Towards Improving Differentially Private Machine Learning

Alireza Akbari
Simon Fraser University

1 Introduction

Machine learning (ML) models are susceptible to privacy attacks. For instance, Carlini et al. has shown language models memorize sensitive training data such as social security numbers which could be extracted by an adversary, leaking the privacy of users' data [4]. Additionally, Shokri et al. proposed an efficient membership inference attack [19]. Such attacks have API access to the prediction of a target model, meaning that they have only access to the prediction of the target model, nothing more. Now, Given an arbitrary sample, either included in the training dataset of a model or not included, MI attacks are able to tell whether the sample was included in the training set of the model or not. Such attacks could make users afraid of sharing their sensitive data to train a model on the dataset including their private information. As a defense, Abadi et al. proposed to take advantage of the notion of differential privacy to privatize Deep Learning models [1]. Differential privacy (DP) is a property of a randomized algorithm that is typically run on a database. It guarantees that if you change one row of the database, it leads to a bounded change in the distribution over the possible outcomes of the algorithm [7]. It summarizes the bounded change in a set of parameters called (ε, δ).

Based on the DP definition, Abadi et al. proposed to privatize stochastic gradient descent (SGD) algorithm, one of the most popular training algorithms for neural networks [1]. The intuition is that if the computation of the gradients with respect to training samples holds the DP property, then no sample could contribute much to the final learned model. To convert SGD to a private algorithm (called DP-SGD), they basically clip the per-sample gradients and add isotropic Gaussian noise with variance σ to the backpropagated gradients. Basically, stochastic gradient descent queries the dataset multiple times to compute the gradients based on them. Differential privacy provides tools to quantify the privacy leakage of each query, but it doesn't provide proper tools to analyze the privacy leakage of all of the steps together. Therefore, it is not easy to quantify the optimal privacy budgets of the DP-SGD. It is important to obtain the optimal budgets, since the variance of the gaussian noise is actually dependent on them, and adding less noise would lead to better model accuracy. Accounting, in DP jargon, means calculating the DP bounds for a privatized algorithm. For DPSGD, it means that if you privatize each step of SGD with (ε, δ), what would be the overall bounds for the whole algorithm (all steps). However, accounting methods do not certify that they provide the optimal accounting, leading to a trend of works trying to get better privacy bounds than DPSGD [6, 15]. And it was in question when this trend comes to an end.

Another line of works in differentially private machine learning started to evaluate the privacy leakage of models practically, meaning that they designed attacks against DP ML models and quantified the privacy parameter ε in practice [11, 12, 16]. Essentially, they provided a lower bound for ε, meaning that it is possible to leak from the model at least to the extent of the lower bound using their attacks. The key observation was that there is a huge gap between the DPSGD proposed bounds (which is actually an upper bound) and the quantified lower bounds. This brings the two lines of work together with this question: Is the observed gap because of the pessimistic accounting of DPSGD and we could still improve the bounds, or is it because of the lack of enough power of the practical attacks [16]. Nasr et al. [16] found an attack which could break the gap and reached the upper bound proposed by Abadi et al. [1]. This would suggest that the privacy accounting of DP-SGD is tight, and one cannot analyze its privacy in another way to obtain better privacy guarantees. This highlights our assumption that in order to obtain better privacy guarantees for DP-SGD, one should change something in the algorithm of DP-SGD, not analyze it in another way.

