Tempered Sigmoid Activations for Deep Learning with Differential Privacy

Nicolas Papernot[1], **Abhradeep Thakurta**[1], **Shuang Song**[1], **Steve Chien**[1], **Úlfar Erlingsson**[2*]
[1]Google Brain, [2]Apple
{papernot, athakurta, shuangsong, schien}@google.com, ulfar@apple.com

Abstract

Because learning sometimes involves sensitive data, machine learning algorithms have been extended to offer differential privacy for training data. In practice, this has been mostly an afterthought, with privacy-preserving models obtained by re-running training with a different optimizer, but using the model architectures that already performed well in a non-privacy-preserving setting. This approach leads to less than ideal privacy/utility tradeoffs, as we show here. To improve these tradeoffs, prior work introduces variants of differential privacy that weaken the privacy guarantee proved to increase model utility. We show this is not necessary and instead propose that utility be improved by choosing activation functions designed explicitly for privacy-preserving training.

A crucial operation in differentially private SGD is gradient clipping, which along with modifying the optimization path (at times resulting in not-optimizing a single objective function), may also introduce both significant bias and variance to the learning process. We empirically identify exploding gradients arising from ReLU may be one of the main sources of this. We demonstrate analytically and experimentally how a general family of bounded activation functions, the tempered sigmoids, consistently outperform the currently established choice: unbounded activation functions like ReLU. Using this paradigm, we achieve new state-of-the-art accuracy on MNIST, FashionMNIST, and CIFAR10 without any modification of the learning procedure fundamentals or differential privacy analysis. While the changes we make are simple in retrospect, the simplicity of our approach facilitates its implementation and adoption to meaningfully improve state-of-the-art machine learning while still providing strong guarantees in the original framework of differential privacy.

Introduction

Machine learning (ML) can be usefully applied to the analysis of sensitive data, e.g., in the domain of healthcare (Kononenko 2001). However, ML models may unintentionally reveal sensitive aspects of their training data, e.g., due to overfitting (Shokri et al. 2017; Song and Shmatikov 2019). To counter this, ML techniques that offer strong guarantees expressed in the framework of differential privacy (Dwork and Roth 2014) have been developed. A seminal example is the differentially private stochastic gradient descent, or DP-SGD, of Abadi et al. (Abadi et al. 2016). The technique is a generally-applicable modification of stochastic gradient descent. In addition to its rigorous privacy guarantees, it has been empirically shown to stop known attacks against the privacy of training data; a representative example being the leaking of secrets (Carlini et al. 2019).

Beyond privacy, training using DP-SGD offers advantages such as strong generalization and the promise of reusable holdouts (Dwork et al. 2015). Yet, its advantages have not been without cost: empirically, the test accuracy of differentially private ML is consistently lower than that of non-private learning unless the dataset is very large (McMahan et al. 2017). Such accuracy loss may sometimes be inevitable: for example, the task may involve heavy-tailed distributions and noise added by DP-SGD hinders visibility of examples in the tails (Feldman 2019; Bagdasaryan and Shmatikov 2019). However, this does not explain the accuracy loss of differentially private ML on benchmarks that are known to be relatively simple when learning without privacy: e.g., MNIST, FashionMNIST, and CIFAR10.

An important step in providing differential privacy guarantees for an algorithm is to assess its *sensitivity*. A learning algorithm's sensitivity characterizes how much an individual training point can, in the worst case, affect the learning algorithm's outputs (i.e., values of the model parameters). The ability to more strictly bound sensitivity leads to stronger privacy guarantees. To strictly bound the impact of any training example, DP-SGD makes two changes to every step of gradient-descent optimization: first, each example's gradient contribution is limited to a fixed bound – usually referred to as the clipping norm (in practice, by clipping all per-example gradients to a maximum ℓ_2 norm); second, random (Gaussian) noise of the scale of the clipping norm is added to each batch's combined gradient, before it is backpropagated to update model parameters. Together, these changes create a new, artificial noise floor at each step of gradient descent, such that the unique signal of any individual example is below this new noise floor; this allows differential privacy to be guaranteed for all training examples (Dwork and Roth 2014). However, the combined effect of clipping and noising also degrades the model's performance. Attempts to mitigate this have included new definitions of differential privacy (Dong, Roth, and Su 2019) or new learning algorithms (McMahan et al. 2018).

*Work done while at Google. The author is now at Apple.

Both require that the privacy analysis be conducted again.

Instead, this paper is the first to show that modifying the activation function can significantly improve the tradeoffs between privacy and utility. This does not involve any modification of the learning procedure or its analysis. We observe that DP-SGD leads to exploding model activations as a deep neural network's training progresses. This makes it difficult to control the training algorithm's sensitivity at a minimal impact to its correctness. Exploding activations cause unclipped gradient magnitudes to also increase, which in turn induces an information loss once the clipping operation is applied to bound gradient magnitudes. This exacerbates the negative impact of noise calibrated to the clipping bound, thus degrading the utility of each gradient step when learning with privacy. Indeed, the gradient clipping of DP-SGD does not bring the nice properties of gradient clipping commonly used to regularize deep learning (Zhang et al. 2019) because DP-SGD clips gradients at the granularity of individual training examples rather than at the level of a batch. According to (Song, Thakkar, and Thakurta 2020; Chen, Wu, and Hong 2020), gradient clipping can introduce bias that significantly affect model utility. Specifically, the sum of a batch of clipped per-example gradients does not necessarily point to the same direction as that of the original gradients, and thus DP-SGD no longer receives unbiased gradient estimators.

One way to address this limitation of differentially private learning is to modify the activation functions to prevent activations from exploding. Rather than using DP-SGD to train architectures that performed well without privacy, we propose to tailor the activation functions to the specificities of training with privacy. We hypothesize that activation functions need to be bounded when learning with DP-SGD. Therefore, for private learning, we propose to employ a general family of bounded activations: *tempered sigmoids*. We note that prior work has explored tempered losses as a means to provide robustness to noise during training (Amid et al. 2019). Because the family of tempered sigmoids can—in the limit—represent an approximation of ReLUs (Nair and Hinton 2010) on the subset of their domain that is exercised in training, we expect that our approach will perform no worse than current architectures. These architectures use ReLUs as the de facto choice of activation function.

