Differentially Private Learning with Adaptive Clipping

Galen Andrew [1] **Om Thakkar** [1] **H. Brendan McMahan** [1] **Swaroop Ramaswamy** [1]

Abstract

We consider the problem of hyperparameter tuning in training neural networks with user-level differential privacy (DP). Existing approaches for DP training (e.g., DP Federated Averaging) involve bounding the contribution of each user's model update by *clipping* them to a fixed norm. However there is no good *a priori* setting of the clipping norm across tasks and learning settings: the update norm distribution depends on the model architecture and loss, the amount of data on each device, the client learning rate, and possibly various other parameters. In this work, we propose a method wherein instead of using a fixed clipping norm, one clips to a value at a specified quantile of the distribution of update norms, where the value at the quantile is itself estimated online, with differential privacy. Experiments demonstrate that adaptive clipping to the median update norm works well across a range of federated learning problems, eliminating the need to tune any clipping hyperparameter.

1. Introduction

Deep learning has become ubiquitous, with applications as diverse as image processing, natural language translation, and music generation (Szegedy et al., 2015; He et al., 2015; Vinyals et al., 2015; Briot et al., 2017). Deep models are able to perform so well in part due to their ability to make use of vast amounts of data for training. However, recent work has shown that it is possible to extract detailed information about individual training examples using only the parameters of a trained model (Fredrikson et al., 2015; Wu et al., 2016; Shokri et al., 2017; Carlini et al., 2018; Melis et al., 2018). When the training data potentially contains privacy-sensitive user information, it becomes imperative to use learning techniques that limit such memorization.

Differential privacy (DP) (Dwork et al., 2006a;b) is widely considered a gold standard for bounding and quantifying the privacy leakage of sensitive data when performing learning tasks. Intuitively, DP prevents an adversary from confidently making any conclusions about whether some users' data was used in training a model, even given access to arbitrary side information. The formal definition of DP depends on the notion of neighboring datasets: we will refer to a pair of datasets $D, D' \in \mathcal{D}$ as neighbors if D' can be obtained from D by adding or removing one element.

Definition 1.1 (Differential Privacy)**.** *A (randomized) algorithm $M : \mathcal{D} \to \mathcal{R}$ with input domain $\mathcal{D}$ and output range $\mathcal{R}$ is (ε, δ)-differentially private if for all pairs of neighboring datasets $D, D' \in \mathcal{D}$, and every measurable $S \subseteq \mathcal{R}$:*

$$\Pr\left(M(D) \in S\right) \leq e^{\varepsilon} \cdot \Pr\left(M(D') \in S\right) + \delta.$$

where probabilities are with respect to the coin flips of M.

Following McMahan et al. (2018), we define two common settings of privacy corresponding to two different definitions of neighboring datasets. In *example-level* DP, datasets are considered neighbors when they differ by the addition or removal of a single example (Chaudhuri et al., 2011; Bassily et al., 2014; Abadi et al., 2016b; Papernot et al., 2017; Wu et al., 2017; Papernot et al., 2018; Iyengar et al., 2019). In *user-level* DP, neighboring datasets differ by the addition or removal of *all of the data of one user* (McMahan et al., 2018). User-level DP is the stronger form, and is preferred when one user may contribute many training examples to the learning task, as privacy is protected even if the same privacy-sensitive information occurs in all the examples from one user. In this paper, we will describe the technique and perform experiments in terms of the stronger user-level form, but we note that example-level DP can be achieved by simply giving each user a single example.

To achieve user-level DP, we employ the Federated Averaging algorithm (McMahan et al., 2017), introduced as a decentralized approach to model training in which the training data is left distributed on user devices, and each training round aggregates updates that are computed locally. On each round, a sample of devices are selected for training, and then each selected device performs potentially many steps of local SGD over minibatches of its own data, sending back the model delta as its update.

Bounding the influence of any user in Federated Averaging is both necessary for privacy and often desirable for stabil-

[1] Google, LLC. Correspondence to: Galen Andrew <galenandrew@google.com>.

ity. If left unbounded, any user can potentially cause the learned system to overfit to its data. One way to bound the influence of users is to cap the total L_2 norm of its gradient update: if the update norm is greater than some C, it gets "clipped" to C before being aggregated. Since such clipping also effectively bounds the sensitivity of the aggregate with respect to the addition or removal of any user's data, adding noise to the aggregate is sufficient for achieving a central differential privacy guarantee for the update (Chaudhuri et al., 2011; Bassily et al., 2014; Abadi et al., 2016b). Standard composition techniques can then be used to extend the per-update guarantee to the final model (Mironov, 2017).

Setting an appropriate value for the clipping threshold C is crucial for the utility of the private training mechanism. Setting it too low can result in high bias since we discard information contained in the magnitude of the gradient. However setting it too high would entail the addition of more noise, because the amount of Gaussian noise added to achieve a given level of privacy must be proportional to the clipping norm (L_2 sensitivity), and this will eventually destroy model utility. Thus setting C either too high or too low can adversely affect the utility of the learned model. This effect was observed empirically by McMahan et al. (2018), and the clipping bias-variance tradeoff is theoretically analyzed and shown to be an inherent property of differentially private learning by Amin et al. (2019).

Learning large models using the Federated Averaging/SGD algorithm (McMahan et al., 2017; 2018) can take thousands of rounds of interaction between the central server and the clients. The norms of the updates can vary as the rounds progress. Using a constant clipping threshold (and constant noise) throughout the learning process can result in decreased utility of the system. Prior work (McMahan et al., 2018) has shown that decreasing the value of the clipping threshold after training a language model for some initial number of rounds actually results in increased accuracy of the system. However, the behavior of the norms can be difficult to predict without prior knowledge about the system, and if it is difficult to choose a fixed clipping norm for a given learning task, it is even more difficult to choose a parameterized clipping norm schedule.