Now, the research problem is whether we could get better privacy guarantees for DP-SGD while having the same utility? For this problem, as inputs, we are given the private training algorithm DP-SGD, its corresponding noise amount injected to the algorithm represented as σ, and its correspond-

28th USENIX Security Symposium Template

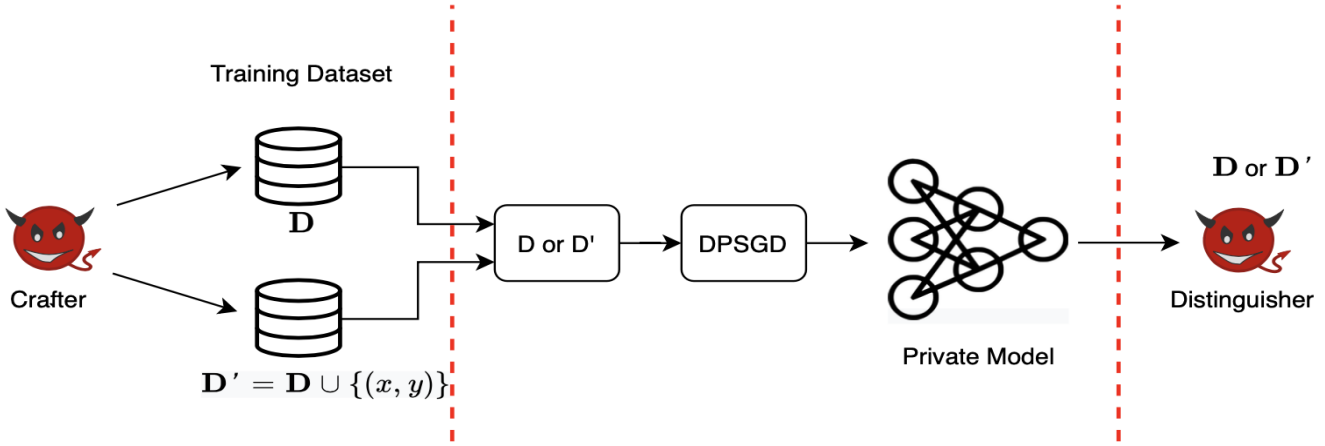

Figure 1: Adversarial game proposed by Nasr et al. [16]. The purpose of this design is to make the adversary guess which dataset was used to train the model.

ing privacy guarantees in terms of (ε, δ). As outputs, we want to find a new private modified version of DP-SGD with the same amount of noise σ, and have better privacy (ε', δ').

In this work, we want to study the role of initialization of the weights of the neural network model which is trained by DP-SGD. Jagielski et al. [11] performed an experiment in which an adversary iteratively has to leak the privacy of a model on a newly trained model at each step. They found out that variation of initialization in the iterations could fool the adversary to leak the privacy. However, they did not address how initialization could be taken advantage of in just one iteration, instead of multiple iterations, which is not the case usually in practice. Therefore, their justification of their observation is not satisfactory.

We are going to check the role of initialization more closely by looking at the learning dynamics of neural networks in DP-GD parlance which is shown to be an ordinary differential equation [3]. We will study how initialization could affect the learning dynamics of DP-GD. We also run experiments to study the learning dynamics more closely. Particularly, we tried to understand to what extent one training sample could change the final learned model, which is consistent to the definition of differential privacy.

2 Background & Related Work

2.1 Data Privacy

From now on, two words dataset and database are going to be used interchangeably. Imagine a database in which each row corresponds to a person's data, and each row has a set of features (columns). These features could be classified into three broad categories: identifiers, which have a one-to-one correspondence to the identity of a person (such as name), quasi-identifiers, which could be associated with a person's identity (such as age), and sensitive attributes, which are the user's private information in the database (such as their disease).

Historically, there have been multiple attempts to protect the privacy of such databases. One of the earliest ones that got people's attention was the notion of k-anonymity [18]. In k-anonymity, the goal is to release the database completely in a way to preserves the privacy of each individual in the database. How k-anonymity works, in a nutshell, is that it removes identifiers from the database. Subsequently, it perturbs the database in a way that for any setting of quasi-identifiers, there should be at least $k-1$ other rows in the database with the same quasi-identifiers (it is called a k-anonymity set). Now, if an attacker could guess a user's identity by its quasi-identifiers, it couldn't necessarily guess its private information since there could be k different values for the private information. However, it is totally possible that the people in a k-anonymity set all have the same private information, making the attacker leak the privacy of the target person completely [9]. This serious issue made k-anonymity research reaches dead ends.