Through both analysis and experiments, we validate the significantly superior performance of tempered sigmoids when training neural networks with DP-SGD. In our analysis, we relate the role of the temperature parameter in tempered sigmoids to the clipping operation of DP-SGD. Unlike prior work, which attempted to adapt the clipping norm to the gradients of each layer's parameters post hoc to training (McMahan et al. 2018), we find that tempered sigmoids preserve more of the signal contained in gradients of each layer because they rescale each layer's activations and better predispose the corresponding layer's gradients to clipping. We conclude that using tempered sigmoids is a better default activation function choice for private ML. In summary, our contributions facilitate DP-SGD learning as follows:

- We analytically show how tempered sigmoid activations control the gradient norm explicitly, and in turn support faster convergence in the settings of differentially private ML, by reducing the negative effects of clipping. Specifically for binary logistic regression, we formally quantify the effect of temperature (in tempered sigmoid) in controlling the bias introduced by clipping.
- To demonstrate empirically the superior performance of tempered sigmoids, we show how using tempered sigmoids instead of ReLU activations significantly improves a model's private-learning suitability and achievable privacy/accuracy tradeoffs.
- We advance the state-of-the-art of deep learning with differential privacy for MNIST, FashionMNIST, and CIFAR10. On these datasets, we find For fixed privacy guarantees $\varepsilon < 3$, we achieve 98.1% test accuracy (instead of 96.1%) on MNIST, 86.1% (instead of 81.9%) on FashionMNIST, and 66.2% (instead of 61.6% on CIFAR10.

While our paper focuses on the role of activation functions, we demonstrate for the first time the benefits of privacy by design: that is, how models should be architected for private learning rather than directly adopted from non-private learning. The changes we make are simple in retrospect, but we argue this simplicity is a strength given the significant utility improvements they provide compared to the alternative: the development of theoretical tools—such as new variants of the differential privacy definition that are better suited for deep learning (Bu et al. 2019)—is often a lengthy process and mandates that a privacy analysis be performed again.

Training-data Memorization, Differential Privacy, and DP-SGD

Machine learning models easily memorize sensitive, personal, or private data that was used in their training, and models may in practice disclose this data—as demonstrated by membership inference attacks (Shokri et al. 2017) and secret extraction results (Song and Shmatikov 2019; Carlini et al. 2019).

To reason about the privacy guarantees of algorithms such as training by stochastic gradient descent, differential privacy has become the established gold standard (Dwork and Roth 2014). Informally, an algorithm is differentially private if it always produces effectively the same output (in a mathematically precise sense), when applied to two input datasets that differ by only one record. Formally, a learning algorithm A that trains models from the set S is (ε, δ)-differentially-private, if the following holds for all training datasets D and D' that differ by exactly one record:

$$\Pr[A(D) \in S] \leq e^{\varepsilon} \Pr[A(D') \in S] + \delta \qquad (1)$$

Here, ε gives the formal privacy guarantee, by placing a strong upper bound on any privacy loss, even in the worst possible case. A lower ε indicates a stronger privacy guarantee or a tighter upper bound. The factor δ allows for some probability that the property may not hold (in practice, this δ is required to be very small, e.g., in inverse proportion to the dataset size).

A very attractive property of differential-privacy guarantees is that they hold true for all attackers—whatever they

are probing and whatever their prior knowledge—and that they remain true under various forms of composition. In particular, the output of a differentially-private algorithm can be arbitrarily post processed, without any weakening of the guarantees. Also, if sensitive training data contains multiple examples from the same person (or, more generally, the same sensitive group), ε-differentially-private training on this data will result in model with a $k\varepsilon$-differential-privacy guarantee for each person, as long as at most k training-data records are present per person.

Abadi et al. (2016) introduced DP-SGD as a method for training deep neural networks with differential-privacy guarantees that was able to achieve better privacy and utility than previous efforts (Chaudhuri, Monteleoni, and Sarwate 2011; Song, Chaudhuri, and Sarwate 2013; Bassily, Smith, and Thakurta 2014). DP-SGD bounds the sensitivity of the learning process to each individual training example by computing per-example gradients $\{g_i\}_{i\in 0..n-1}$ with respect to the loss, for the n model parameters $\{\theta_i\}_{i\in 0..n-1}$, and clipping each per-example gradient to a maximum fixed ℓ_2 norm C: Subsequently, to the average of these per-example gradients, DP-SGD adds (Gaussian) noise whose standard deviation σ is proportional to this sensitivity. In this work, we use the canonical implementation of DP-SGD and its associated analysis from the TensorFlow Privacy library (Google 2019).

Approach

When training a model with differential privacy, gradients computed during SGD are computed individually for each example (i.e., the gradient computation is not averaged across all samples contained in a minibatch). The gradient g_i for each model parameter θ_i is then clipped such that the total l_2 norm of the gradient across all parameters is bounded by C:

$$g_i \leftarrow g_i \cdot \min\left(1, \frac{C}{\sqrt{\sum_{i=0}^{n-1} g_i^2}}\right) \tag{2}$$

Because this operation is performed on per-example gradients, this allows DP-SGD to control the sensitivity of learning to individual training examples. However, this clipping operation will lead to information loss when some of the signal contained in gradients is discarded because the magnitude of gradients is too large. One way to reduce the magnitude (or at least control it), is to prevent the model's activations from exploding. This is one of the reasons why common design choices for the architecture of modern deep neural networks make it difficult to optimize model parameters with DP-SGD: prominent activation functions like the REctified Linear Unit (ReLU) are unbounded.

We hypothesize that replacing ReLUs with a bounded activation function prevents activations from exploding and thus keeps the magnitude of gradients to a more reasonable value. This in turn implies that, given a fixed level of privacy guarantee, the clipping operation applied by DP-SGD will discard less signal from gradient updates—eventually resulting in higher performance at test time.