While there has been substantial work on DP techniques for learning, almost every technique has hyperparameters which need to be set appropriately for obtaining good utility. Besides the clipping norm, learning techniques have other hyperparameters which might interact with privacy hyperparameters. For example, the server learning rate in DP SGD might need to be set to a high value if the clipping threshold is very low, and vice-versa. Such tuning for large networks can have an exorbitant cost in computation and efficiency, which can be a bottleneck for real-world systems that involve communicating with millions of samples for training a single network. Tuning may also incur an additional cost for privacy, which needs to be accounted for when providing a privacy guarantee for the released model with tuned hyperparameters (though in some cases, hyperparameters can be tuned using the same model and algorithm on a sufficiently-similar public proxy dataset).

Related Work DP-SGD has been the focus of many recent works (Chaudhuri et al., 2011; Bassily et al., 2014; Abadi et al., 2016b; Wu et al., 2017). Privacy amplification via subsampling was introduced in (Kasiviswanathan et al., 2011). The moments accountant, which tightly bounds the privacy loss of the Guassian mechanism when used with amplification via subsampling, was introduced by Abadi et al. (2016a). It was further extended in (McMahan et al., 2018) to incorporate estimating heterogeneous sets of vectors from batches of subsamples. The technique of Federated Averaging was introduced in (McMahan et al., 2017), and was subsequently used in (McMahan et al., 2018) to train recurrent language models with user-level differential privacy. Pichapati et al. (2019) propose a method for coordinate-wise adaptive clipping for DP-SGD, however it also contains a hyperparameter (h_2) that is difficult to tune.

Several works have studied the problem of privacy-preserving hyperparameter tuning. An approach based on target accuracy was provided in (Gupta et al., 2010), which was further improved in terms of privacy cost and computational efficiency in (Liu and Talwar, 2018). A method based on data splitting was provided in (Chaudhuri et al., 2011), whereas one based on satisfying certain stability conditions was introduced in (Chaudhuri and Vinterbo, 2013). These prior works all focused on the general problem of parameter search, whereas we aim to automatically adjust the value of a parameter in iterative procedures to eliminate the need for tuning.

Contributions In this paper, we describe a method for adaptively tuning the clipping threshold to track a given quantile of the update norm distribution during training. We explore the implications of using a fixed clipping norm vs. our adaptive method via extensive experiments on six public FL tasks, using a range of server learning rates and levels of noise. The experiments indicate that on five of the six tasks, adaptive clipping to the median update norm performs as well as any fixed clip chosen in hindsight.

2. Private adaptive quantile clipping

In this section, we will describe the adaptive strategy that can be used for adjusting the clipping threshold so that it comes to approximate the value at a specified quantile.

Let $X \in \mathbb{R}$ be a random variable, let $\gamma \in [0, 1]$ be a quantile

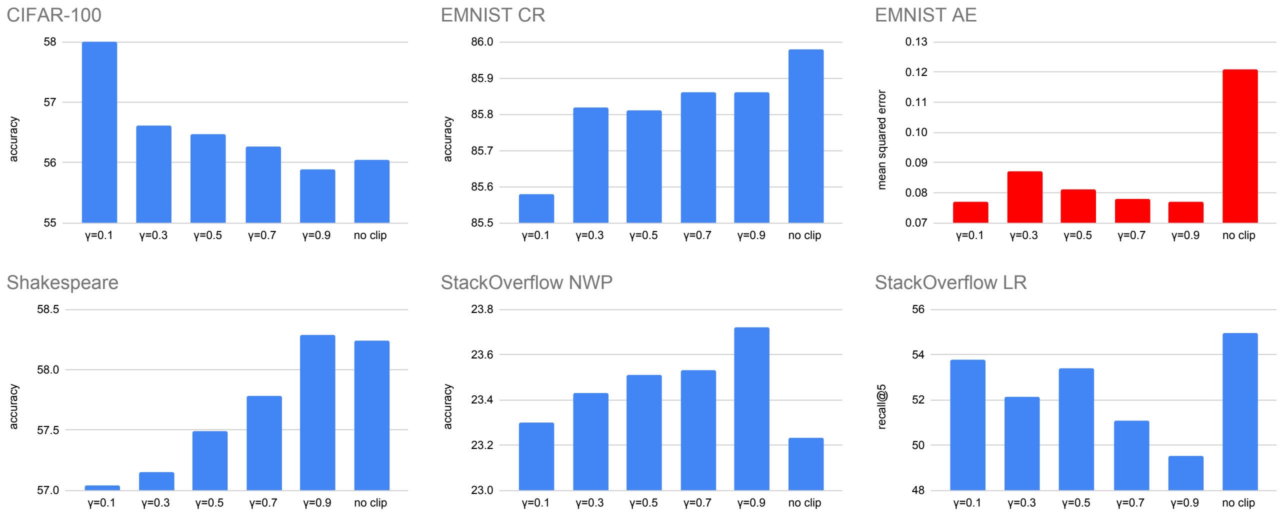

Figure 1. **Impact of clipping without noise.** Performance of the unclipped baseline compared to five settings of γ, from $\gamma = 0.1$ (aggressive clipping) to $\gamma = 0.9$ (mild clipping). The values shown are the evaluation metrics on the validation set averaged over the last 100 rounds. Note that the y-axes have been compressed to show small differences, and that for EMNIST-AE lower values are better.

to be matched. For any C, define

$$\ell_\gamma(C, X) = \begin{cases} (1-\gamma)(C - X) & \text{if } X \le C, \\ \gamma(X - C) & \text{otherwise.} \end{cases}$$

which implies

$$\nabla \ell_\gamma(C, X) = \begin{cases} (1-\gamma) & \text{if } X \le C, \\ -\gamma & \text{otherwise.} \end{cases}$$

Hence, $\mathbb{E}[\nabla \ell_\gamma(C, X)] = (1-\gamma) \cdot \Pr[X \le C] - \gamma \cdot \Pr[X > C] = \Pr[X \le C] - \gamma$.