2.2 Differential Privacy

Major issues in private data releasing motivated exploring alternative methods to preserve data privacy. Imagine there is a curator having a private dataset, and there is an analyst wants to perform data analysis based on the private dataset. Differential privacy, as a de facto standard nowadays, suggests that the curator could only let the analyst query the private database (or run algorithms on) in a way that the output of the query (algorithm) doesn't reveal that much about the sensitive data of any individual in the database. To do so, the curator could add a certain amount of noise to the result of the query,

and report the noisy version of the result.

To be more concrete about what differential privacy means, consider this example. Consider that there is a data collection process asking people to announce their salaries in the database in order to enable some downstream data analysis based on the salaries of people. Alice as an individual is worried about the privacy of her salary by contributing to the data collection. Now, the curator claims that the queries such as the average salary of the people in the dataset would not be different that much whether Alice was in the dataset or not. That's what the term "differential" means. Therefore, Alice is more willing to contribute knowing the fact that its private data doesn't make much difference in the queries to the database.

As mentioned earlier, the curator provides privacy by perturbing the output of the query by a specific noise. In other words, the curator is sacrificing the utility of the queries in order to preserve the privacy of individuals present in the dataset. We will see how the noise is going to be designed. Moreover, it is not meaningful for the curator to just tell Alice that her privacy would be preserved without providing any specific quantification of the extent that which Alice's privacy would be preserved. This brings us to the mathematical formalism of differential privacy.

Before any formulation of differential privacy, it is necessary to understand the notion of neighboring datasets. Database D' is a neighboring database of $D \in \mathcal{D}^n$ if they differ in exactly one entry. For instance, D could be the dataset of the salaries of people, and D' includes D and Alice's salary when she decides to share her salary.

Definition 1 *Let f be a randomized algorithm $f : \mathcal{D}^n \to \mathcal{Y}$. It satisfies $(\varepsilon, \delta)-$differential privacy if for all neighboring databases $D, D' \in \mathcal{D}$ and for all $\mathcal{S} \subset \mathcal{R}$*

$$Pr(f(D) \in \mathcal{S}) \leq e^{\varepsilon} Pr(f(D') \in \mathcal{S}) + \delta \tag{1}$$

Note that here, the source of randomness is the noise added by the curator, and therefore, the probability is defined over the noise distribution. Consequently, the algorithm results in a probability distribution. Differential privacy makes the probability distribution of the output of the algorithm run over D be similar to the probability distribution of the output of the algorithm run over D'. In other words, Alice does not incur that much change to the distribution of the output of the algorithm. Differential privacy proposes to quantify the amount of closeness of these two distributions by two parameters (ε, δ) which are called privacy budgets. These parameters are summarizing the privacy guarantee provided by DP quantitatively. Therefore, the curator could inform Alice about the privacy of the algorithms run over the database, and Alice has a clear image of the extent that the privacy is guaranteed.

There are multiple observations from definition 1 to make. Note that this definition is a worst-case upper-bound guarantee since it requires that the inequality holds for all neighboring datasets and any output set. $\mathcal{S}$. For a fixed δ, it is easy to see that smaller ε means better privacy. However, it is not obvious at all to compare two pairs of $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$ with each other in terms of their privacy guarantees (for instance, a case in which $\varepsilon_1 < \varepsilon_2$, but $\delta_1 > \delta_2$). We are going to see how this would affect the composition of DP algorithms. These kinds of problems of (ε, δ)-DP led to the definition of other notions of differential privacy. [6, 14]

Differential privacy has multiple properties and aspects, but we are going to discuss how to make an algorithm differentially private, and how to compose multiple differentially private algorithms with each other because analysts query databases multiple times, not just once.

