CS 480/680: Introduction to Machine Learning Winter 2021

Lecture 23 — Differentially Private Machine Learning

Prof. Gautam Kamath

Machine learning algorithms are frequently trained on sensitive data sets, containing private information belonging to individuals. We will see that without proper care, machine learning systems might leak this sensitive information. We will then discuss differential privacy, a strong notion of data privacy which is intended to prevent disclosures of this nature. We conclude by discussing its application to a differentially private version of stochastic gradient descent.

The Netflix Prize

One case study of data anonymization gone wrong is the Netflix Prize competition. Netflix is a very data-driven and statistically-minded company: many of their hit TV shows are conceived based on user data, and their fabled recommendation algorithm is tuned to optimize user engagement. Between 2006 and 2009, they hosted a contest, challenging researchers to improve their recommendation engine. The grand prize was a highly-publicized US$1,000,000, claimed by a team named BellKor's Pragmatic Chaos, based on matrix factorization techniques.

In order to help teams design their strategies, Netflix provided a training dataset of user data. Each datapoint consisted of an (anonymized) user ID, movie ID, rating, and date. Netflix assured users that the data was appropriately de-anonymized to protect individual privacy. Indeed, the Video Privacy Protection Act of 1988 requires them to do this. One's media consumption history is generally considered to be sensitive or private information, as one might consume media associated with certain minority groups (including of a political or sexual nature).

Unfortunately, Narayanan and Shmatikov demonstrated that this naïve form of anonymization was insufficient to preserve user privacy [NS08]. Their approach is illustrated in Figure 1. They took the dataset provided by Netflix, and cross-referenced it with public information from the online movie database IMDb, which contains hundreds of millions of movie reviews. In particular, they tried to match users between the two datasets by finding users who gave similar ratings to a movie at similar times. While the Netflix data was de-identified, the IMDb data was not, and a review was associated with either the user's name or an online pseudonym. It turns out this approach was sufficient to re-identify many users from only a few weak matches, thus giving information on these users' movie watching history, which they chose not to reveal publicly. This discovery led to a class action lawsuit being filed against Netflix, and the cancellation of a sequel competition.

This example shows that de-anonymization is insufficient to guarantee privacy, especially in the presence of side-information.

Memorization in Neural Networks

In the previous example, we tried to output an entire dataset, which seems to be challenging to appropriately privatize. What if we instead released some function or model of the dataset?

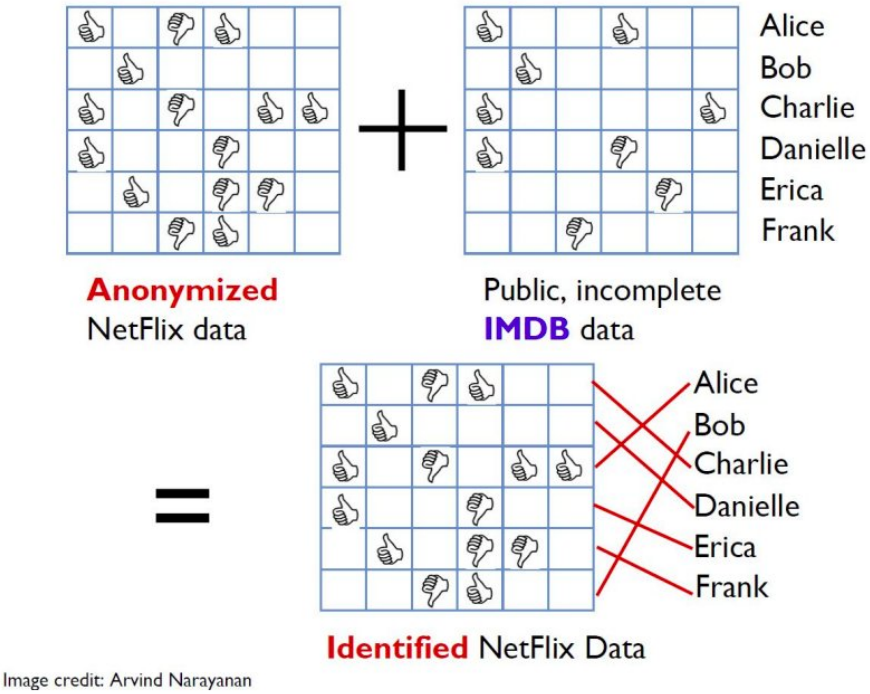

Figure 1: Figure due to Arvind Narayanan, illustrating the attack in [NS08].