For C^* s.t. $\mathbb{E}[\nabla \ell_\gamma(C^*, X)] = 0$, we have $\Pr(X \le C^*) = \gamma$. Therefore, C^* is at the γ^{th} quantile of X (Koenker and Bassett Jr, 1978). Because the loss is convex and has gradients bounded by 1, we can produce an online estimate of C that converges to the γ^{th} quantile of X using online gradient descent (see, e.g., Shalev-Shwartz (2012)). See Figure 2 for a plot of the loss function for a discrete random variable that takes six values with equal probability.

Suppose at some round we have n samples of X, with values $(x_1, \ldots, x_n)$. The average derivative of the loss for that round is

$$\begin{aligned} \nabla L_\gamma(C, X) &= \frac{1}{n} \sum_{i=1}^{n} \begin{cases} (1-\gamma) & \text{if } x_i \le C, \\ -\gamma & \text{otherwise} \end{cases} \\ &= \frac{1}{n} \left((1-\gamma) \sum_{i \in [n]} \mathbb{I}_{x_i \le C} - \gamma \sum_{i \in [n]} \mathbb{I}_{x_i > C} \right) \\ &= \bar{b} - \gamma, \end{aligned}$$

where $\bar{b} \equiv \frac{1}{n} \sum_{i \in [n]} \mathbb{I}_{x_i \le C}$ is the empirical fraction of samples with value at most C. For a given learning rate η_C, we can perform the update: $C \leftarrow C - \eta_C(\bar{b} - \gamma)$.

Geometric updates. Since $\bar{b}$ and γ take values in the range $[0, 1]$, the linear update rule described above changes C by a maximum of η_C at each step. This can be slow if C is on the wrong order of magnitude. At the other extreme, if the optimal value of C is orders of magnitude smaller than η_C, the update can be very coarse, and may overshoot to become negative. To remedy such issues, we propose the following geometric update rule: $C \leftarrow C \cdot \exp(-\eta_C(\bar{b} - \gamma))$. This update rule converges quickly to the true quantile even if the initial estimate is orders of magnitude off. It also has the attractive property that the variance of the estimate around the true quantile at convergence is proportional to the value at that quantile. In our experiments, we use the geometric update rule with $\eta_C = 0.2$.

2.1. Adaptive quantile clipping

Let n be the number of users in the population, and $q \in (0, 1]$ be the user selection probability, so the expected number of users in a sample is qn. Let $\gamma \in [0, 1]$ denote the target quantile of the norm distribution at which we want to clip. For iteration $t \in [T]$, let C^t be the clipping threshold, and η_C be the learning rate. Let $\mathcal{Q}^t$ be set of users sampled in round t. Each user $i \in \mathcal{Q}^t$ will send b_i^t along with the usual model delta update Δ_i^t, where bit $b_i^t = \mathbb{I}_{||\Delta_i^t||_2 \le C^t}$.

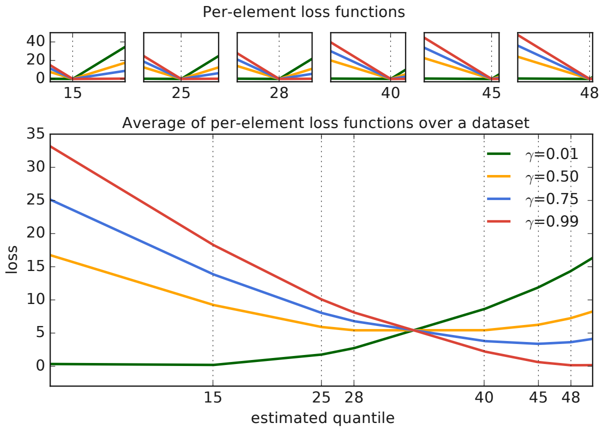

Figure 2. Loss functions to estimate the 0th, 50th, 75th, and 100th quantiles for a random variable that uniformly takes values in $\{15, 25, 28, 40, 45, 48\}$. The loss function is the average of convex piecewise-linear functions, one for each value. For instance, for the median ($\gamma = 0.5$), this is just $\ell_\gamma(C, X) = \frac{1}{2}|X - C|$, where X is the random value, and C is the estimate. When we average these functions, we arrive at the yellow function in the plot showing the average loss, which indeed is minimized by any value between the middle two elements, i.e., in the interval $[28, 40]$. The function for $\gamma = 0.75$ is minimized at $C = 45$ because for values in $[40, 45)$, the quantile is less than γ while for values in $[45, 48)$ the quantile is greater than γ.