2.2.1 DP Mechanisms

Now, the question is how an algorithm can be transformed to a differentially private algorithm. For this, it is necessary to define the notion of sensitivity. Let $f : \mathcal{D}^n \to \mathcal{Y}^k$. Then the ℓ_2-sensitivity of f is:

$$\Delta_2^{(f)} = \max_{D,D'} \|f(D) - f(D')\|_2 \tag{2}$$

where D and D' could be any neighboring databases. The motivation behind sensitivity in the context of differential privacy should be clear. Since differential privacy is limiting the contribution of each individual in the algorithm, it is natural to study the sensitivity of the algorithm f in order to understand its maximum change over neighboring datasets. In the running example, if Alice is a millionaire with an outlier salary, then it could be the case that Alice's inclusion in the database could leak its privacy, cause for example it could change the average salary a lot. Therefore, sensitivity could inform the curator how much noise should be added to the output of the algorithm.

Gaussian mechanism, is one of the well-known used mechanisms used to privatize an algorithm f. Let $f : \mathcal{D}^n \to \mathcal{Y}^k$. Let Z_i $N(o, 2ln(\frac{1.25}{\delta})\frac{\delta_2^2}{\varepsilon^2})$. Then the Gaussian mechanism is defined as:

$$M(D) = f(D) + (Z_1, \ldots, Z_k) \tag{3}$$

It is proved that if you apply the Gaussian mechanism to an algorithm f, it would be (ε, δ)-differentially private. [7] Note that if an algorithm has an unbounded sensitivity, it cannot be privatized by this mechanism. In the running example, if Alice has a way too high salary, then it is impossible for the curator to guarantee privacy while Alice is included in the database. One solution that people use while dealing with algorithms with unbounded sensitivity is to clip the maximum value that the function could output by a specific threshold. In other words, they force the algorithm to have a specific sensitivity. For instance, if a person has a salary of more than \$100k, it would be clipped and modified to \$100k. This notion of clipping would be important when we study DP-SGD.

2.2.2 DP Composition

One of the most useful properties of differential privacy is its natural ability to compose, meaning that if you queried a database k times with their corresponding privacy parameters, you can easily compose the queries and compute how much privacy is guaranteed over the whole k queries. This is essential since naturally, an analyst would like to query the database multiple times, even adaptively based on the results of previous queries. As in Alice's example, an analyst, after finding out about the average income, could be curious about how many people are paid less than the mean. The first composition technique that differential privacy provides is as follows.

Theorem 1 ***Basic Composition***. *Let $M = (M_1, \ldots, M_k)$ where M_i is $(\varepsilon_i, \delta_i)$-DP, and each of them can be chosen either in advance or sequentially based on the previous mechanisms $M_j (j < i)$. Then, M is $(\Sigma_{i=1}^{k} \varepsilon_i, \Sigma_{i=1}^{k} \delta_i)$-DP.*

However, theorem 1 does not provide the tightest guarantee for the big composed mechanism, meaning that the composed query could satisfy a better $\varepsilon_1, \delta_1)$ but basic composition provided a loose upper bound $\varepsilon_2, \delta_2)$ in which $\varepsilon_2 > \varepsilon_1, \delta_2 > \delta_1$. Note that finding the optimal (ε, δ) is quite important since, in the gaussian mechanism, the variance of the noise that is injected into the model is dependent on $\varepsilon, \delta)$. Therefore, with a loose upper bound, the curator has to inject noise with a larger variance which could ruin the utility of the algorithm. It has been shown that composing $(\varepsilon_i, \delta_i)$-DP mechanisms could be NP-hard, because of the problem of impossibility to compare (ε, δ) pairs, mentioned earlier [13]. We are not expanding that much on the literature on composition, since it only affects one of the assumptions related to our research problem, and it is necessary to know in order to understand what privacy auditing means.

2.3 Differentially-Private Stochastic Gradient Descent (DP-SGD)

Differentially private stochastic gradient descent is the most used private machine learning algorithm. The algorithm that DP-SGD privatize is the gradients of a neural network. It applies the Gaussian mechanism to the gradients computed from the backpropagation. As mentioned earlier, for the gaussian mechanism, it is necessary that the sensitivity is bounded. Since gradients could be unbounded potentially, DP-SGD clips the norm of gradients using a threshold C. The algorithm of DP-SGD can be found in algorithm 1.

