.12219.

Yuxin Wen, Arpit Bansal, Hamid Kazemi, Eitan Borgnia, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Canary in a coalmine: Better membership inference with ensembled adversarial queries. In *ICLR*, 2023. URL https://arxiv.org/abs/2210.10750.

Jiayuan Ye and Reza Shokri. Differentially private learning needs hidden state (or much faster convergence). *Advances in Neural Information Processing Systems*, 35:703–715, 2022. URL https://arxiv.org/abs/2203.05363.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Ahmed Salem, Victor Rühle, Andrew Paverd, Mohammad Naseri, and Boris Köpf. Bayesian estimation of differential privacy. *arXiv preprint arXiv:2206.05199*, 2022.

Sajjad Zarifzadeh, Philippe Cheng-Jie Marc Liu, and Reza Shokri. Low-cost high-power membership inference by boosting relativity. 2023. URL https://arxiv.org/abs/2312.03262.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2204–2213, 2018.

A Proof of Theorem 1

Proof. Let x_{i^*} be the canary, let D be the dataset with the canary and D' be the dataset without the canary. Since ℓ and r are linear, wlog we can assume $r = 0$ and $\nabla_{\mathbf{m}_{t-1}} \ell(\mathbf{m}_{t-1}, x_i) = \mathbf{v}_i$ for some set of vectors $\{\mathbf{v}_i\}$, such that $\|\mathbf{v}_i\|_2 \leq 1$. We can also assume wlog $\|\mathbf{v}_{i^*}\| = 1$ since, if $\|\mathbf{v}_{i^*}\| < 1$, the final privacy guarantee we show only improves.

We have the following recursion for $\mathbf{m}_t$:

$$\mathbf{m}_t = \mathbf{m}_{t-1} - \eta \left(\sum_{i \in B_t} \mathbf{v}_i + \xi_t \right), \qquad \xi_t \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma^2 I_d).$$

Unrolling the recursion:

$$\mathbf{m}_t = \mathbf{m}_0 - \eta \left[\sum_{t \in [T]} \sum_{i \in B_t} \mathbf{v}_i + \xi \right], \qquad \xi \sim N(0, T\sigma^2 I_d).$$

By the post-processing property of DP, we can assume that in addition to the final model $\mathbf{m}_T$, we release $\mathbf{m}_0$ and $\{B_t \setminus \{x_{i^*}\}\}_{t\in[T]}$, that is we release all examples that were sampled in each batch except for the canary. The following f is a bijection, computable by an adversary using the released information:

$$f(\mathbf{m}_T) := -\left[\frac{\mathbf{m}_T - \mathbf{m}_0}{\eta} - \sum_{t\in[T]}\sum_{i\in B_t\setminus\{x_{i^*}\}} \mathbf{v}_i.\right]$$

Since f is a bijection, distinguishing $\mathbf{m}_T$ sampled using D and D' is equivalent to distinguishing $f(\mathbf{m}_T)$ instead. Now we have $f(\mathbf{m}_T) = \mathcal{N}(0, T\sigma^2 I_d)$ for D', and $f(\mathbf{m}_T) = \mathcal{N}(0, T\sigma^2 I_d) + k\mathbf{v}_{i^*}$, $k \sim \mathsf{Binomial}(T, q)$. For any vector $\mathbf{u}$ orthogonal to $\mathbf{v}_{i^*}$, by isotropy of the Gaussian distribution the distribution of $\langle f(\mathbf{m}_T), \mathbf{u}\rangle$ is the same for both D and D' and independent of $\langle f(\mathbf{m}_T), \mathbf{v}_{i^*}\rangle$, hence distinguishing $f(\mathbf{m}_T)$ given D and D' is the same as distingushing $\langle f(\mathbf{m}_T), \mathbf{v}_{i^*}\rangle$ given D and D'. Finally, the distribution of $\langle f(\mathbf{m}_T), \mathbf{v}_{i^*}\rangle$ is exactly P for D and exactly Q for D'. By post-processing, this gives the theorem.