As it only gives a restricted view of the dataset, perhaps this prevents it from revealing private information? Unfortunately, this is not the case.

We discuss an investigation of Carlini et al. [CLE+19]. Consider training a neural network model on a text corpus Y (potentially containing sensitive information), and creating a generative sequence model f_θ. Given a sequence $x_1, \ldots, x_n$, the model is able to compute

$$P_\theta(x_1, \ldots, x_n) = -\log_2 \Pr(x_1, \ldots, x_n | f_\theta) = \sum_{i=1}^{n} (-\log_2 \Pr(x_i | f_\theta(x_1, \ldots, x_{i-1}))) .$$

This quantity P_θ is known as the *log-perplexity* of the sequence. By inspecting the expression, it can be seen that a low perplexity indicates that the sequence is assigned a high probability by the model, and a high perplexity indicates that the sequence is assigned a low probability by the model. For instance, a well-trained language model is likely to assign a low perplexity score to a phrase like "Mary had a little lamb," but a high perplexity score to "correct horse battery staple."

The question is, what if "correct horse battery stapler" *were* in the training data? Would this lead to it having a low perplexity, thus signalling that this is the case? You might think that this is not a big deal (unless this is someone's password) – but what if instead, the phrase "my social security number is 078-05-1120" were assigned a low perplexity? This might reveal the SSN of an individual in the training data. There are two core questions here:

1. Do neural networks "memorize" "secrets" in the training data?
2. If so, is it possible to efficiently discover these secrets?

Carlini et al. [CLE+19] investigate these questions with the following experimental setup. They add a "canary"[1] to the training data, which is a sensitive phrase of a particular format – we will use the example "my social security number is 078-05-1120". The question is whether this inclusion significantly lowers the perplexity in comparison to other semantically similar phrases, such as "my social security number is 867-53-0900." If so, that is an indication that the particular canary has been memorized, and may be extractable from the final model. Note that on large datasets, the canary may have to be added many times before the perplexity becomes low enough to be noticed.

Let us be a bit more quantitative: suppose we have a set of phrases $\mathcal{R}$. For the present example, this will be the set of all phrases of the form "my social security number is ???-??-????," where each ? is a digit. Imagine sorting all of these phrases in increasing order of log-perplexity according to some model f_θ: the *rank* of the canary is its index in this list. A random element of $\mathcal{R}$ would fall somewhere in the middle of the list. On the other hand, elements at the top of the list are the best candidate secrets – we consider these to be more "exposed." With this mindset, the *exposure* of some secret $r \in \mathcal{R}$ is $\log_2 |\mathcal{R}| - \log_2 \operatorname{rank}(r)$. This value ranges from 0 to $\log_2 |\mathcal{R}|$, where a large exposure corresponds to the secret being more noticeable. In particular, an exposure of $\log_2 |\mathcal{R}|$ indicates that the secret would have the lowest log-perplexity of phrases in this list.

On the bright side, it seems like extremely large models are not prone to exposing secrets in this sense. In Figure 2, the results of an experiment on Google's Smart Compose are displayed. Smart Compose is an automatic sentence completion algorithm employed in Gmail, and is trained on billions of word sequences. Even when the canary is inserted thousands of times, the exposure remains comparatively low – $|R|$ used in this experiment is on the order of 10^{12}, so an exposure of > 40 is needed for extraction, whereas it only reaches values of 10 after $10,000$ insertions. On the other hand, Figure 3 illustrates a much smaller example, with a training dataset of size roughly $100,000$, and using the social security number example given above. As we can see, with only four insertions of the canary, its exposure passes $\log_2 10^9$, at which point it can easily be extracted.

In the paper, the authors also give efficient methods of extracting exposed secrets. Naïvely, one would have to try all possible sets of phrases and determine their perplexity. A more efficient approach is possible using a Dijkstra's algorithm style method.

Finally, the authors discuss how to mitigate such memorization and exposure. Interestingly, standard techniques to avoid overfitting in machine learning are ineffective, including dropout and regularization. Differential privacy appears to be the only effective approach.