In order to analyze the privacy guarantees of DP-SGD, one should use the composition tools of DP to find the guarantees over the whole DP-SGD algorithm. Abadi et al. [1] themselves proposed a new technique to account the privacy, which itself is a significant improvement to the prior tools such as basic composition [8]. However, it was still doubted whether the proposed bound is tight or not. Moreover, the role of subsampling at each step because of the stochastic gradient descent also has a huge impact on the privacy guarantees of the algorithm. This can be explained by the fact that when an algorithm run on a datset is private, it is more private when it is run on a subset of the first dataset, which is called privacy amplification by subsampling [2].

Algorithm 1 Differentially private Stochastic Gradient Descent

Require: Parameters: subsampling probability p, number of iterations T, learning rate η_t, noise scale σ, group size L, gradient norm bound C.
Initialize $_0$ randomly
for $t \in [T]$ **do**
  Take a random sample L_t with sampling probability p
  Compute gradient
  For each $i \in L_t$, compute $v_t(x_i) \leftarrow \nabla_{\mathbf{w}_t} \ell(\mathbf{w}_t, x_i)$
  Clip gradient
  $C_i = min\{1, C/\|v_t(x_i)\|\}$
  $\bar{v}_t(x_i) \leftarrow C_i . v_t(x_i)$
  Add noise
  $\tilde{v}_t \leftarrow \frac{1}{L} \left(\sum_i \bar{v}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2) \right)$
  Descent
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \tilde{v}_t$
Output $\mathbf{w}_T$ and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Another line of work in differentially private machine learning started to audit differentially private ML models, meaning that they evaluated the privacy leakage of models practically to obtain a lower bound for (ε, δ). If the practical computed lower bound could reach the proposed theoretical upper bound, then one could claim that the proposed upper bound was really tight. This line of work is described more in the next section.

2.4 Auditing Differentially Private Machine Learning

In order to calculate lower bound ε, Nasr et al. [16] proposed an iterative game, which can be found in Figure 1. In a nutshell, in this game the adversary (which includes two parts, the crafter and the distinguisher) crafts two neighboring datasets and passes them to the trainer. The trainer randomly chooses one of the datasets and trains a private model on that using DPSGD. Then, the distinguisher has to detect which dataset is used to train the model. Since this is an iterative game, there would be a set of false positives and false negatives collected at the end of the game. Based on the DP definition, one could show that the lower bound ε can be calculated as:

$$\varepsilon = max\{log \frac{1 - \delta - FP}{FN}, log \frac{1 - \delta - FN}{FP}\} \qquad (4)$$

In this work, Nasr et al. [16] came up with an attack in which the lower bound (ε, δ) could reach the proposed upper bound by Abadi et al. [1]. In a nutshell, in their dataset attack, they designed the neighboring datasets in a way that the gradients with respect to any points other than (x, y) would be zero. In this case, the initial random model would be changed only if it includes (x, y) in its training set. Hence, the distinguisher could definitely detect which dataset was used to train the model. This result suggests that the privacy analysis of DP-SGD is already tight. Hence, other factors in DP-SGD should be changed to obtain better privacy. Therefore, one assumption of our research problem is that we can't analyze the privacy of DP-SGD better, and we should look for ways to improve the algorithm itself.

There has been another similar work that audited differentially private machine learning models by Jagielski et al. [11]. Similarly, they trained machine learning models iteratively to obtain lower bounds of (ε, δ). One experiment that they additionally performed was as follows. In one case, at each step of the game, they used the same initialization for all of the models. In another case, they used different initialization for the models at each step of the game. They found out it was hard for the adversary to leak privacy when they used different initializations at each step. This is the major motivating observation of our work. They did not provide any justification of why this happens, and they did not include any observation of what the role of initialization is in just one training of DP-SGD.