We define the loss for user i in the t^{th} round as

$$\ell_\gamma(C^t, \Delta_i^t) = \begin{cases} (1-\gamma)(C^t - ||\Delta_i^t||_2) & \text{if } ||\Delta_i^t||_2 \le C^t, \\ \gamma(||\Delta_i^t||_2 - C^t) & \text{otherwise.} \end{cases}$$

Then define

$$L(C^t) = \frac{1}{qn} \sum_{i \in \mathcal{Q}^t} \ell(C^t, \Delta_i^t),$$

and

$$\bar{b}^t = \frac{1}{qn} \sum_{i \in \mathcal{Q}^t} b_i^t.$$

As $\mathbb{E}[|\mathcal{Q}^t|] = qn$, $\bar{b}^t$ is an unbiased estimate of the fraction of unclipped updates in the t^{th} round. Thus, we have $\nabla L(C^t) = (1-\gamma)\bar{b}^t - \gamma(1 - \bar{b}^t) = \bar{b}^t - \gamma$. Observe that if $\bar{b}^t = \gamma$, then $\nabla L(C^t) = 0$.

However, we can't use $\bar{b}^t$ directly, since it may reveal private information about the magnitude of users' updates. To remedy this, we add Gaussian noise to the sum:

$$\tilde{b}^t = \frac{1}{qn} \left(\sum_{i \in \mathcal{Q}^t} b_i^t + \mathcal{N}(0, \sigma_b^2) \right).$$

Since both the average update and the clipped fraction are estimated with the Gaussian mechanism, we can analyze the combined mechanism on model updates and clipped counts using the technique described by McMahan et al. (2018) for multiple vector groups. Considering the clipped counts b_i^t to be just another vector group with norm at most 1, the effective noise multiplier[1] of the combined mechanism is $z = \left(z_\Delta^{-2} + \sigma_b^{-2}\right)^{-1/2}$. Expressed another way, to achieve a target effective combined noise multiplier of z, one can set the noise multiplier on the updates to $z_\Delta = \left(z^{-2} - \sigma_b^{-2}\right)^{-1/2}$.

Algorithm 1 DPFedAvg-M with adaptive clipping

```
function Train($q$, $\gamma$, $\eta_c$, $\eta_s$, $\eta_C$, $z$, $\sigma_b$, $\beta$)
    Initialize model $\theta^0$, initial clipping bound $C^0$
    $z_\Delta \leftarrow \left(z^{-2} - (2\sigma_b)^{-2}\right)^{-1/2}$
    for each round $t = 0, 1, 2, \dots$ do
        $\mathcal{Q}^t \leftarrow$ (sample users with probability $q$)
        for each user $i \in \mathcal{Q}^t$ in parallel do
            $(\Delta_i^t, b_i^t) \leftarrow$ FedAvg$(i, \theta^t, \eta_c, C^t)$
        $\sigma_\Delta \leftarrow z_\Delta C^t$
        $\tilde{\Delta}^t = \frac{1}{qn}\left(\sum_{i \in \mathcal{Q}^t} \Delta_i^t + \mathcal{N}(0, I\sigma_\Delta^2)\right)$
        $\bar{\Delta}^t = \beta\bar{\Delta}^{t-1} + (1-\beta)\tilde{\Delta}^t$
        $\theta^{t+1} \leftarrow \theta^t + \eta_s \bar{\Delta}^t$
        $\tilde{b}^t = \frac{1}{qn}\left(\sum_{i \in \mathcal{Q}^t} b_i^t + \mathcal{N}(O, \sigma_b^2)\right)$
        $C^{t+1} \leftarrow C^t \cdot \exp\left(-\eta_C(\tilde{b}^t - \gamma)\right)$

function FedAvg($i$, $\theta^0$, $\eta$, $C$)
    $\theta \leftarrow \theta^0$
    $\mathcal{G} \leftarrow$ (user $i$'s local data split into batches)
    for batch $g \in \mathcal{G}$ do
        $\theta \leftarrow \theta - \eta\nabla\ell(\theta; g)$
    $\Delta \leftarrow \theta - \theta^0$
    $b \leftarrow \mathbb{I}_{||\Delta|| \le C}$
    $\Delta' \leftarrow \Delta \cdot \min\left(1, \frac{C}{||\Delta||}\right)$
    return $(\Delta', b)$
```

This analysis can be tightened a bit with a slight conceptual change to the algorithm: instead of sending an indicator $b_i^t \in \{0, 1\}$, we imagine shifting each b_i^t by one half so that it equals -0.5 if the update is unclipped and 0.5 if it is clipped. Then the server can add 0.5 after averaging to determine $\tilde{b}^t$ for the clip update. This reduces the sensitivity of the clipped count query to 0.5, so the noise on the updates needed for equivalent privacy to fixed clipping decreases slightly to

$$z_\Delta = \left(z^{-2} - (2\sigma_b)^{-2}\right)^{-1/2}. \tag{1}$$

[1] The privacy cost of one round of DP-FedAvg (without adaptive clipping) is fully specified by the sampling fraction q along with the ratio $z_\Delta = \sigma_\Delta / C$. We call this quantity the "noise multiplier" because when C^t is variable, it determines the amount of noise to add on that iteration: $\sigma_\Delta^t = z_\Delta C^t$.

In practice, we recommend using a value of $\sigma_b = qn/20$. Since the noise is Gaussian, this implies that the error $|\tilde{b}^t - \bar{b}^t|$ will be less than 0.1 with 95.4% probability, and will be no more than 0.15 with 99.7% probability. Even in this unlikely case, assuming a geometric update and a learning rate of $\eta_C = 0.2$, the error on the update would be a factor of $\exp(0.2 \times 0.15) = 1.03$, a small deviation. So this default gives high privacy for an acceptable amount of noise in the quantile estimation process. Using Eq. 1, we can compute that to achieve an effective combined noise multiplier of $z = 1$, with $qn = 100$ clients per round, the noise multiplier z_Δ is approximately 1.005. So we are paying only a factor of 0.5% more noise on the updates for adaptive clipping with the same privacy guarantee. These constants ($\sigma_b = qn/20$ and $\eta_C = 0.2$) are what we use in the experimental results.