Differential Privacy

In security and privacy, it is important to be precise about the precise setting in which we are working. We now define the setting for differential privacy, sometimes called *central differential privacy* or the *trusted curator* model. We imagine there are n individuals, X_1 through X_n, who each have their own datapoint. They send this point to a "trusted curator" – all individuals trust this curator with their raw datapoint, but no one else. Given their data, the curator runs an algorithm M, and publicly outputs the result of this computation. Differential privacy is a property of this algorithm M,[2] saying that no individual's data has a large impact on the output of the algorithm.

[1] A reference to the concept of a canary in a coal mine.

[2] In differential privacy lingo, an algorithm is sometimes (confusingly) called a "mechanism."

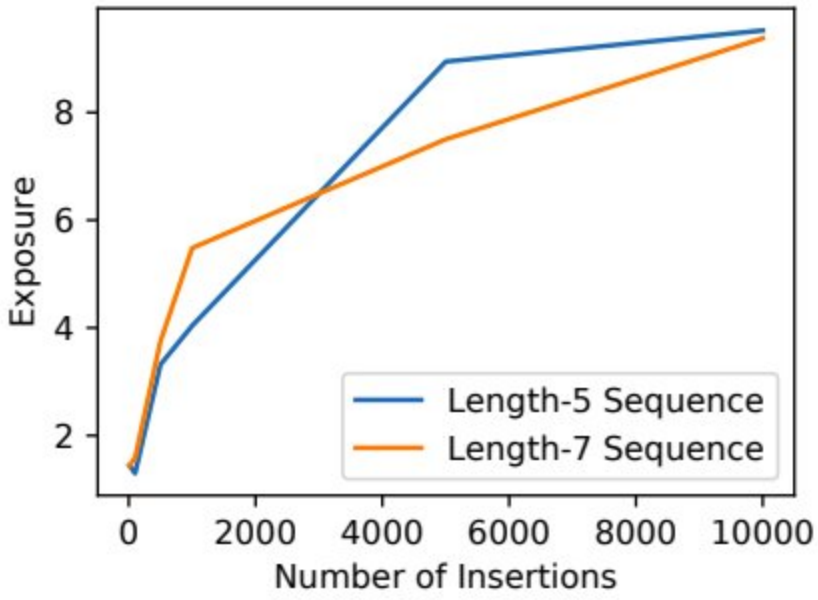

Figure 2: Figure from [CLE+19]. Even with 10000 insertions, the canary has relatively low exposure in the Google Smart Compose model.

More formally, suppose we have an algorithm $M : \mathcal{X}^n \to \mathcal{Y}$. Consider any two datasets $X, X' \in \mathcal{X}^n$ which differ in exactly one entry. We call these *neighbouring datasets*, and sometimes denote this by $X \sim X'$. We say that M is (ε, δ)-*differentially private* ((ε, δ)-DP) if, for all neighbouring X, X', and all $T \subseteq \mathcal{Y}$, we have

$$\Pr[M(X) \in T] \leq e^{\varepsilon} \Pr[M(X') \in T] + \delta,$$

where the randomness is over the choices made by M.

Differential privacy was defined by Dwork, McSherry, Nissim, and Smith in their seminal paper in 2006 [DMNS06] (with $\delta = 0$, the general version here is from [DKM+06]). It is now widely accepted as a strong and rigorous notion of data privacy. It has received acclaim in theory, winning the 2017 Gödel Prize, and the 2016 TCC Test-of-Time Award. At the same time, it has now seen adoption in practice at many organizations, including Apple [Dif17], Google [EPK14], Microsoft [DKY17], the US Census Bureau for the 2020 US Census [DLS+17], and much more.

Differential Privacy is an unusual sounding definition the first time you see it, so some discussion is in order.

- Differential privacy is quantitative in nature. A small ε corresponds to strong privacy, degrading as ε increases.
- ε should be thought of as a small-ish constant. Anything between (say) 0.1 and 5 might be a reasonable level privacy guarantee (smaller corresponds to stronger privacy), and you should be skeptical of claims significantly outside this range.
- δ should be much, much smaller, smaller than $1/n$. This is because very trivial but obviously non-private algorithms have $\delta = 1/n$. Namely, just choosing a random individual and outputting their data.