2.5 Gradient Flow in Neural Networks

With the advent of the notion of neural tangent kernels [10], which in a nutshell means that a deep learning model could be thought of as a linear model with a kernel (with a few assumptions), it becomes much more easier the learning dynamics of a neural network model and its convergence behavior. To do so, one natural thing to study is the gradient flow. Gradient flow, in other words, is basically the gradient descent with infinitesimal step size. The gradient flow for a usual neural network trained by gradient descent is:

$$\frac{d\mathbf{w}(t)}{dt} = -\frac{1}{n}\sum_{i}^{n} \nabla_{\mathbf{w}} \ell_i(t) \tag{5}$$

where n is the total number of samples in the training set. Note that it is not to write the same thing for stochastic gradient descent, since it gets updated by a batch of samples at each step.

However, the same learning dynamics cannot be applied to differentially private gradient descent, since it differs from normal stochastic gradient descent in two ways, as described earlier in algorithm 1. Consequently, it is necessary to rewrite it for DP-GD. One thing to notice here is that it should be rewritten for DP-GD, not DP-SGD (mentioned earlier why). Bu et al. [3] showed how it would change in this scenario.

Theorem 2 ***Training Dynamics in DP-GD***. *Let* $\mathbf{w}$ *be the weights of 1, if the step size* $\eta \to 0$*, then the continuous gradients flow of DP-GD would be the following ordinary differential equation.*

$$\frac{d\mathbf{w}(t)}{dt} = -\frac{1}{n}\sum_{i}^{n} \nabla_{\mathbf{w}} \ell_i(t) C_i(t) \tag{6}$$

where n is the total number of samples in the training set, and $C_i(t)$ *is the clipping factor for the i-th sample.*

The proof sketch is easy to see since as $\eta \to 0$, the expectation and variance of the normal distribution (noise distribution) would become zero since η would appear in the expectation and variance of this random variable by basic probability theory facts. Therefore, the additive noise term can be replaced with zero, and just the other terms would be preserved in the equation.

3 Our approach

First, we have to think of how to analyze the role of the initialization of weights of neural networks in the privacy of the model. As we saw, the most natural way of analyzing the privacy of a mechanism is by composition. However, initialization is only a part of the model training dynamics that does not depend on the dataset at all. Therefore, the tools of composition are not useful in this case, since they are useful when we analyze algorithms using sensitive data.

Another possible direction is to think of it in this way. Suppose the model $f_{\mathbf{w}}$ is learned by DP-GD on dataset D with initialization of Z, and the model $f_{\mathbf{w}'}$ is learned by DP-GD on dataset D' with initialization of Z'. If we could study the difference between $\mathbf{w}$ and $\mathbf{w}'$ ($\|\mathbf{w} - \mathbf{w}'\|$), it could also be useful to analyze privacy. The reason is that, in differential privacy, it is important that $\mathbf{w}$ and $\mathbf{w}'$ do not become too different from each other.

We build on the latter approach. To do so, we build on the gradient flow of DP-GD and study whether we are able to find a relation between $f_{\mathbf{w}}$ and $f_{\mathbf{w}'}$.

3.1 Results

3.1.1 Privacy Analysis of Initialization

Suppose $\mathcal{T}$ is DP-GD which takes dataset D and initialization Z, and outputs the learned weights $W \leftarrow \mathcal{T}(D, Z)$. Similarly, $W' \leftarrow \mathcal{T}(D', Z')$ where D' is the neighboring dataset, and Z' is another initialization. Then

First, rewrite theorem 2 for following cases. First, when DP-GD took dataset D and initialization Z ($W \leftarrow \mathcal{T}(D, Z)$).

$$\frac{dW(t)}{dt} = -\frac{1}{n}\sum_{i}^{n} \nabla_{W} \ell_i(t) C_i(t), \quad W(0) = Z \tag{7}$$