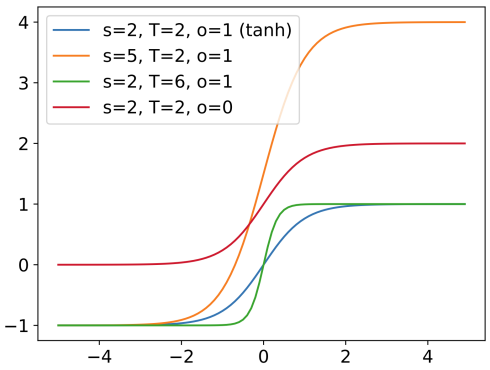

Figure 1: Tempered sigmoids: we plot examples for representative values of the scale s, inverse temperature T, and offset o. The blue line corresponds to the triplet $(s = 2, T = 2, o = 1)$ where the tempered sigmoid is exactly a tanh.

Tempered sigmoids. Based on this intuition, we propose replacing the unbounded activations typically used in deep neural networks with a general family of bounded activations: the tempered sigmoids. We note that an idea that is conceptually close to ours, the use of tempered losses, was recently found to provide robustness to noise during training (Amid et al. 2019).[1] Tempered sigmoids are the family of functions that take the form of:

$$\phi_{s,T,o} : x \mapsto \frac{s}{1 + e^{-T \cdot x}} - o \tag{3}$$

where s controls the scale of the activation, T is the inverse temperature, and o is the offset. By decreasing the value of s, we reduce the magnitude of a neuron's activation. Complementary to this, the inverse temperature rescales a neuron's weighted inputs. We note that setting $s = 2$, $T = 2$, and $o = 1$ in particular yields the tanh function exactly, i.e., we have $\phi_{2,2,1} = \texttt{tanh}$.

Controlling the gradient norm with tempered sigmoids. One of the main issues in practice with DP-SGD is tuning the value of the clipping parameter:

- If C is set too low, then clipping introduces bias by changing the underlying objective optimized during learning.
- Instead if the clipping parameter C is set too high, clipping increases variance by forcing DP-SGD to add too much noise. Indeed, recall that DP-SGD adds Gaussian noise with variance σ^2 to the average of (clipped) per-example gradients. This noise is scaled to the clipping norm such that $\sigma^2 = M^2C^2$ where M is a hyperparameter called the noise multiplier. Thus, large clipping norms lead to noise with large variance being added to the average gradient before it is applied to update model parameters.

It turns out that the temperature parameter T in Equation (3) can be used as a knob to control the norm of the gradient of the loss function, and with an appropriate choice avoids these

[1] We experimented with the tempered loss of (Amid et al. 2019) but did not find any improvements for DP-SGD training.

two issues of clipping. In the following, we formalize the relationship between our tempered sigmoids and the clipping of DP-SGD in the context of the binary logistic loss and its multiclass counterpart.

Consider the tempered logistic loss: $\ell(\theta; z, y) = \ln(1 + \exp(-y \cdot T \cdot \langle z, \theta \rangle))$, where $z, \theta \in \mathbb{R}^d$, $y \in \{-1, +1\}$, and $T \in \mathbb{R}$ is the *inverse* temperature. Notice that the above expression is an instantiation of $\ln(1/\phi_{s,T,o})$, where $s = 1$ and $x = y \cdot \langle z, \theta \rangle$. Now, if we take the gradient of ℓ w.r.t θ, we have the following for the ℓ_2-norm of the gradient.

$$\|\nabla_\theta \ell\|_2 = \left\| \frac{-T \cdot y \cdot z}{1 + \exp(T \cdot y \cdot \langle z, \theta \rangle)} \right\|_2 \leq |T| \cdot \|z\|_2 \quad (4)$$

We observe the following two things from (4): i) Controlling T directly controls the norm of the gradient of ℓ, and hence controls clipping norm in general (when using tempered sigmoid as an activation), ii) Specifically, for logistic loss, T can be thought of as linear scaling of the feature vector z, when the point (z, y) is "grossly misclassified". These observations suggest that one can use the inverse temperature to control the norm of the gradient, and may not ever cross the "clipping threshold" in DP-SGD.

One can extend this observations to multiclass logistic loss $\ell(\theta; z, y) = \ln\left(\frac{\exp(T \cdot \langle z, \theta y \rangle)}{\sum_{j \in 1..k} \exp(T \cdot \langle z, \theta_j \rangle)}\right)$ where y is now a one-hot label vector and $\theta \in \mathbb{R}^{d \times k}$ for a problem with k classes. The partial gradient of ℓ w.r.t. θ_m, for any class $m \in 1..k$, becomes:

$$\partial_{\theta_m} \ell = \left(\mathbb{1}_{m=y} - \frac{\exp(T \cdot \langle z, \theta_m \rangle)}{\sum_{j \in 1..k} \exp(T \cdot \langle z, \theta_j \rangle)} \right) \cdot T \cdot z \quad (5)$$

Because the norm of the expression in parenthesis is smaller than 1, we thus have that $\|\partial_{\theta_m} \ell\|_2 \leq |T| \cdot \|x\|_2$. From this we derive that the norm of the gradient $\nabla_\theta \ell = [\partial_{\theta_1} \ldots \partial_{\theta_k}]$ would correspondingly be bound by $\sqrt{k} \cdot \|\partial_{\theta_m} \ell\|_2 \leq \sqrt{k} \cdot |T| \cdot \|z\|_2$, where k is the number of classes.

Controlling gradient norm helps control bias. In this section we will formally quantify how temperature T can control the bias introduced for binary logistic regression. The analysis will be along line of (Song, Thakkar, and Thakurta 2020). The formal quantification, however, cannot be extended beyond binary logistic regression, since (Song, Thakkar, and Thakurta 2020) already shows that even for simple problem like multi-class logistic regression, clipped gradients may not conform to the gradient field of a single objective function. Hence, any formal quantification of the bias in terms of excess empirical risk is not possible in that case.