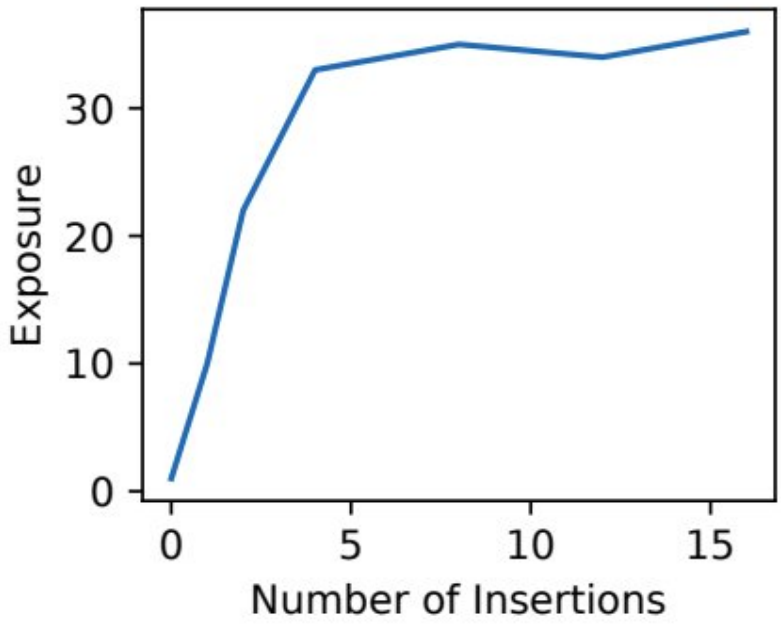

Figure 3: Figure from [CLE+19]. In a smaller example, the canary has very high exposure (and is recoverable) with only a few insertions.

- This is a worst-case guarantee, over all neighbouring datasets X and X'. Even if we expect our data to be randomly generated (and some realizations are incredibly unlikely), we still require privacy for all possible datasets nonetheless. While there do exist some notions of average-case privacy, these should be approached with caution – Steinke and Ullman write a series of posts which warn about the pitfalls of average-case notions of differential privacy [SU20a, SU20b].
- In words, the definition bounds the multiplicative increase (incurred by changing a single point in the dataset) in the probability of M's output satisfying any event.
- The use of a multiplicative e^{ε} in the probability might seem unnatural. For small ε, a Taylor expansion allows us to treat this as $\approx (1 + \varepsilon)$. The given definition is convenient because of the fact that $e^{\varepsilon_1} \cdot e^{\varepsilon_2} = e^{\varepsilon_1 + \varepsilon_2}$, which is useful when we examine the property of "group privacy" later.
- While the definition may look asymmetric, it is not: one can simply swap the role of X and X'.
- Convince yourself that any non-trivial (i.e., one that is not independent of the dataset) differentially private algorithm must be randomized.

Let's take a step back: what does differential privacy *mean*? Simply repeating the definition: differential privacy says that, the probability of any event is comparable in the cases when an individual does or does not include their data in the dataset. This has a number of implications of what differential privacy does and does not ensure.

First, it prevents many of the types of attacks we have seen before. The linkage-style attacks that we have observed are essentially ruled out – if such an attack were effective with your data in the

dataset, it would be almost as effective without. This holds true for existing auxiliary datasets, as well as any *future* data releases as well. It also prevents reconstruction attacks, in some sense "matching" the bounds shown in the Dinur-Nissim attacks [DN03], as we will quantify in a later lecture. In fact, it protects against *arbitrary* risks, which can be reasoned about by simply revisiting the fact that any outcome is comparably likely whether or not the individual's data was actually included.

Differential privacy does *not* prevent you from making inferences about individuals. Stated alternatively: differential privacy does not prevent statistics and machine learning. Consider the folklore "Smoking Causes Cancer" example. Suppose an individual who smokes cigarettes is weighing their options in choosing to participate in a medical study, which examines whether smoking causes cancer. They know that a positive result to this study would be detrimental to them, as it would cause their insurance premiums to rise. They also know that the study is being performed using differentially privately, so they choose to participate, and they know their privacy will be respected. Unfortunately for them, the study reveals that smoking does cause cancer! This is a privacy violation, right? No: differential privacy ensures that the outcome of the study would not be significantly impacted by their participation. In other words, whether they participated or not, the result was going to come out anyway. For more discussion of the compatibility of privacy and learning, see [McS16].