Second, when DP-GD took dataset D' and initialization Z' ($W' \leftarrow \mathcal{T}(D', Z')$)

$$\frac{dW'(t)}{dt} = -\frac{1}{n+1}\sum_{i}^{n+1} \nabla_{W'} \ell_i(t) C_i(t), \quad W'(0) = Z' \tag{8}$$

The bound of the summation is over $n+1$, because D' also includes Alice's data. We can rewrite this ordinary differential equation in this way:

$$\frac{dW'(t)}{dt} = -\frac{1}{n+1}\left[\sum_{i}^{n} \nabla_{W'} \ell_i(t) C_i(t) - \nabla_{W'} \ell_{n+1}(t) C_{n+1}(t)\right], \tag{9}$$

$$W'(0) = Z' \tag{10}$$

Here, $\nabla_{W'} \ell_{n+1}(t) C_{n+1}(t)$ is the corresponding term to the extra sample in D'. From a worst-case perspective, this sample could lead to the most privacy leakage when it has a large gradient. In other words, one point is affecting the gradients, and subsequently the learned model, at the most. It is necessary to take a worst-case perspective here, since the nature of differential privacy deals with worst-case scenarios. Therefore, in the worst case, we assume the gradient of the new point is always clipped during training (although this could be a harsh assumption since we know that the scale of gradients are going to be decreased throughout the training). Therefore, we can replace $\nabla_{W'} \ell_{n+1}(t) C_{n+1}(t)$ with C, which is the clipping threshold.

$$\frac{dW'(t)}{dt} = -\frac{1}{n+1}\left[\sum_{i}^{n} \nabla_{W'} \ell_i(t) C_i(t) - C\right], W'(0) = Z' \tag{11}$$

3.2 Analysis

First observation is that the gradient flow ordinary differential equation for D' (11) is a perturbation of the corresponding ordinary differential equation for D (7). A perturbed version of an ordinary differential equation is natural and faced with different problems. As a simple physical example, consider an object falling free without considering gravity, just dealing with friction and having an initial velocity. Now, consider the same object with gravity affecting it. The corresponding ordinary differential equations of these two objects are just different by a constant perturbation. Currently, we are studying to analyze the relation of the solutions of these two ordinary differential equations. For instance, if it could let us compute $\|W(\infty) - W'(\infty)\|$, then we could analyze the privacy better.

More specifically, we are dealing with two ODEs with these forms:

1. $\frac{dy(t)}{dt} = f(y(t)), \quad y(0) = Z$
2. $\frac{dh(t)}{dt} = f(h(t)) + C, \quad h(0) = Z'$

At first glance, it seems not to be easy to understand the relation between the solutions of these two. To analyze the relation of $y(t)$ and $h(t)$, f has an important effect. For example, if it was monotonic (like the example in the falling objects), it was easier to analyze. For instance, if it was monotonically decreasing, then the solution reaches a steady state.

3.3 Empirical Evaluation

To study how the final models would differ from each other in each case, we also performed empirical experiments. From a high-level perspective, the experiment is in this way. Once we train a model on a dataset and obtain the final parameters. Subsequently, we just include a new sample to the dataset and train the model and obtain the new parameters. Note that we performed the experiment in a way that the only different factor between these two cases is the extra sample. To do so, we build on the Opacus library, which is a library in PyTorch enabling differential privacy [17].

3.3.1 Model

The model is a simple convolutional model with two consecutive convolutional layers followed by two dense layers.

3.3.2 Dataset

We used simple MNIST dataset for this experiment [5]. MNIST has 60000 training samples and 10000 test samples. For our experiment, we take one of the samples in the training set randomly, and train a model on 59999 samples and a model on 60000 samples.

3.3.3 Experiments

First, we have to think of how to design the experiment to be similar enough to the theoretical setting. In our first experiment, we tried to perform gradient descent to train our model on neighboring versions of MNIST. However, this setting is not stil the same as equation 11, since there we assumed that the gradient of the extra sample is always equal to the clipping threshold. We leave this requirement to future work. The other difference is that the initialization for the both cases is the same.