We can also see that the function $\delta_{T,q,\sigma}$ is tight (i.e., even if we do not release $B_t \setminus \{x_{i^*}\}$), by considering the 1-dimensional setting, where $\mathbf{v}_i = 0$ for $i \neq i^*$ and $\mathbf{v}_{i^*} = -1, \eta = 1, \mathbf{m}_0 = 0$. Then, the distribution of $\mathbf{m}_T$ given D is exactly P, and given D' is exactly Q. □

A.1 Computing δ from ε

Here, we give an efficiently computable expression for the function $\delta_{T,q,\sigma}(\varepsilon)$. Using P, Q as in Theorem 1, let $f(y)$ be the privacy loss for the output y:

$$\begin{aligned} f(y) = \log\left(\frac{P(y)}{Q(y)}\right) &= \log\left(\sum_{k=0}^{T}\binom{T}{k}q^k(1-q)^{n-k}\frac{\exp(-(y-k)^2/2T\sigma^2)}{\exp(-y^2/2T\sigma^2)}\right) \\ &= \log\left(\sum_{k=0}^{T}\binom{T}{k}q^k(1-q)^k\exp\left(\frac{2ky-k^2}{2T\sigma^2}\right)\right). \end{aligned}$$

Then for any ε, using the fact that $S = \{y : f(y) \geq \varepsilon\}$ maximizes $P(S) - e^\varepsilon Q(S)$, we have:

$$\begin{aligned} H_{e^\varepsilon}(P, Q) &= P(\{y : f(y) \geq \varepsilon\}) - e^\varepsilon Q(\{y : f(y) \geq \varepsilon\}) \\ &= P(\{y : y \geq f^{-1}(\varepsilon)\}) - e^\varepsilon Q(\{y : y \geq f^{-1}(\varepsilon)\}) \\ &= \sum_{k=0}^{T}\binom{T}{k}q^k(1-q)^k \Pr[\mathcal{N}(k, T\sigma^2) \geq f^{-1}(\varepsilon)] - e^\varepsilon \Pr[\mathcal{N}(0, T\sigma^2) \geq f^{-1}(\varepsilon)]. \end{aligned}$$

Similarly, $S = \{y : f(y) \leq -\varepsilon\}$ maximizes $Q(S) - e^\varepsilon P(S)$ so we have:

$$\begin{aligned} H_{e^\varepsilon}(Q, P) &= Q(\{y : f(y) \leq -\varepsilon\}) - e^\varepsilon P(\{y : f(y) \leq -\varepsilon\}) \\ &= Q(\{y : y \leq f^{-1}(-\varepsilon)\}) - e^\varepsilon P(\{y : y \leq f^{-1}(-\varepsilon)\}) \\ &= \Pr[\mathcal{N}(0, T\sigma^2) \leq f^{-1}(-\varepsilon)] - e^\varepsilon \sum_{k=0}^{T}\binom{T}{k}q^k(1-q)^k \Pr[\mathcal{N}(k, T\sigma^2) \leq f^{-1}(-\varepsilon)]. \end{aligned}$$

These expressions can be evaluated efficiently. Since f is monotone, it can be inverted via binary search. We can also use binary search to evaluate ε as a function of δ. Pseudocode for this is provided in the next subsection.