Differential privacy is also not suitable for the case where the goal is to identify a specific individual, and this is antithetical to the definition. As a timely example, despite the clamoring for privacy-preserving solutions for tracking the spread of COVID-19, it is not immediately clear how one could use differential privacy to facilitate *individual-level* contact tracing. This would seem to require information about where a specific individual has been, and which particular individuals they have interacted with. On the other hand, it might be possible to facilitate aggregate-level tracking, say if many people who tested positive all attended the same event. In this vein, there is some interesting work done by Google on DP analysis of location traces, to see which types of locations people spend more and less time at since COVID-19 struck [ABC+20].

The definition of differential privacy is information theoretic in nature. That is, an adversary with unlimited amounts of computational power and auxiliary information is still unable to get an advantage. This is in contrast to cryptography, which typically focuses on computationally bounded adversaries. There has been some work on models of differential privacy where the adversary is computational bounded, see, e.g., [BNO08].

Properties of Approximate Differential Privacy

One of the reasons that differential privacy is so popular is that it has a number of convenient properties. We simply state and discuss them, one can refer to [DR14, Vad17] for proofs.

Post-Processing

One convenient fact about differentially private algorithms is that once a quantity is privatized, it can't be "un-privatized," if the data is not used again. This is called closure under post-processing: if an algorithm is (ε, δ)-DP, then any post-processing is also (ε, δ)-DP.

Theorem 1. *Let $M : \mathcal{X}^n \to \mathcal{Y}$ be (ε, δ)-differentially private, and let $F : \mathcal{Y} \to \mathcal{Z}$ be an arbitrary randomized mapping. Then $F \circ M$ is (ε, δ)-differentially private.*

Group Privacy

So far, we've discussed differential privacy with respect to neighbouring datasets – ones which differ in exactly one entry. But one might wonder about datasets which differ in multiple entries. The definition of differential privacy allows for the guarantee to decay gracefully as the distance is increased. This is called *group privacy*.

Theorem 2. *Let $M : \mathcal{X}^n \to \mathcal{Y}$ be an (ε, δ)-differentially private algorithm. Suppose X and X' are two datasets which differ in exactly k positions. Then for all $T \subseteq \mathcal{Y}$, we have*

$$\Pr[M(X) \in T] \leq \exp(k\varepsilon) \Pr[M(X') \in T] + ke^{(k-1)\varepsilon}\delta.$$

Composition

As a final but important property, we discuss *composition* of differentially private algorithms. Suppose you ran k differentially private algorithms on the same dataset, and released all of their results – how private is this as a whole?

There are two composition theorems, one is the basic composition theorem which states that the ε's and δ's add up to give a final privacy guarantee. If all the ε's are the same, then running k algorithms degrades the privacy parameter by a factor of k.

Theorem 3. *Suppose $M = (M_1, \ldots, M_k)$ is a sequence of algorithms, where M_i is $(\varepsilon_i, \delta_i)$-differentially private, and the M_i's are potentially chosen sequentially and adaptively.*[3] *Then M is $(\sum_{i=1}^k \varepsilon_i, \sum_{i=1}^k \delta_i)$-differentially private.*

However, there is also an advanced composition theorem which says that we only need to pay a factor of $\sqrt{k}$.

Theorem 4. *Suppose $M = (M_1, \ldots, M_k)$ is a sequence of algorithms, where M_i is (ε, δ)-differentially private, and the M_i's are potentially chosen sequentially and adaptively. Then for any $\delta' > 0$, M is $(O(\varepsilon\sqrt{k}\log(1/\delta')), k\delta + \delta')$-differentially private.*

Gaussian Mechanism

The main way we make an algorithm differentially private is by adding noise. One of the main tools is the Gaussian mechanism. As the name suggests, this privatizes a statistic by adding Gaussian noise. Before we get to that, we define the sensitivity of a function.

Definition 5. *Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The ℓ_2-sensitivity of f is*

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

[3] While the algorithms themselves may be sequentially and adaptively chosen, the privacy parameters may not be – see [RRUV16] for more discussion.

where X and X' are neighbouring databases.

The Gaussian mechanism is as follows:

Definition 6. *Let $f : \mathcal{X}^n \to \mathbb{R}^k$. The* Gaussian mechanism *is defined as*

$$M(X) = f(X) + (Y_1, \ldots, Y_k),$$

where the Y_i are independent $N(0, 2\ln(1.25/\delta)\Delta_2^2/\varepsilon^2)$ random variables.