However, the setting of first experiment turned out to not be an effective setting. The reason is that when we use gradient descent (or full batch gradient descent), it is more likely to be stuck in a local minima comparing to stochastic gradient descent. Consequently, they usually have way less accuracy which also was seen in our experiment in table 1. Note that the privacy guarantee is much more bad since in DP-GD, we don't have the amplification by subsampling phenomenon. Therefore, it doesn't make sense to use a useless model as our baseline.

Dataset	Training Algorithm	Baseline Test Accuracy	Baseline Privacy Budget ε
MNIST	DP-GD	47.10%	19.05%
MNIST	DP-SGD	90.98%	0.8%

Table 1: Difference between the baseline model when trained on DP-GD and DP-SGD

Therefore, we have to move towards using DP-SGD. In this new experiment, at each batch, we intentionally include the extra sample to affect the update of weights. This could be thought of that there are multiple copies of the sample in the training set. If we only leave the sample to affect in each epoch, it is likely that it couldn't make any difference at all. Using this scenario, we train the model on both cases for 10 epochs. Again notice that the only different thing is just the extra sample. After training both cases, we calculate $\|W - W'\|$ as a comparison metric between the weights of them (the norm here is the frobenius norm). We perform this experiment for 20 samples, and report the average over these 20 runs, which is shown in table 2.

We compared the weights by computing the frobenius norm of their difference. The problem with this metric is that we necessarily don't know how much difference can be tolerated as a private difference. Nevertheless, larger difference in the first layer comparing to the second dense layer could be not consistent with our intuition about neural networks, since last layers are learning more specific features than first layers.

We also leave the experiments on changing the initialization between the two runs to future works.

4 Conclusion & Future Works

Limitation One major limitation of our approach is that it couldn't let us directly discuss the privacy parameters (ε, δ), which are the only natural quantifications of privacy that people care about. For future works, it is natural to study the relation of the solutions of those two differential equations (7, 11) more closely, in order to further analyze how close or far the final learned weights could be from each other. Additionally, we would experiment the two mentioned future works earlier, about initialization and clipping threshold. **Conclusion** In conclusion, in this work, we analyzed the role of initialization in a privacy of a machine learning model. There has been no theoretical method to analyze this problem before, and we suggest studying the learning dynamics and the gradients flow of neural networks in order to compare the final models learned in neighboring datasets. We showed that the ordinary differential equation of the weights corresponding to the neighboring dataset D' is just a perturbation of the differential equations corresponding to D.

References

[1] 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*, pages 308–318, 2016.

[2] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. *Advances in Neural Information Processing Systems*, 31, 2018.

[3] Zhiqi Bu, Hua Wang, Qi Long, and Weijie J Su. On the convergence and calibration of deep learning with differential privacy. *arXiv preprint arXiv:2106.07830*, 2021.

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

[5] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

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

[7] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[8] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[9] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273, 2008.

[10] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Weight	Difference when trained on D and D'
First Convolution Layer	0.095
Second Convolution Layer	0.176
First Dense Layer	0.250
Second Dense Layer	0.135

Table 2: Difference between each layer weights when trained on D and D'. The reported average is computed over 20 distinct samples of D and D'.

[11] 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.

[12] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, 2019.

[13] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[14] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

[15] Ilya Mironov, Kunal Talwar, and Li Zhang. R\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

[16] Milad Nasr, Shuang Songi, Abhradeep Thakurta, Nicolas Papemoti, and Nicholas Carlin. Adversary instantiation: Lower bounds for differentially private machine learning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 866–882. IEEE, 2021.

[17] Opacus PyTorch library. Available from opacus.ai.

[18] Pierangela Samarati and Latanya Sweeney. Generalizing data to provide anonymity when disclosing information. In *PODS*, volume 98, pages 10–1145, 1998.

[19] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.