We define the tempered logistic loss on a data set $D = \{(z_i, y_i)\}_{i=1}^n$, $\|z_i\|_2 \leq 1$, $y_i \in \{-1, +1\}$ as $\mathcal{L}_T(\theta; D) = \frac{1}{n} \sum_{i=1}^{n} \ln(1 + \exp(-T \cdot y_i \cdot \langle z_i, \theta \rangle))$. Let $\mathcal{D}$ be the domain of such data sets. For the purpose of exposition, we will only consider full-gradient descent (GD) (Song, Thakkar, and Thakurta 2020) where the state updates are made on the full-gradient of $\mathcal{L}_T(\theta; D)$, rather than a a stochastic gradient computed on a mini-batch of D.

Theorem 1. *Consider the objective function $\mathcal{L}_T(\theta, D)$ for logistic regression as defined above, and a domain of models $\Theta \subseteq \mathbb{R}^d$. Let $\theta^{\mathsf{clipped}} \in \Theta$ be the output of GD on $\mathcal{L}_T(\theta, D)$ with clipping norm C. For any $C \in \mathbb{R}^+$ and for all $D \in \mathcal{D}$ (where $\mathcal{D}$ is the domain of data sets defined above), the following is true for appropriate choice of the learning rate and running GD upto convergence.*

$$\mathcal{L}_T(\theta^{\mathsf{clipped}}; D) - \min_{\theta \in \Theta} \mathcal{L}_T(\theta; D) = O\left(\max\left\{(T - C) \cdot \|\Theta\|_2, 0\right\}\right).$$

where $\|\Theta\|_2 = \sup_{\theta, \theta' \in \Theta} \|\theta - \theta'\|_2$ is the diameter of set Θ.

Theorem 1 essentially shows that as we reduce the temperature, the bias in terms of the excess empirical risk goes down, for any clipping norm C. The proof can be found in the supplemental material.

Experimental Setup

We use three common benchmarks for differentially private ML: MNIST (LeCun, Cortes, and Burges 1998), FashionMNIST (Xiao, Rasul, and Vollgraf 2017), and CIFAR10 (Krizhevsky, Hinton et al. 2009). While the three datasets are considered as "solved" in the computer vision community, achieving high utility with strong privacy guarantees remains difficult on all three datasets (Abadi et al. 2016; Google 2019). Concretely, the state-of-the-art for MNIST is a test accuracy of 96.6% given an $(\varepsilon, \delta) = (2.93, 10^{-5})$ differential privacy guarantee. With stronger guarantees, the accuracy continues to degrade. In the same privacy-preserving settings, prior approaches achieve a test accuracy of 81.9% on FashionMNIST. For CIFAR10, a test accuracy of 61.6% can be achieved given an $(\varepsilon, \delta) = (7.53, 10^{-5})$ differential privacy guarantee. To ensure a fair evaluation, we fix the configuration of DP-SGD to achieve these guarantees in all of our experiments. We note that prior work already explored tuning the configuration of DP-SGD, e.g., the clipping norm, and concluded that this was unable to improve the privacy/utility tradeoff (McMahan et al. 2018).

All of our experiments are performed with the JAX framework in Python, on a machine equipped with a 5th generation Intel Xeon processor and NVIDIA V100 GPU acceleration. For both MNIST and FashionMNIST, we use a convolutional neural network whose architecture is described in Table 1. For CIFAR10, we use the deeper model in Table 2. The choice of architectures is motivated by prior work which showed that training larger architectures is detrimental to generalization when learning with privacy (Bassily, Smith, and Thakurta 2014). This can be explained in two ways. Given a fixed privacy guarantee, increasing the number of parameters increases (a) how much each parameter needs to be clipped relatively and (b) how much noise needs to be added, with the norm of noise increasing as a function of the square root of the number of parameters. For completeness and to ensure a fair evaluation, we confirm that finetuning the number of parameters does not affect our results in Table 3.

When we train these architectures with ReLU activations for both the convolution and fully-connected layers, we are

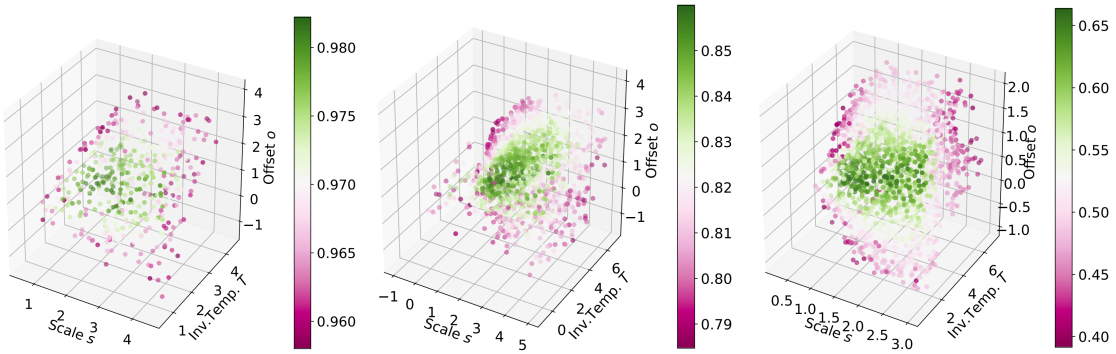

Figure 2: Test accuracy of architectures with tempered sigmoids $\phi_{s,T,o}$ as a function the scale s, inverse temperature T, and offset o. Results are plotted for MNIST, FashionMNIST, and CIFAR10 from left to right. All models are trained with DP-SGD to achieve a fixed privacy guarantee (i.e., constant ε).

Layer	Parameters
Convolution	16 filters of 8x8, strides 2
Max-Pooling	2x2
Convolution	32 filters of 4x4, strides 2
Max-Pooling	2x2
Fully connected	32 units
Softmax	10 units

Table 1: Convolutional model architecture.