Note that we can also write this using the multivariate Gaussian as $f(X) + Y$, where $Y \sim N(0, 2\ln(1.25/\delta)\Delta_2^2/\varepsilon^2 \cdot I)$. This algorithm is (ε, δ)-DP.

Differentially Private Stochastic Gradient Descent

The current prevailing approach for differentially private machine learning is gradient perturbation, in which we perform a noisy form of gradient descent. This approach was first suggested by [WM10] and later by [SCS13], but it was most developed by Bassily, Smith, and Thakurta [BST14] (whose work we will be covering today). A work by Abadi, Chu, Goodfellow, McMahan, Mironov, Talwar, and Zhang [ACG+16] develops this method, making it more practical, and applying it to neural networks.

Before we talk about noisy gradient descent, we first recall stochastic gradient descent.

1. Select a random minibatch B of points from the dataset,
2. Compute the average gradient: $\frac{1}{|B|}\sum_{(x_i,y_i)\in B} \nabla \ell(\theta_t, x_i, y_i)$,
3. Take a step in the negative direction of the gradient.

The plan will be as follows: we would like to noise the gradient using the Gaussian mechanism as described above. However, the gradients could be arbitrarily large, and thus we must *clip* them. Namely, if a gradient has norm larger than some threshold C, we rescale it so that its norm is exactly C. This will lose some information from the gradients, but it seems to work well enough in practice.

1. Sample a "lot" of points of (expected) size L by selecting each point to be in the lot independently with probability L/n,
2. For each point (x_i, y_i) in the lot, compute the gradient of $\ell(\theta_t, x_i, y_i)$ and "clip" it to have ℓ_2 norm at most C.
3. Average these clipped gradients and add Gaussian noise.
4. Take a step in the negative direction of the resulting vector.

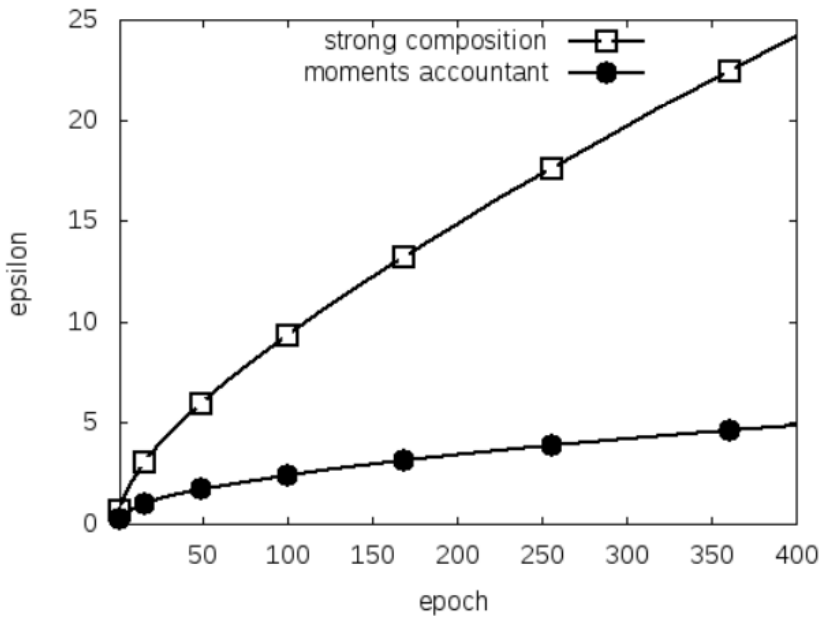

Figure 4: Figure from [ACG+16], showing the difference between using the improved analysis provided in their paper (called moments accountant) versus advanced composition.

This will result in each individual step of the algorithm being (ε, δ)-differentially private. Fortunately, properties such as composition of differential privacy allow us to reason about multiple steps of the algorithm. Specifically, running k steps of this algorithm will result in an overall algorithm which is (informally speaking) differentially private with parameter $\varepsilon\sqrt{k}$. This can be further improved by the fact that the first line subsamples each point with probability L/n, due to a property known as privacy amplification by subsampling (which we will not cover here). This will "boost" the privacy guarantee to roughly $\varepsilon\sqrt{k}L/n$. Even with all this work, the privacy guarantees would still be very weak and lossy, which is why more advanced methods known as moments accountant and Rényi differential privacy have been introduced [ACG+16, WBK19, MTZ19, Mir17]. We will not discuss these, but Figure 4 demonstrates the significant advantages that they afford.