The clipping norm can be initialized to any value C^0 that is safely on the low end of the expected norm distribution. If it is too high and needs to adapt downward, a lot of noise may be added at the beginning, which may swamp the model. However there is not much danger in setting it quite low, since the geometric update will make it grow exponentially until it matches the true quantile. In our experiments we use an initial clip of 0.1 for all tasks. It is easy to compute that with a learning rate of $\eta_C = 0.2$ and a target quantile of $\gamma = 0.5$, if every update is clipped, it would only take about 25 rounds for the quantile estimate to increase by a factor of ten.

The DPFedAvg algorithm with adaptive clipping is shown in Algorithm 1. We augment basic federated averaging with server momentum, which improves convergence (Hsu et al., 2019; Reddi et al., 2020). Note that since the momentum update is computed using privatized estimates of the average client delta, privacy properties are unchanged when momentum is added. We also experimented with adaptive learning rates, but found that they were less effective when noise is added for DP, perhaps because the noise causes the preconditioner v_t to become large prematurely.

3. Experiments

To empirically validate the approach, we examine the behavior of our algorithm on six of the public benchmark federated learning tasks defined by Reddi et al. (2020). Two of the tasks derived from StackOverflow data (SO-NWP and SO-LR) are ideal for DP research due to the very high number of users (342k) making it possible to train models with good user-level privacy without sacrificing accuracy. The other four tasks (CIFAR-100, EMNIST-AE, EMNIST-CR, SHAKESPEARE) are smaller research datasets. They are representative learning tasks, but not representative population sizes for real world cross-device FL applications. We focus on establishing that adaptive clipping works well with 100 clients per round on these tasks in the regime where the noise is at a level such that utility is just beginning to degrade. Under the assumption that a larger population were available, one could increase the number of clients per round and increase the level of noise to achieve comparable utility with high privacy. This should not significantly affect convergence (indeed, it might be beneficial) since the only effect is to increase the number of users in the average $\tilde{\Delta}^t$, reducing the variance. Table 1 shows the number of clients per round with which our experiments indicate we could achieve $(5, 10^{-9})$-DP for each dataset with acceptable model performance loss (less than 5% relative to non-private training, as discussed later) if each dataset had 1M clients.

Task	T	z	qn
CIFAR-100	4000	0.705	2350
EMNIST-CR	1500	0.573	573
EMNIST-AE	3000	0.69	2290
SHAKESPEARE	1200	0.57	570
SO-NWP & SO-LR	1500	0.855	8550

Table 1. Noise multiplier z and number of clients per round qn necessary to achieve $(5, 10^{-9})$-DP with less than 5% model performance loss if each task had a population of 1M. T is the number of training rounds used in our experiments (following Reddi et al. (2020)).

The code used in our experiments is available at [see supplementary review material].

3.1. Baseline client and server learning rates

Reddi et al. (2020) provide optimized client and server learning rates for federated averaging with momentum that serve as a starting point for our experimental setup. For almost all hyperparameters (model configuration, evaluation metrics, client batch size, total rounds, etc.) we replicate their experiments, but with two changes. First, we increase the number of clients per round to 100 for all tasks. This reduces the variance in the updates to a level where we can reasonably assume that adding more clients is unlikely to change convergence properties (McMahan et al., 2018), giving us confidence that our results on the smaller datasets would still hold if it were possible scale up the number of clients as just discussed. Second, as shown in Algorithm 1 we use *unweighted* federated averaging, thus eliminating the need to set yet another difficult-to-fit hyperparameter: the expected total weight of clients in a round.

Since these changes might require different settings, we reoptimize the client and server learning rates for our baseline with no clipping or noise. We ran a small grid of 25 configurations for each task jointly exploring client and server learning rates whose logarithm (base-10) differs from the