A.2 PSEUDOCODE FOR COMPUTING ε AND δ VALUES

```
def f(y,n,q,sigma):
  """computes log likelihood ratio of Binomial(n,q)+N(0,sigma^2) over N(0,sigma^2)"""
  if q==1: # k=n
    return (2*y-n)/(2*sigma**2)
  if q==0: # k=0
    return 0
  terms = [0]*(n+1)
  # terms[k] = log({n \choose k} * q^k * (1-q)^{n-k} * exp((2ky-k^2)/2\sigma^2))
  logpck = 0  # log({n \choose 0})
  for k in range(n):
    terms[k] = logpck + k*math.log(q) + (n-k)*math.log1p(-q) + (2*y-k)*k/(2*n*sigma**2)
    # {n \choose k+1} = {n \choose k} * (n-k)/(k+1)
    logpck = logpck + math.log(n-k) - math.log(k+1)
  assert abs(logpck) <= 1e-9  # log({n \choose n}) = 0
  terms[n] = n*math.log(q) + (2*y-n)*n/(2*n*sigma**2)
  # rescale so largest term is 1
  a = max(terms)
  return math.log(math.fsum(math.exp(t-a) for t in terms)) + a

def finv(eps,n,q,sigma):
  """computes inverse of f(y) i.e. y such that f(y)=eps"""
  if q < 1 and eps <= n * math.log1p(-q):
    # smallest possible value is f(-inf) = log((1-q)^n)
    # so no inverse exists
    # note q=1 implies f(-inf)=-inf so inverse exists
    return None
  # f(y) is increasing function of y so we can use binary search
  ymin, ymax = -1, 1
  while f(ymin,n,q,sigma) > eps:
    ymin *= 2
  while f(ymax,n,q,sigma) < eps:
    ymax *= 2
  while True:
    y = (ymin+ymax)/2
    val = f(y,n,q,sigma) - eps
    if abs(val) < 1e-9:
      break
    elif val > 0:
      ymax = y
    else:
      ymin = y
  return y

def gauss_tail(mean,var,threshold):
  """computes P[N(mean,var)>=threshold]"""
  return 0.5*math.erfc((threshold-mean)/math.sqrt(2*var))
```

```python
def binomial_gaussian_dp_delta(iterations, probability, noise_multiplier, eps, sign=0):
  """computes delta corresponding to (eps,delta)-DP of
   Binomial(n,q)+N(0,sigma^2) vs N(0,sigma^2)

  n=iterations, q=probability, sigma=noise_multiplier,
  sign=1 computes hockey stick divergence for Binomial(n,q)+N(0,sigma^2) over N(0,sigma^2),
  sign=-1 computes hockey stick divergence for N(0,sigma^2) over Binomial(n,q)+N(0,sigma^2)
  sign=0 computes max delta over both cases.

  Note that sigma needs to be multiplied by sqrt(iterations) to match the paper.
  """
  if sign != 1 and sign != -1:  # return max(delta_+,delta_-)
    delta_plus = binomial_gaussian_dp_delta(iterations, probability, noise_multiplier, eps, sign=1)
    delta_minus = binomial_gaussian_dp_delta(iterations, probability, noise_multiplier, eps, sign=-1)
    return max(delta_plus, delta_minus)
  # otherwise return delta_sign
  y = finv(sign * eps, iterations, probability, noise_multiplier)
  if y is None: return 0  # no inverse, in this case delta=0
  y = sign * y
  if probability==1 or probability==0: # k=n or k=0
    if probability==1: k = iterations
    if probability==0: k = 0
    if sign == 1:  # delta_+
      return gauss_tail(k,iterations*noise_multiplier**2,y) - math.exp(eps)*gauss_tail(0,iterations*noise_multiplier**2,y)
    if sign == -1:  # delta_-
      return gauss_tail(0,iterations*noise_multiplier**2,y) - math.exp(eps)*gauss_tail(-k,iterations*noise_multiplier**2,y)
  terms = [0]*(iterations+1)
  logpck = 0
  for k in range(iterations):
    terms[k] = math.exp(logpck + k*math.log(probability) + (iterations-k)*math.log1p(-probability)) * gauss_tail(sign*k,iterations*noise_multiplier**2,y)
    logpck = logpck + math.log(iterations-k) - math.log(k+1)
  assert abs(logpck) <= 1e-9
  terms[iterations] = (probability**iterations) * gauss_tail(iterations,iterations*noise_multiplier**2,y)
  if sign == 1:
    return math.fsum(terms) - math.exp(eps)*gauss_tail(0,iterations*noise_multiplier**2,y)
  if sign == -1:
    return gauss_tail(0,iterations*noise_multiplier**2,y) - math.exp(eps) * math.fsum(terms)

def binomial_gaussian_dp_eps(iterations, probability, noise_multiplier, delta, sign=0):
```

```
  """computes eps corresponding to (eps,delta)-DP of
  ↪  Binomial(n,q)+N(0,sigma^2) vs N(0,sigma^2)"""
  epsmin,epsmax = 0, 1
  while binomial_gaussian_dp_delta(iterations, probability,
  ↪  noise_multiplier, epsmax, sign=sign) > delta:
    epsmax += 1
  while True:
    eps = (epsmin+epsmax)/2
    val = binomial_gaussian_dp_delta(iterations, probability,
    ↪  noise_multiplier, eps, sign=sign) - delta
    if abs(val) < 1e-9:
      return eps
    elif val < 0:
      epsmax = eps
    else:
      epsmin = eps
```

B Linear Worst Case for Full Batch Setting

It turns out that in the full-batch setting, the worst-case analyses of DP-GD with `intermediate_iterates` and with `last_iterate_only` are the same. This phenomenon arises because there is no subsampling (because $q = 1$ in Algorithm 1) and thus the algorithm is "just" the Gaussian mechanism. Intuitively, DP-GD with `intermediate_iterates` corresponds to T calls to the Gaussian mechanism with noise multiplier σ, while DP-GD with `last_iterate_only` corresponds to one call to the Gaussian mechanism with noise multiplier $\sigma/\sqrt{T}$; these are equivalent by the properties of the Gaussian distribution.