The results are passable, but not phenomenal. For instance, one common benchmark is image classification on the MNIST dataset. One state-of-the-art result (based on DPSGD, but with additional tuning) gets 98.1% accuracy with $(1.2, 10^{-5})$-DP [TB21], whereas the best non-private methods get closer to 99.8% accuracy. A more challenging task is image classification on the CIFAR-10 dataset. Non-privately, methods are able to achieve 99.7% accuracy, while with $(3, 10^{-5})$-DP, we are able to achieve only 69.3% accuracy [TB21]. Neither the privacy guarantee or the accuracy guarantee are very compelling at the moment. There is clearly a lot of work to be done to improve the state of differentially private machine learning.

While DPSGD acts as a drop in replacement for SGD, there are many qualitative differences in the differentially private setting. We discuss some of them here.

One common complaint is that DPSGD is much slower than traditional SGD. The reason is that DPSGD requires one to clip each individual gradient in order to limit the sensitivity. Most modern machine learning frameworks are not built for this procedure, having methods which are optimized to use the GPU to compute the gradient over an entire mini-batch at once in parallel. The require-

ment of per-example gradients diverges from this standard, and the naive method of computing them would necessitate processing all points sequentially, thus losing all speedup granted by parallel processing on GPUs. As such, alternative algorithms for obtaining per-example gradients have been proposed [Goo15, RMT19], and sometimes alternative frameworks may be more efficient due to their low-level features, notably the recently-introduced framework JAX [SVK20].

As mentioned before, the ReLU is the most popular activation function in non-private machine learning, due to several convenient properties such as the ability to avoid "vanishing gradients" (an issue we will not get into here). However, in the differentially private setting, it appears that the tanh function (considered obsolete in the non-private setting) yields significant performance improvements [PTS+20]. This is one example showing that there can be benefits associated with modifying a network's architecture for the differentially private setting. On a similar note, non-private neural networks have grown exceptionally large, with the number of parameters growing into the hundreds of billions. This is because the size of a model is associated with higher "capacity," meaning that it can learn more functions. However, DPSGD requires the addition of Gaussian noise of magnitude proportional to the square root of the number of parameters. Thus, if we tried to run DPSGD on such a large model, our noise would drown out all signal. One must carefully choose the network architecture with this in mind – too small and the network wouldn't be able to represent the function, and too large and the noise would be overwhelming.

Hyperparameter tuning is a common challenge in machine learning tasks, and even more are introduced in the differentially private setting. For instance, how does one choose the learning rate, the lot size, the clipping norm, or the number of epochs? The canonical way to do this (non-privately) is to run a number of analyses on the training data with various hyperparameter settings, and choose the one which performs best on a validation set. Doing this in the differentially private setting would incur a cost in our privacy budget with every run, a cost which is currently omitted in most DP machine learning papers. This can be seen as pushing the methods to their limits, though they do not correspond to true privacy guarantees. Some methods have been proposed for hyperparameter optimization in a differentially private manner [LT19]. Another approach is to use public (non-sensitive) data that may have come from the same distribution as the private dataset. One can thus perform hyperparameter tuning on the public data, which will hopefully be suitable for the private data. An example of this is presented in [ACG+16], in which they treat the large CIFAR-100 dataset as public, and use this to train a neural network. They then freeze the majority of parameters in the model, and train the remainder privately on the sensitive CIFAR-10 dataset, achieving much better accuracy on the test set than without the CIFAR-100 training.

References

[ABC+20] Ahmet Aktay, Shailesh Bavadekar, Gwen Cossoul, John Davis, Damien Desfontaines, Alex Fabrikant, Evgeniy Gabrilovich, Krishna Gadepalli, Bryant Gipson, Miguel Guevara, Chaitanya Kamath, Mansi Kansal, Ali Lange, Chinmoy Mandayam, Andrew Oplinger, Christopher Pluntke, Thomas Roessler, Arran Schlosberg, Tomer Shekel, Swapnil Vispute, Mia Vu, Gregory Wellenius, Brian Williams, and Royce J. Wilson. Google covid-19 community mobility reports: Anonymization process description (version 1.0). *arXiv preprint arXiv:2004.04145*, 2020.