Layer	Parameters
Convolution $\times 2$	32 filters of 3×3, strides 1
Avg-Pooling	2×2, stride 2
Convolution $\times 2$	64 filters of 3×3, strides 1
Avg-Pooling	2×2, stride 2
Convolution $\times 2$	128 filters of 3×3, strides 1
Avg-Pooling	2×2, stride 2
Convolution	256 filters of 3×3, strides 1
Convolution	10 filters of 3×3, strides 1
Averaging	over spatial dimensions

Table 2: CIFAR10 model architecture.

able to exactly reproduce the previous state-of-the-art results mentioned above for MNIST, FashionMNIST, and CIFAR10. To experiment with the tempered sigmoid proposed in Section , we implement it in JAX and use it in lieu of the ReLU in the architecture from Table 1 and Table 2. Our code will be open-sourced through a pull request to the JAX repository on GitHub, and we include the code snippet for the tempered sigmoid activation below—to demonstrate the practicality of implementing the change we propose in neural architectures.

```
from jax.scipy.special import expit

def temp_sigmoid(x, scale=2., inverse_temp=2., offset=1., axis=-1):
    return scale * expit(inverse_temp * x) - offset

def elementwise(fun, **fun_kwargs):
    """Layer that applies a scalar function elementwise."""
    init_fun = lambda rng, input_shape: (input_shape, ())
    apply_fun = lambda params, inp, **kwargs: fun(inp, **fun_kwargs)
    return init_fun, apply_fun

TemperedSigmoid = elementwise(temp_sigmoid, axis=-1)
```

Evaluating the Family of Tempered Activation Functions

We now explore the family of tempered sigmoids empirically. As hypothesized in the introduction, tempered sigmoids not only do no worse than ReLU, there exists in fact many functions in the tempered sigmoid family which outperform ReLU given a fixed privacy guarantee (i.e., constant ε). We then measure how tempered sigmoids help control the norm of activation norms during training, thus preventing DP-SGD from resulting in exploding gradients detrimental to learning, as previously exposed analytically in our Approach.

Improved Privacy-Utility Tradeoffs

For each of the three datasets considered, we use DP-SGD to train a pair of models. The first model uses ReLU whereas the second model uses a tempered sigmoid $\phi_{s,T,o}$ as the activation for all of its hidden layers (i.e., both convolutional and fully-connected layers). The models are based off the architecture of Table 1 for MNIST and FashionMNIST, or Table 2 for CIFAR10. All other architectural elements are kept identical. In our experiments, we subsequently fine-tuned architectural aspects (i.e., model capacity) as well as the choice of optimizer and its associated hyperparameters, separately for the activation function in each setting (ReLU and tempered sigmoid), to avoid favoring any one choice.

Recall from Section that tempered sigmoids $\phi_{s,T,o}$ are bounded activations that are parameterized such that their inputs and output can be rescaled—through the inverse temperature T and scale s parameters respectively—and their output recentered with the offset o. Tempered sigmoids help control the norm of the gradient of the loss function, and in turn mitigate some of the negative effects from clipping. In Figure 2, we visualize the influence of the scale s, inverse temperature T, and offset o on the test performance of models trained with DP-SGD and tempered sigmoids $\phi_{s,T,o}$.

Tempered sigmoids significantly outperform models trained with ReLU on all three datasets. On MNIST, the best tempered sigmoid achieves 98.1% test accuracy whereas the baseline ReLU model trained to provide identical pri-

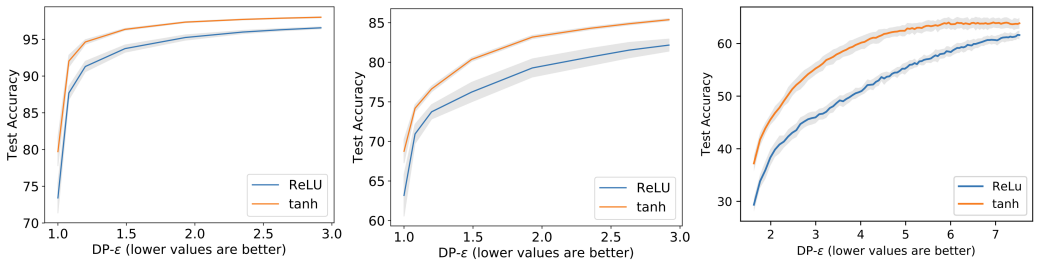

Figure 3: Test accuracy as a function of the privacy loss when training a pair of models with DP-SGD on MNIST, FashionMNIST, and CIFAR10 (left to right). The only difference between the two models is the activation function for their hidden layer: ReLU or `tanh`. All other elements of the architecture (number, type, and dimension of layers) and the training algorithm (optimizer, learning rate, number of microbatches, clipping norm, and noise multiplier) are identical. Results averaged over 10 runs.

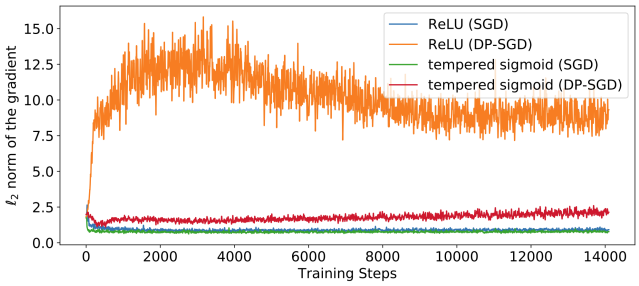

Figure 4: ℓ_2 norm of pre-clipped gradients, for models trained with SGD or DP-SGD and ReLU or tempered sigmoids. A triangular moving average of window 3 is applied for clarity.

vacy guarantees ($\varepsilon = 2.93$) achieved 96.6% accuracy. This contributes to bridging the gap between privacy-preserving learning and non-private learning, which results in a test accuracy of 99.0% for this architecture with both ReLU and `tanh`. On FashionMNIST, we achieve a best performing model of 86.0% with tempered sigmoids in comparison with 81.9% with ReLUs. A non-private model achieves 89.3% with `tanh` and 89.4% with ReLUs. On CIFAR10, the best tempered sigmoid architectures achieve 66.0% test accuracy whereas the ReLU variant obtained 61.6% under the same privacy guarantees and the non-private baseline 76.6%.