We can formalize this using the language of Gaussian DP (Dong et al., 2019): DP-GD (Algorithm 1 with $q = 1$) satisfies $\sqrt{T}/\sigma$-GDP. (Each iteration satisfies $1/\sigma$-GDP and adaptive composition implies the overall guarantee.) This means that the privacy loss is exactly dominated by that of the Gaussian mechanism with noise multiplier $\sigma/\sqrt{T}$. Linear losses give an example such that DP-GD with `last_iterate_only` has exactly this privacy loss, since the final iterate reveals the sum of all the noisy gradient estimates. The worst-case privacy of DP-GD with `intermediate_iterates` is no worse than that of DP-GD with `last_iterate_only`. The reverse is also true (by postprocessing).

In more detail: For T iterations of (full-batch) DP-GD on a linear losses, if the losses are (wlog) 1-Lipschitz and we add noise $\mathcal{N}(0, \frac{\sigma^2}{n^2} \cdot I)$ to the gradient in every round, distinguishing the last iterate of DP-SGD on adjacent databases is equivalent to distinguishing $\mathcal{N}(0, T\sigma^2)$ and $\mathcal{N}(T, T\sigma^2)$. This can be seen as a special case of Theorem 1 for $p = 1$, so we do not a give a detailed argument here.

If instead we are given every iteration $\mathbf{m}_t$, for *any* 1-Lipschitz loss, distinguishing the joint distributions of $\mathbf{m}_t$ given $\mathbf{m}_{t-1}$ on adjacent databases is equivalent to distinguishing $\mathcal{N}(0, \sigma^2)$ and $\mathcal{N}(1, \sigma^2)$. In turn, distinguishing the distribution of all iterates on adjacent databases is equivalent to distinguishing $\mathcal{N}(\mathbf{0}^T, \sigma^2 I_T)$ and $\mathcal{N}(\mathbf{1}^T, \sigma^2 I_T)$, where $\mathbf{0}^T$ and $\mathbf{1}^T$ are the all-zeros and all-ones vectors in $\mathbb{R}^T$. Because the Gaussian distribution is isotropic, distinguishing $\mathcal{N}(\mathbf{0}^T, \sigma^2 I_T)$ and $\mathcal{N}(\mathbf{1}^T, \sigma^2 I_T)$ is equivalent to distinguishing $\langle \mathbf{x}, \mathbf{1}^T \rangle$ where $\mathbf{x} \sim \mathcal{N}(\mathbf{0}^T, \sigma^2 I_T)$ and $\langle \mathbf{x}, \mathbf{1}^T \rangle$ where $\mathbf{x} \sim \mathcal{N}(\mathbf{1}^T, \sigma^2 I_T)$. These distributions are $\mathcal{N}(0, T\sigma^2)$ and $\mathcal{N}(T, T\sigma^2)$, the exact pair of distributions we reduced to for last-iterate analysis of linear losses.

C Malicious Dataset Attack Details

Algorithms 2, 3, and 4 summarizes the construction for the attack. The attack assume the model parameters have dimension equal to the number of iterations. It also assumes each data point can reference which iteration of training is currently happening (this can be implemented by having

a single model parameter which increments in each step, independently of the training examples, without impacting the privacy of the training process). Then we build our two datasets D and $D' = D \cup \{x\}$ so that all points in dataset D ("repeaters") run Algorithm 3 to compute gradients and the canary point in D' runs Algorithm 2 to compute its gradient. Our attack relies heavily on DP-SGD's lack of assumptions on the data distribution and any specific properties of the model or gradients. Algorithm 2, which generates the canary data point, is straightforward. Its goal is to store in the model parameters whether it was present in iteration i by outputting a gradient that changes only the i-th index of the model parameters by 1 (assuming a clipping threshold of 1).

All other data points, the "repeaters", are present in both datasets (D and D'), and have three tasks:

- Cancel out any noise added to the model parameters at an index larger than the current iteration. At iteration i, their gradients for parameters from index i onward will be the same as the current value of the model parameter, scaled by the batch size and the learning rate to ensure this parameter value will be 0 after the update.
- Evaluate whether the canary point was present in the previous iteration by comparing the model parameter at index $i-1$ with a threshold, and rewrite the value of that model parameter to a large value if the canary was present.
- Ensure that all previous decisions are not overwritten by noise by continuing to rewrite them with a large value based on their previous value.

To achieve all of these goals simultaneously, we require that the batch size is large enough that the repeaters' updates are not clipped.

Finally Algorithm 4 runs DP-SGD, with repeater points computing gradients with Algorithm 3 and the canary point, sampled with probability p, computing its gradient using Algorithm 2. In our experiments we run Algorithm 4 100,000 times. And to evaluate if the model parameters was from dataset D or D' we run a hypothesis test on the values of the model parameters. All constants are chosen to ensure all objectives of the repeaters are satisfied.