[ACG+16] 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 Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. ACM.

[BNO08] Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *Proceedings of the 28th Annual International Cryptology Conference*, CRYPTO '08, pages 451–468, Berlin, Heidelberg, 2008. Springer.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '14, pages 464–473, Washington, DC, USA, 2014. IEEE Computer Society.

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

[Dif17] Differential Privacy Team, Apple. Learning with privacy at scale. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf, December 2017.

[DKM+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '06, pages 486–503, Berlin, Heidelberg, 2006. Springer.

[DKY17] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems 30*, NIPS '17, pages 3571–3580. Curran Associates, Inc., 2017.

[DLS+17] Aref N. Dajani, Amy D. Lauger, Phyllis E. Singer, Daniel Kifer, Jerome P. Reiter, Ashwin Machanavajjhala, Simson L. Garfinkel, Scot A. Dahl, Matthew Graham, Vishesh Karwa, Hang Kim, Philip Lelerc, Ian M. Schmutte, William N. Sexton, Lars Vilhuber, and John M. Abowd. The modernization of statistical disclosure limitation at the U.S. census bureau, 2017. Presented at the September 2017 meeting of the Census Scientific Advisory Committee.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, pages 265–284, Berlin, Heidelberg, 2006. Springer.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, pages 202–210, New York, NY, USA, 2003. ACM.

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[EPK14] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM Conference on Computer and Communications Security*, CCS '14, pages 1054–1067, New York, NY, USA, 2014. ACM.

[Goo15] Ian Goodfellow. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*, 2015.

[LT19] Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM Symposium on the Theory of Computing*, STOC '19, pages 298–309, New York, NY, USA, 2019. ACM.

[McS16] Frank McSherry. Statistical inference considered harmful. https://github.com/frankmcsherry/blog/blob/master/posts/2016-06-14.md, June 2016.

[Mir17] Ilya Mironov. Rényi differential privacy. In *Proceedings of the 30th IEEE Computer Security Foundations Symposium*, CSF '17, pages 263–275, Washington, DC, USA, 2017. IEEE Computer Society.

[MTZ19] Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

[NS08] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 29th IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[PTS+20] Nicolas Papernot, Abhradeep Thakurta, Shuang Song, Steve Chien, and Úlfar Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. *arXiv preprint arXiv:2007.14191*, 2020.

[RMT19] Gaspar Rochette, Andre Manoel, and Eric W Tramel. Efficient per-example gradient computations in convolutional neural networks. *arXiv preprint arXiv:1912.06015*, 2019.

[RRUV16] Ryan M. Rogers, Aaron Roth, Jonathan Ullman, and Salil Vadhan. Privacy odometers and filters: Pay-as-you-go composition. In *Advances in Neural Information Processing Systems 29*, NIPS '16, pages 1921–1929. Curran Associates, Inc., 2016.

[SCS13] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing*, GlobalSIP '13, pages 245–248, Washington, DC, USA, 2013. IEEE Computer Society.

[SU20a] Thomas Steinke and Jonathan Ullman. The pitfalls of average-case differential privacy. https://differentialprivacy.org/average-case-dp/, July 2020.

[SU20b] Thomas Steinke and Jonathan Ullman. Why privacy needs composition. https://differentialprivacy.org/privacy-composition/, August 2020.

[SVK20] Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. Enabling fast differentially private sgd via just-in-time compilation and vectorization. *arXiv preprint arXiv:2010.09063*, 2020.

[TB21] Florian Tramèr and Dan Boneh. Differentially private learning needs better features (or much more data). In *Proceedings of the 9th International Conference on Learning Representations*, ICLR '21, 2021.

[Vad17] Salil Vadhan. The complexity of differential privacy. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, chapter 7, pages 347–450. Springer International Publishing AG, Cham, Switzerland, 2017.

[WBK19] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, AISTATS '19, pages 1226–1235. JMLR, Inc., 2019.

[WM10] Oliver Williams and Frank McSherry. Probabilistic inference and differential privacy. In *Advances in Neural Information Processing Systems 23*, NIPS '10, pages 2451–2459. Curran Associates, Inc., 2010.