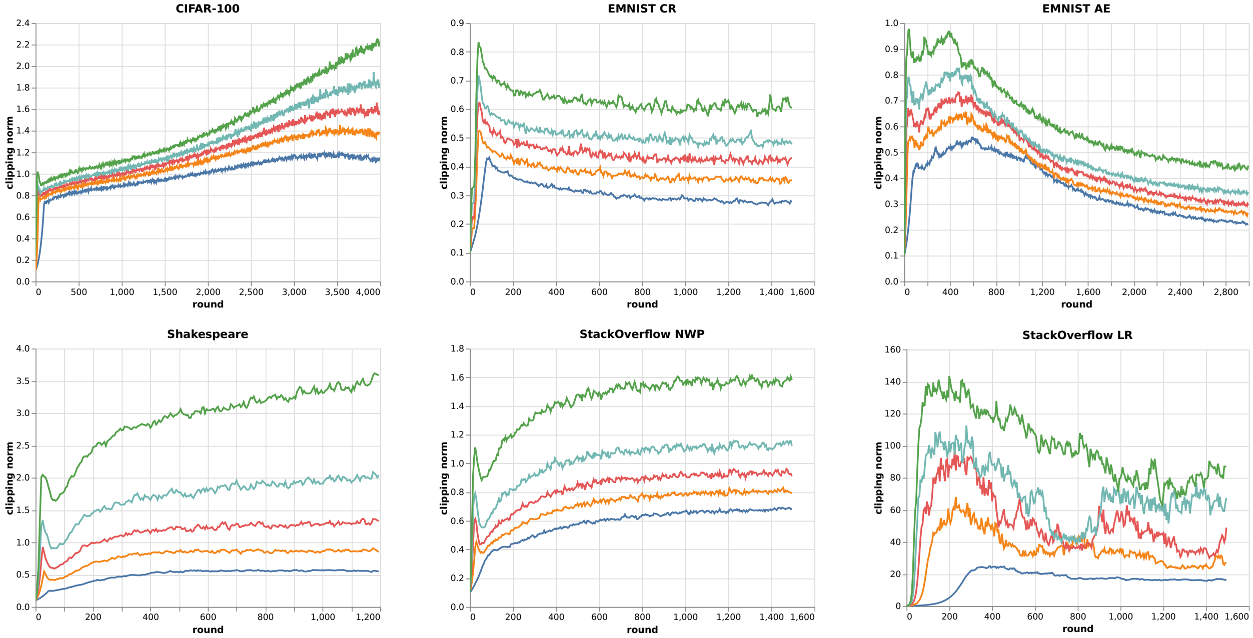

Figure 3. Evolution of the adaptive clipping norm at five different quantiles (0.1, 0.3, 0.5, 0.7, 0.9) on each task with no noise. The norms are estimated using geometric updates with $\eta_C = 0.2$ and an initial value $C^0 = 0.1$.

Task	η_c	η_s	C_{min}	C_{max}
CIFAR-100	0.1	0.32	0.75	2.2
EMNIST-CR	0.032	1.0	0.28	0.85
EMNIST-AE	3.2	1.78	0.22	0.95
SHAKESPEARE	1.0	0.32	0.25	3.6
SO-NWP	0.18	1.78	0.30	1.6
SO-LR	320.0	1.78	16.0	135.0

Table 2. The optimal client and server learning rates (Sec. 3.1) for each task and chosen values of minimum and maximum fixed clips (Sec. 3.2).

values in Table 10 of Reddi et al. (2020) by { -½, -¼, 0, ¼, ½}. The optimal baseline client and server learning rates for our experimental setup are shown in Table 2.

Because clipping (whether fixed or adaptive) reduces the average norm of the client updates, it may be necessary to use a higher server learning rate to compensate. Therefore, for all approaches with clipping—fixed or adaptive—we search over a small grid of five server learning rates, scaling the values in Table 2 by $\{1, 10^{1/4}, 10^{1/2}, 10^{3/4}, 10\}$. For all configurations, we report the best performing model whose server learning rate was chosen from this small grid on the validation set. On no configuration was the optimal server learning rate as high as 10 times the baseline, indicating that this grid is large enough. Note that this modest retuning of the server learning rate is only necessary because we are starting from a configuration that was optimized without clipping. In practice, as we will discuss in Section 4, we would recommend that all hyperparameter optimization should be done with adaptive clipping enabled from the start, eliminating the need for this extra tuning.

We first examine the impact of adaptive clipping without noise to see how it affects model performance. Figure 1 compares baseline performance without clipping to adaptive clipping with five different quantiles. For each quantile, we present the best performing model after tuning over the five server learning rates mentioned above on the validation set. On three tasks (CIFAR-100, EMNIST-AE, SO-NWP) clipping actually improves performance relative to the unclipped baseline. On SHAKESPEARE and SO-LR performance is slightly worse, but we can conclude that adaptive clipping to the median generally fares well compared to not using clipping across tasks. It is worth emphasizing that for our primary goal, training with DP, using some form of clipping is essential.

3.2. Fixed-clip baselines

We would like to compare our adaptive clipping approach to a fixed clipping baseline, but comparing to just one fixed-clip baseline may not be enough to demonstrate that adaptive clipping consistently performs well. Instead, our strategy will be to show that quantile-based adaptive clipping performs as well or nearly as well as *any* fixed clip chosen in hindsight. If we can first identify clipping norms that span the range of normal values during training on each problem/configuration, we can compare adaptive clipping to fixed clipping with those norms.

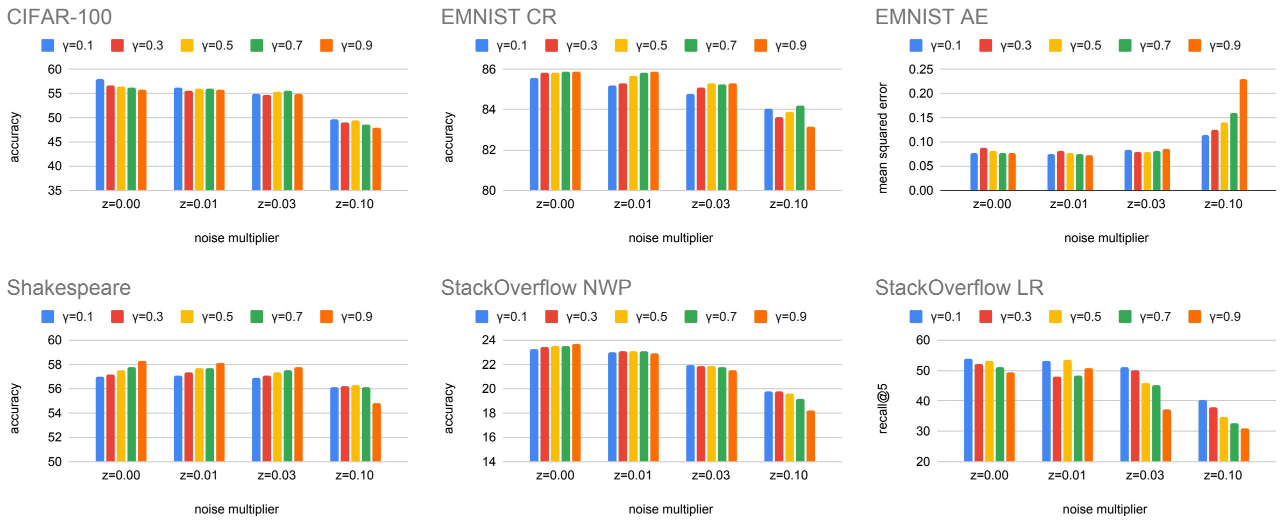

Figure 4. Performance of adaptive clipping with five settings of γ for each of five effective noise multipliers z. Note that the y-axes have been compressed to show small differences, and that for EMNIST-AE lower values are better.