Algorithm 2 Canary data point

function ADV($\mathbf{x}$, i)
    Initialize $\mathbf{a}$ as a zero vector of the same dimension as $\mathbf{x}$
    Set $a_i \leftarrow 1$ ▷ Set the i-th component to 1
    return $-\mathbf{a}$
end function

Algorithm 3 Additional data points

Require: model parameters $\mathbf{x}$, iteration number i, batch size N, learning rate η, previous history threshold t_{past}, last iteration threshold t_{last}, history amplification value BIG_VAL

function REPEATERS($\mathbf{x}$, i, N, η, t_{past}, t_{last}, BIG_VAL)
    $\mathbf{h} \leftarrow \mathbf{x}_{0:i}$ ▷ Parameter "history" up to iteration i, not inclusive
    $\mathbf{f} \leftarrow \mathbf{x}_{i:end}$ ▷ Future and current parameters, starting from iteration i
    $\mathbf{f} \leftarrow -\mathbf{f}/(\eta \cdot N)$ ▷ Remove noise from last iteration
    base_history $\leftarrow -\mathbf{x}_{0:i}/(\eta \cdot N)$ ▷ By default, zero out entire history
    if length($\mathbf{h}$) > 1 **then**
        $\mathbf{h}_{0:i-1} \leftarrow$ BIG_VAL$/(\eta \cdot N) \cdot (2\mathbb{1}[\mathbf{h}_{0:i-1} \geq t_{\text{past}}] - 1)$ ▷ If an old iteration is large enough, it was a canary iteration, so amplify it
    end if
    if length($\mathbf{h}$) > 0 **then**
        $\mathbf{h}_{i-1} \leftarrow$ BIG_VAL$/(\eta \cdot N) \cdot (2\mathbb{1}[\mathbf{h}_i \geq t_{\text{last}}] - 1)$ ▷ If the last iteration is large enough, it was a canary iteration, so amplify it
    end if
    $\mathbf{h} \leftarrow \mathbf{h} +$ base_history ▷ Don't zero out canary iterations
    $\mathbf{a} \leftarrow$ concatenate($\mathbf{h}, \mathbf{f}$)
    return $-\mathbf{a}$
end function

Algorithm 4 Encoding Attacking

Require: add-diff, whether to add the canary, batch size N, sampling rate p, learning rate (η), iteration count/parameter count D

function RUN_DPSGD(add-diff)
  $C \leftarrow 1$
  Initialize model $\mathbf{m} \leftarrow \mathbf{0}$ of dimension D
  for $i = 0$ to D **do**
    Generate a uniform random value $q \in [0, 1]$
    $\mathbf{r} \leftarrow$ repeaters$(\mathbf{m}, i)$
    Compute norm $c \leftarrow ||\mathbf{r}||$
    if $c > 0$ **then**
      Normalize $\mathbf{r} \leftarrow \mathbf{r} / \max(c, C)$
    end if
    Adjusted vector $\mathbf{z} \leftarrow \mathbf{r} \times N$
    Verify condition on $\mathbf{m}_i$
    if $p \leq q$ and add-diff **then**
      $\mathbf{r} \leftarrow$ adv$(\mathbf{m}, i)$
      Normalize and update $\mathbf{z}$
    end if
    Apply Gaussian noise to $\mathbf{z}$
    Update model $\mathbf{m} \leftarrow \mathbf{m} - \mathbf{z} \times \eta$
  end for
  return $\mathbf{m}$
end function

D Implementation Details

For image classification model we used De et. al, (De et al., 2022) training approach with different batch sizes. We used Google Cloud `A2-megagpu-16g` machines with 16 Nvidia A100 40GB GPUs. Overall, we use roughly 60,000 GPU hours for our experiments. To compute the standard theoretical privacy bounds we use DP-accounting library [3]. For each experiment we trained the model of image classification setting we trained 1000 models with and without the canary (i.e, differing example).

We implemented fine tuning with differential privacy on Gemma 2 (Team et al., 2024) in JAX. The fine tuning is done in FP16. The attack here directly looks at the difference in the embedding table of the model for the tokens used as canaries before and after the finetuning. We trained 128 model with and without carries for language model fine tuning experiments.

For the experiments with different batch sizes, we fixed the training iterations to 60 and 2 epochs for image and text modalities respectively and modified the sub-sampling rates, calculating the noise parameters required to achieve a similar privacy guarantee based on our heuristic.

[3] https://github.com/google/differential-privacy/tree/main