From Figure 2, it appears clearly that a subset of tempered sigmoids performs best when learning with DP-SGD on the three datasets we considered. This demonstrate that the family of tempered sigmoids as a whole, rather than one of its members only, benefits learning with privacy—as we analytically derived previously. These members of the tempered sigmoid family form a large cluster of points (s, T, o) which result in models with significantly higher test accuracy. These points are colored in dark green. For each dataset, we compute the average value of the 10% best-performing triplets (s, T, o). On MNIST, the average triplet obtained is $(s, T, o) = (1.97, 2.27, 1.15)$, on FashionMNIST $(s, T, o) = (2.27, 2.61, 1.28)$, and on CIFAR10 $(s, T, o) = (1.58, 3.00, 0.71)$. We come back later to this observation and its beneficial implications for the practicality of DP-SGD with tempered sigmoids.

Impact of tempered sigmoids on gradient norms. To explain why a simple change of activations has such a large positive impact on model accuracy, we conjectured that the bounded nature of the tempered sigmoid activations prevents the norm of gradients from exploding during training.

We monitored the ℓ_2 norm of the model's gradients, computed before the clipping operation is applied, for our MNIST model, while it is being trained in four scenarios: (a) without privacy using vanilla SGD and ReLU activations, (b) without privacy using vanilla SGD and tempered sigmoid activations, (c) with ReLU activations and DP-SGD, and (d) with a tempered sigmoid activation and DP-SGD. For (b,d), we set the hyperparameter triplet in the tempered sigmoid accordingly to the average best-performing triplet noted above, i.e., a tanh. The norm of gradients are visualized in Figure 4. As conjectured previously, the gradients of our ReLU model explode by a factor of more than 5 when training with privacy compared to without privacy. Switching to a tempered sigmoid brings down the norms of DP-SGD gradients back to levels comparable with the gradients of our non-private ReLU network. This helps us to learn with privacy—because it reduces the negative effects of clipping and noising large gradients. By predisposing gradients to the operations performed by DP-SGD, less information is lost: the norm of unclipped gradients is closer to the clipping norm, and is also more adequate to the noise scale. We observe the same qualitative differences on other datasets, but do not repeat the plots due to space constraints.[2]

Tempered Sigmoids Do Not Increase the Computational Cost of Hyperparameter Tuning

At first glance, tempered sigmoids introduce three new hyperparameters in the training procedure. It is thus interesting to note how the average values taken by the best-performing triplets in Figure 2 happen to be close to the triplet setting

[2]If we repeat the experiment, but instead measure the norm of activations, we draw qualitatively consistent conclusions: training with tempered sigmoids prevents activations from exploding with DP-SGD. This was explained in our analytical treatment of the role of temperature in our Approach. This in turn explains in part why the norm of gradients, a global object that includes among other things activations, also does not explode in DP-SGD with tempered sigmoids. This is found in the supplementary material.

Dataset	Technique	Acc.	ε	δ
MNIST	SGD w/ ReLU (not private)	99.0%	∞	0
	DP-SGD w/ ReLU	96.6%	2.93	10^{-5}
	DP-SGD w/ tempered sigmoid [ours]	**98.1%**	**2.93**	10^{-5}
FashionMNIST	SGD w/ ReLU (not private)	89.4%	∞	0
	DP-SGD w/ ReLU	81.9%	2.7	10^{-5}
	DP-SGD w/ tempered sigmoid [ours]	**86.1%**	**2.7**	10^{-5}
CIFAR10	SGD w/ ReLU (not private)	76.6%	∞	0
	DP-SGD w/ ReLU	61.6%	7.53	10^{-5}
	DP-SGD w/ tempered sigmoid [ours]	**66.2%**	**7.53**	10^{-5}

Table 3: Summary of results comparing ReLU to tempered sigmoids in their respective best performing setting (i.e., each row is the result of a hyperparameter search). Values of (s, T, o) are set according to the average best-performing triplet, yielding a tanh.

$(s, T, o) = (2, 2, 1)$ for MNIST and FashionMNIST—and to a lesser extent for CIFAR10. Recall that this setting corresponds exactly to the `tanh` function. While this observation may not hold for other datasets, we seek to understand, for the datasets we considered, whether tanh is able to sustain the significant improvements of tempered sigmoids over ReLU without having to fine-tune the three hyperparameters introduced by tempered sigmoids.

Outperforming ReLU without additional hyperparameters using tanh. Figure 3 visualizes the privacy-utility Pareto curve (Avent et al. 2019) of the ReLU and tanh models trained with DP-SGD for all three datasets. Rather than plotting the test accuracy as a function of the number of steps, we plot it as a function of the privacy loss ε (but the privacy loss is a monotonically increasing function of the number of steps). The tanh models outperform their ReLU counterparts consistently regardless of the privacy loss ε expended: the test accuracy of the tanh model is 98.0% on MNIST, 85.5% on FashionMNIST, and 63.84% on CIFAR10. The performance of tanh is in line with the best test accuracy observed across tempered sigmoids on Figure 2. Thus, we find that for these datasets positive results observed on the general family of tempered sigmoids can be reproduced with tanh alone, which is obtained by setting $(s, T, o) = (2, 2, 1)$. This means that replacing ReLU with a tanh will already give a significant improvement without additional hyperparameter tuning.

Fine-tuning the optimizer. To ensure that the comparison between ReLU and tempered sigmoids is fair, we now turn our attention to the training algorithm itself and verify that the superior behavior of tempered sigmoids holds after a thorough hyperparameter search.