To that end, we first use adaptive clipping without noise to discover the value of the update norm distribution at the following five quantiles: $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Then we choose as the minimum of our fixed clipping range the smallest value at the 0.1 quantile over the course of training, and as the maximum the largest value at the 0.9 quantile. Plots of the update norms during training on each of the tasks are shown in Figure 3.

On each task there is a ramp up period where the clipping norm, which is initialized to 0.1 for all tasks, catches up to the correct norm distribution. Thus we disregard norm values collected until the actual fraction of clipped counts $\bar{b}^t$ on some round is within 0.05 of the target quantile γ. The chosen values for the minimum and maximum fixed clips for each task are shown in Table 2. Our fixed-clipping baseline uses five fixed clipping norms logarithmically spaced in that range. Here we are taking advantage of having already run adaptive clipping to minimize the number of fixed clip settings we need to explore for each task. If we had to explore over the entire range knowing only the endpoints across all tasks (0.22, 135.0) at the same resolution, we would need nearly four times as many clips per task.

For each value of noise multiplier $z \in \{0, 0.01, 0.03, 0.1\}$ we trained using the five fixed clipping norms and compare to adaptive clipping with the five quantiles (0.1, 0.3, 0.5, 0.7, 0.9). Note that for the fixed clipping runs $z_\Delta = z$; that is, for fixed clip C, the noise applied to the the updates has standard deviation zC. As discussed in section 2.1, on the adaptive clipping runs z_Δ is slightly higher due to the need to account for privacy when estimating the clipped counts.

3.3. Comparison of fixed and adaptive clipping

Validation set results with adaptive clipping are shown in Figure 4 and with fixed clipping in Figure 5. These charts show that we have identified the noise regime in which performance is beginning to degrade. There is always a tradeoff between privacy and utility: as the amount of noise increases, eventually performance will go down. For the purpose of this study we consider more than a 5% relative reduction in evaluation metric to be unacceptable. Therefore for each task, we look at the level of noise z^* at which the evaluation metric on the validation set is still within 5% of the value with no noise, but adding more noise would degrade performance beyond 5%. Given z^* for each task, we then choose C^* to be the fixed clip value that gives best performance on the validation set.

For our final test set evaluation, we compare adaptive clipping to the median to fixed clipping at C^*. The results are in Table 3. On most tasks, clipping to the median gives nearly the same performance as the best fixed clip chosen in hindsight. Only on SO-LR the best fixed clip does perform somewhat better. This task seems to be unusual in that best performance comes from very aggressive clipping. However, looking at Figure 5 (and noting the scale of the y axis), on this task more than the others, getting the exact right fixed clip is important. The development set recall@5 value of 55.1 corresponds to the optimal fixed clip of 16.0. The next larger fixed clip of 27.3 gave a recall of only 51.8, and larger clips fared even worse. So an expensive hyperparameter search may be necessary to even get close to this high-performing fixed clip value.

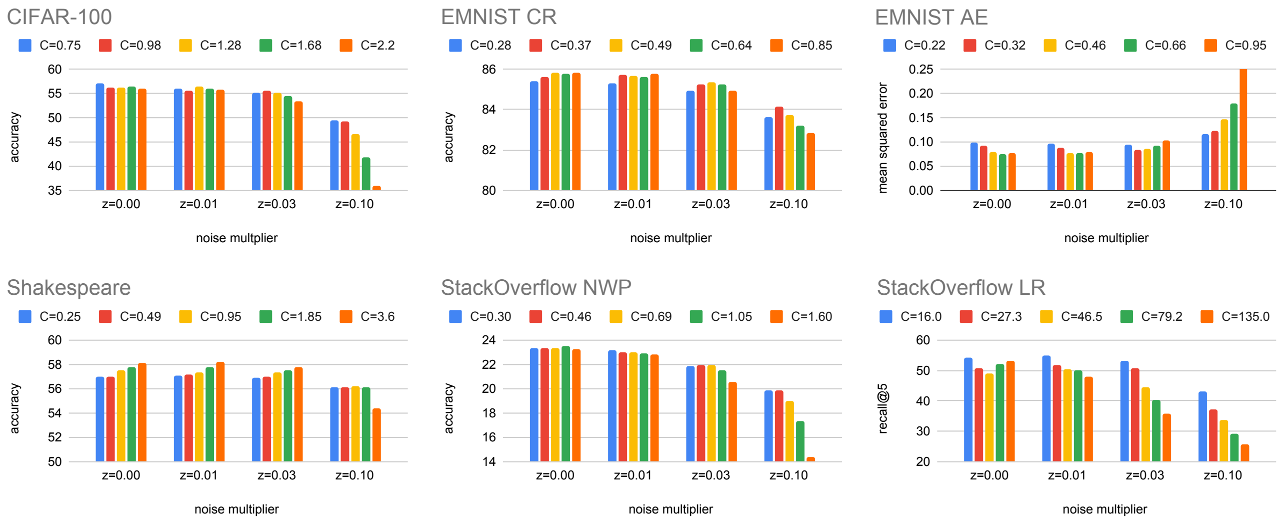

Figure 5. Performance of fixed clipping with five settings of C for each of five noise multipliers z. Note that the y-axes have been compressed to show small differences, and that for EMNIST-AE lower values are better.