To motivate why it is important to tailor algorithm and hyperparameter choices to the specificities of DP-SGD, Table 4 (in the supplementary material) shows how learning rates obtained through a hyperparameter search based on Batched Gaussian Process Bandits (Desautels, Krause, and Burdick 2014) vary across the non-DP and DP settings when training on FashionMNIST. On the contrary, the choice of optimizer (and in particular whether it is adaptive or not) does not influence results as much.

This begs the question of whether additional hyperparameter tuning is required to compare fairly ReLU and tempered sigmoids. Among the hyperparameters mentioned above, it is particularly important to fine-tune the learning rate to maximize performance given a fixed privacy budget. This is because the privacy budget limits the number of steps we can possibly take on the training set (as visualized on Figure 3).

Table 3 summarizes the results after performing a hyperparameter search over the number of filters k, learning rate, optimizer, batch size, and number of epochs. This is done separately for each of the three datasets considered in our experiments. We compare the non-private baseline and the DP-SGD with ReLU baseline to our DP-SGD approach with tempered sigmoids after all hyperparameters have been jointly finedtuned. For clarity of presentation, we report here results on a tempered sigmoid using the average best-performing triplet (s, T, o), i.e., a tanh. Even in their own individually-best setting, tempered sigmoids continue to consistently outperform ReLU with 98.1% test accuracy (instead of 96.6% for ReLU) on MNIST, 86.1% test accuracy (instead of 81.9% for ReLU) on FashionMNIST, 66.2% test accuracy (instead of 61.6% for ReLU) on CIFAR10. Thus, our results are not impacted and do not call for large-scale hyperparameter finue-tuning. This in turn means that our findings may be implemented by practitioners with little overhead.

Conclusions

Rather than first train a non-private model and later attempt to make it private, we bypass non-private training altogether and directly incorporate specificities of private learning in the selection of activation functions. Selecting a tempered sigmoid renders the architecture more suitable for learning with differential privacy. We are able to improve substantially upon the state-of-the-art privacy/accuracy trade-offs on three benchmarks which remain challenging for deep learning with differential privacy: MNIST, FashionMNIST, and CIFAR10. Future work may continue to explore this avenue: model architectures need to be chosen explicitly for privacy-preserving training. In addition, choosing the parameters (s, T, o) to be shared across all layers was not a necessity. We found that layer-wise parameters did not improve results on our datasets, but it may be the case for different tasks: this is related to the layer-wise clipping of McMahan et al. (2018).

Broader Impact

Our work helps make privacy-preserving training more practical. Our analysis and experimental results help practitioners make better choices when design neural architectures for privacy-preserving deep learning. In particular, the conclusions from our paper can readily be applied in real-world machine learning pipelines. For this reason, we expect the broader impact of this work to be generally positive given the numerous applications of machine learning to sensitive datasets. This includes applications in domains like healthcare or language modeling.

We also observe that the improved utility of tempered sigmoids does not negatively impact the fairness of a classifier, where fairness is defined as equal performance across classes (Bagdasaryan and Shmatikov 2019). In a preliminary experiment, we indeed compared the per-class accuracy of models trained with tanh and ReLU activations. The standard deviation of per-class accuracies is 0.27 for ReLU models compared to 0.16 for tanh, for results averaged over 10 runs.

Appendix

Proof of Theorem 1

Proof of Theorem 1. By Lemma 3.1 in (Song, Thakkar, and Thakurta 2020) we know that any clipped gradient descent optimizes a well-defined objective function (call it $\mathcal{L}_T^{\mathsf{clipped}}(\theta; D)$). In particular, $\mathcal{L}_T^{\mathsf{clipped}}$ is formed by replacing the individual loss functions $\ell(\theta; z, y) = \ln(1 + \exp(-T \cdot y\langle z, \theta\rangle))$ in $\mathcal{L}_T$ with a function $\hat{\ell}$ which matches ℓ when $\|\nabla_\theta \ell\|_2 \leq C$, and has gradient $\left(\frac{\nabla_\theta \ell}{\|\nabla_\theta \ell\|_2} \cdot C\right)$ anywhere else. The differentiability and convexity of $\hat{\ell}$ follows from Lemma 3.1 in (Song, Thakkar, and Thakurta 2020).

Consider $\hat{\ell}(\theta; z, y) - \ell(\theta; z, y)$. Its gradient is zero when $\|\nabla \ell\|_2 \leq C$ and is $\left(\frac{C}{\|\nabla_\theta \ell\|_2} - 1\right) \cdot \nabla_\theta \ell$ when $\|\nabla \ell\|_2 > C$. The norm of the gradient is upper bounded by $\max\{T - C, 0\}$, and thus $\forall \theta \in \Theta$, $|\hat{\ell}(\theta; z, y) - \ell(\theta; z, y)| \leq \max\{T - C, 0\} \cdot \|\Theta\|_2$. Define $\Delta := \max\{T - C, 0\} \cdot \|\Theta\|_2$. For any $\theta \in \Theta$, we have $|\mathcal{L}_T^{\mathsf{clipped}}(\theta; D) - \mathcal{L}_T(\theta; D)| \leq \frac{1}{n} \sum_{i=1}^n \left|\hat{\ell}(\theta; z_i, y_i) - \ell(\theta; z_i, y_i)\right| \leq \Delta$.

By definition of gradient descent in the theorem statement, $\theta^{\mathsf{clipped}} = \arg\min_{\theta \in \Theta} \mathcal{L}_T^{\mathsf{clipped}}(\theta; D)$. Furthermore, let $\theta^* = \arg\min_{\theta \in \Theta} \mathcal{L}_T(\theta; D)$. Notice that $\theta^{\mathsf{clipped}}, \theta^* \in \Theta$.