Task	z^*	C^*	adaptive	fixed
CIFAR-100	0.03	0.98	55.1	55.1
EMNIST-CR	0.10	0.37	85.0	84.8
EMNIST-AE	0.03	0.32	0.080	0.083
SHAKESPEARE	0.10	0.95	56.4	56.3
SO-NWP	0.01	0.30	24.4	24.4
SO-LR	0.01	16.0	51.5	56.1

Table 3. For each task, with the maximum noise possible before performance begins to significantly degrade (z^*), the best fixed clip (C^*) chosen on the development set, and the test set performance of adaptive clipping to the median compared to fixed clipping to C^*. In practice, finding the best fixed clipping norm would require substantial additional hyperparameter tuning.

4. Conclusions and implications for practice

In our experiments, we started with a high-performing non-private baseline with optimized client and server learning rates. We then searched over a small grid of larger server learning rates for our experiments with clipping (adaptive or fixed). This is one way to proceed in practice, if such non-private baseline results are available. More often, such baseline learning rates are not available, which will necessitate a search over client learning rates as well. In that case, it would be beneficial to enable adaptive clipping to the median *during* that hyperparameter search. The advantage of clipping relative to the unclipped baseline observed on some tasks could only increase if the other hyperparameters such as client learning rate were also chosen conditioned on the presence of clipping.

Although the experiments indicate that adaptive clipping to the median yields generally good results, on some problems (like SO-LR in our study) there may be gains to be had from tuning the target quantile. It would require adding another dimension to the hyperparameter grid, exponentially increasing the tuning effort, but even this would be preferable to tuning the fixed clipping norm from scratch, since the grid can be smaller: we obtained good results on all problems exploring over only five quantiles, but the update norms in the experiments range over three orders of magnitude, from a minimum of 0.22 to a maximum of 135.

Combining our results with the lessons taken from (McMahan et al., 2018) and (Reddi et al., 2020), the following strategy emerges for training a high-performing model with user-level differential privacy. We assume some non-private proxy data is available that may have comparatively few users n', as well as that the true private data has enough users n that the desired level of privacy is achievable.

1. With adaptive clipping to the median enabled, using a relatively small number of clients per round ($qn' \approx 100$), and a small amount of noise ($z = 0.01$), search over client and server learning rates on non-private proxy data.

2. Fix the client and server learning rates. Still using the non-private data and low value of qn', train several models increasing the level of noise z until model performance at convergence begins to degrade.

3. To train the final model on private data, set $q \leftarrow \frac{q\tilde{n}}{n}$ so that the expected number of clients per round is unchanged and performance is likely to be the same. Now if (q, z) is still too small for the desired level of privacy, set $q \leftarrow qR$ and $z \leftarrow zR$ for some R such

that the privacy target ϵ is achieved.[2] Finally, train the private model using that value of q and z.

By eliminating the need to tune the fixed clipping norm hyperparameter which interacts significantly with the client and server learning rates, the adaptive clipping method proposed in this work exponentially reduces the work necessary to perform the expensive first step of this procedure.

[2] Here we employ our assumption that n is sufficiently large, since q cannot exceed 1.

References

Martin Abadi, Andy Chu, Ian Goodfellow, Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *23rd ACM Conference on Computer and Communications Security (ACM CCS)*, 2016a.

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*, CCS '16, pages 308–318, New York, NY, USA, 2016b. ACM. ISBN 978-1-4503-4139-4. doi: 10.1145/2976749.2978318.

Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*, pages 263–271. PMLR, 2019.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 464–473. IEEE, 2014.

Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep Learning Techniques for Music Generation - A Survey. *arXiv e-prints*, art. arXiv:1709.01620, Sep 2017.

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *CoRR*, abs/1802.08232, 2018. URL http://arxiv.org/abs/1802.08232.

Kamalika Chaudhuri and Staal Vinterbo. A stability-based validation procedure for differentially private machine learning. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2652–2660, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999908.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006b.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1322–1333, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. doi: 10.1145/2810103.2813677.

Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1106–1125, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-98-6. URL http://dl.acm.org/citation.cfm?id=1873601.1873691.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Dec 2015. doi: 10.1109/ICCV.2015.123.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification, 2019.

Roger Iyengar, Joseph P. Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. Towards practical differentially private convex optimization. In *S&P 2019*, 2019.

Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3): 793–826, 2011.

Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978.

Jingcheng Liu and Kunal Talwar. Private selection from private candidates. *CoRR*, abs/1811.07971, 2018. URL http://arxiv.org/abs/1811.07971.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282, 2017. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

H. Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A General Approach to Adding Differential Privacy to Iterative Training Procedures. *arXiv e-prints*, art. arXiv:1812.06210, Dec 2018.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *ICLR 2018*, 2018.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. *arXiv e-prints*, art. arXiv:1805.04049, May 2018.

Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

Nicolas Papernot, Martın Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *stat*, 1050, 2017.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate. *arXiv preprint arXiv:1802.08908*, 2018.

Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd, 2019.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization, 2020.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 2012.

R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, May 2017. doi: 10.1109/SP.2017.41.

C. Szegedy, , , P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2773–2781, Cambridge, MA, USA, 2015. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969442.2969550.

X. Wu, M. Fredrikson, S. Jha, and J. F. Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370, June 2016. doi: 10.1109/CSF.2016.32.

Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 1307–1322, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4197-4. doi: 10.1145/3035918.3064047.