Since $\theta^{\mathsf{clipped}}$ is the minimizer of $\mathcal{L}_T^{\mathsf{clipped}}$, we have $\mathcal{L}_T^{\mathsf{clipped}}(\theta^{\mathsf{clipped}}; D) - \mathcal{L}_T(\theta^*; D) \leq \mathcal{L}_T^{\mathsf{clipped}}(\theta^*; D) - \mathcal{L}_T(\theta^*; D) \leq \Delta$. Therefore, we have $\mathcal{L}_T(\theta^{\mathsf{clipped}}; D) - \mathcal{L}_T(\theta^*; D) = \left(\mathcal{L}_T(\theta^{\mathsf{clipped}}; D) - \mathcal{L}_T^{\mathsf{clipped}}(\theta^{\mathsf{clipped}}; D)\right) + \left(\mathcal{L}_T^{\mathsf{clipped}}(\theta^{\mathsf{clipped}}; D) - \mathcal{L}_T(\theta^*; D)\right) \leq 2\Delta$.

□

Additional Table and Figure

On the next page, we include additional results on the trade-off between accuracy and privacy involving different optimizers (see Table 4), as well as a detailed measurement of the norm of activations for models trained with different activation functions (see Figure 5).

		Non-private		Differentially-private	
Optimizer	Epochs	Learning Rate	Test Accuracy	Learning Rate	Test Accuracy
SGD	40	$1.07 \cdot 10^{-1}$	90.3%	$3.32 \cdot 10^{-1}$	86.1%
Adam	40	$1.06 \cdot 10^{-3}$	90.5%	$1.32 \cdot 10^{-3}$	86.0%

Table 4: Impact of learning rate on trade-off between accuracy and privacy. The privacy budget is fixed to $\varepsilon = 2.7$ for all rows. A hyperparameter search is then conducted to find the best learning rate to train the model with or without differential privacy on FashionMNIST.

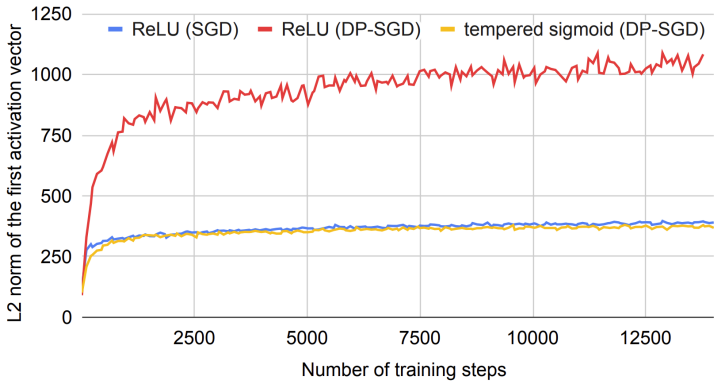

Figure 5: ℓ_2 norm of the first conv activations. Three scenarios are plotted: (a) the model is trained without privacy using plain SGD, (b) the model is trained with ReLU activations with DP-SGD, and (c) the model is trained with tempered sigmoid activations (here instantiated according to the average best-performing triplet, a tanh) with DP-SGD.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318. ACM.

Amid, E.; Warmuth, M. K.; Anil, R.; and Koren, T. 2019. Robust Bi-Tempered Logistic Loss Based on Bregman Divergences. In *Advances in Neural Information Processing Systems*, 14987–14996.

Avent, B.; Gonzalez, J.; Diethe, T.; Paleyes, A.; and Balle, B. 2019. Automatic Discovery of Privacy-Utility Pareto Fronts. *arXiv preprint arXiv:1905.10862* .

Bagdasaryan, E.; and Shmatikov, V. 2019. Differential Privacy Has Disparate Impact on Model Accuracy.

Bassily, R.; Smith, A.; and Thakurta, A. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, 464–473. IEEE.

Bu, Z.; Dong, J.; Long, Q.; and Su, W. J. 2019. Deep learning with Gaussian differential privacy. *arXiv preprint arXiv:1911.11607* .

Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; and Song, D. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security Symposium*.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12(Mar): 1069–1109.

Chen, X.; Wu, Z. S.; and Hong, M. 2020. Understanding Gradient Clipping in Private SGD: A Geometric Perspective. *arXiv preprint arXiv:2006.15429* .

Desautels, T.; Krause, A.; and Burdick, J. W. 2014. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research* 15: 3873–3923.

Dong, J.; Roth, A.; and Su, W. J. 2019. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383* .

Dwork, C.; Feldman, V.; Hardt, M.; Pitassi, T.; Reingold, O.; and Roth, A. 2015. The reusable holdout: Preserving validity in adaptive data analysis. *Science* 349(6248): 636–638.

Dwork, C.; and Roth, A. 2014. *The Algorithmic Foundations of Differential Privacy*. now.

Feldman, V. 2019. Does Learning Require Memorization? A Short Tale about a Long Tail. *arXiv preprint arXiv:1906.05271* .

Google. 2019. TensorFlow Privacy. https://github.com/tensorflow/privacy. Last accessed on September 1st, 2020.

Kononenko, I. 2001. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine* 23(1): 89–109.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

LeCun, Y.; Cortes, C.; and Burges, C. 1998. The mnist database of handwritten digits. *URL http://yann.lecun.com/exdb/mnist* Last accessed on September 1st, 2020.

McMahan, B.; Andrew, G.; Mironov, I.; Papernot, N.; Kairouz, P.; Chien, S.; and Erlingsson, Ú. 2018. A General Approach to Adding Differential Privacy to Iterative Training Procedures. *NeurIPS 2018 workshop on Privacy Preserving Machine Learning, Montreal, Canada* .

McMahan, H. B.; Ramage, D.; Talwar, K.; and Zhang, L. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* .

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE.

Song, C.; and Shmatikov, V. 2019. Auditing Data Provenance in Text-Generation Models. *arXiv preprint arXiv:1811.00513* .

Song, S.; Chaudhuri, K.; and Sarwate, A. D. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, 245–248. IEEE.

Song, S.; Thakkar, O.; and Thakurta, A. 2020. Characterizing private clipped gradient descent on convex generalized linear problems. *arXiv preprint arXiv:2006.06783* .

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

Zhang, J.; He, T.; Sra, S.; and Jadbabaie, A. 2019. Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity. In *International Conference on Learning Representations*.