# The Role of Adaptive Optimizers for Honest Private Hyperparameter Selection[*]

**Shubhankar Mohapatra**[1], **Sajin Sasy**[1], **Xi He**[1], **Gautam Kamath**[1], **Om Thakkar**[2]
[1]University of Waterloo
[2]Google
{shubhankar.mohapatra, ssasy, xi.he, gckamath}@uwaterloo.ca, omthkkr@google.com

## Abstract

Hyperparameter optimization is a ubiquitous challenge in machine learning, and the performance of a trained model depends crucially upon their effective selection. While a rich set of tools exist for this purpose, there are currently no practical hyperparameter selection methods under the constraint of differential privacy (DP). We study honest hyperparameter selection for differentially private machine learning, in which the process of hyperparameter tuning is accounted for in the overall privacy budget. To this end, we i) show that standard composition tools outperform more advanced techniques in many settings, ii) empirically and theoretically demonstrate an intrinsic connection between the learning rate and clipping norm hyperparameters, iii) show that adaptive optimizers like DPAdam enjoy a significant advantage in the process of honest hyperparameter tuning, and iv) draw upon novel limiting behaviour of Adam in the DP setting to design a new and more efficient optimizer.

## Introduction

Over the last several decades, the field of machine learning has flourished. However, training machine learning models frequently involves personal data, which leaves data contributors susceptible to privacy attacks (Shokri et al. 2017; Carlini et al. 2019; Nasr, Shokri, and Houmansadr 2019; Fredrikson, Jha, and Ristenpart 2015; Song, Ristenpart, and Shmatikov 2017). The leading approaches for privacy-preserving machine learning are based on differential privacy (DP) (Dwork et al. 2006). Informally, DP rigorously limits and masks the contribution that an individual datapoint can have on an algorithm's output. To address the aforementioned issues, DP training procedures have been developed (Williams and McSherry 2010; Bassily, Smith, and Thakurta 2014; Song, Chaudhuri, and Sarwate 2013; Abadi et al. 2016), which generally resemble non-private gradient-based methods, but with the incorporation of gradient clipping and noise injection.

In both settings, *hyperparameter selection* is instrumental to achieving high accuracy. The most common methods are grid search or random search, both of which incur a computational overhead scaling with the number of hyperparameters under consideration. In the private setting, this issue is often magnified as most private training procedures introduce new hyperparameters. More importantly, hyperparameter tuning on a sensitive dataset also costs in terms of *privacy*, naively incurring a multiplicative cost which scales as the square root of the number of candidates (based on composition properties of differential privacy).

Most prior works on private learning choose not to account for this cost (Abadi et al. 2016; Yu et al. 2019; Tramer and Boneh 2021), focusing instead on demonstrating the accuracy achievable by private learning under idealized conditions; if the best hyperparameters were somehow known ahead of time. Recent work (Papernot and Steinke 2021) show that non-private hyperparameter tuning can expose outliers of the dataset. Some assume the presence of supplementary public data resembling the sensitive dataset (Avent et al. 2020; Ramaswamy et al. 2020), which may be freely used for hyperparameter tuning. Such public data may be scarce or nonexistent in settings where privacy is a concern, leaving practitioners with little guidance on how to choose hyperparameters. As explored in our paper, poor hyperparameter selection with standard private optimizers can have catastrophic effects on model accuracy.

Hope is afforded by the success of adaptive optimizers in the non-private setting. The canonical example is Adam (Kingma and Ba 2014), which exploits moments of the gradients to adaptively determine the learning rate. It often works out of the box, providing accuracy comparable with tuned SGD. However, Adam has been overlooked in the private setting since previous works show that fine-tuned DPSGD tends to perform better than DPAdam (Papernot et al. 2020). This leads to several subsequent works (Yu et al. 2021; Tramer and Boneh 2021) that limit themselves to highlighting accuracy under ideal DPSGD hyperparameters. We navigate the different options available to a practitioner to solve the *honest private hyperparameter tuning problem* and ask, *are there optimizers that provide strong privacy, require minimal hyperparameter tuning, and perform competitively with tuned counterparts?*

**Our Contributions** 1) We investigate techniques for private hyperparameter tuning. We perform a comprehensive empirical evaluation of the proposed theoretical method

[*]Authors SM and SS have equal contribution and are listed in alphabetical order. Authors XH, GK, OT have equal contribution and are listed in alphabetical order

of Liu and Talwar (2019) and show that it can be expensive depending on the hyperparameter grid size; in certain cases, one can tune over many hyperparameters using standard composition tools such as moments accountant (Abadi et al. 2016). This analysis extends Koskela and Honkela (2020), in which they focus on comparing DPSGD and their optimizer ADADP, while we analyze the overheads of hyperparameter tuning for any given optimizer. (Section )

2) We empirically and theoretically demonstrate that two hyperparameters, the learning rate and clipping threshold, are intrinsically coupled for non-adaptive optimizers. We show that this coupling makes DPSGD sensitive to these parameter choices, which can drastically affect the validation accuracy. While other hyperparameters and the model architecture are restricted by the scope of the task, privacy, utility targets, and computational resources, the learning rate and clipping norm have no a priori bounds. Since the resulting hyperparameter grid adds up to the privacy cost while tuning to achieve the model with the best utility, we explore leveraging adaptive optimizers, DPAdam, to reduce the hyperparameter space. (Section )

3) We empirically show that the DPAdam optimizer (with default values for hyperparameters), can match the performance of tuned non-adaptive optimizers on a variety of datasets, thus enabling private learning with honest hyperparameter selection. This finding complements a prior claim of Papernot et al. (2020), which suggests that a well-tuned DPSGD can outperform DPAdam. However, our findings show that this difference in performance is relatively insignificant. Furthermore, when hyperparameter tuning is accounted for in privacy loss, we show that DPAdam is much more likely to produce non-catastrophic results. (Section )

4) We show that the adaptive learning rate of DPAdam converges to a static value. We introduce a new private optimizer, DPAdamWOSM that matches DPAdam in performance without computing the second moments. (Section 6)

## Related Work

Hyperparameter tuning plays a vital role in machine learning practice. In the non-private setting, ML practitioners use grid search, random search, Bayesian optimization techniques (Swersky, Snoek, and Adams 2013) or AutoML (He, Zhao, and Chu 2021) techniques to tune their models. However, there hasn't been much research on private hyperparameter tuning procedures due to the significant associated privacy costs. Each set of hyperparameter configurations results in a privacy-utility tradeoff that can be captured by Pareto frontiers using multivariate Bayesian optimization over parameter dimensions (Avent et al. 2020). However, this method asks the model curator to query the dataset multiple times which requires non-private access to the dataset. There are some end-to-end private tuning procedures (Chaudhuri, Monteleoni, and Sarwate 2011; Chaudhuri and Vinterbo 2013; Kusner et al. 2015), that works in restricted settings for few combinations of candidates. The most relevant work to ours is an approach for private selection from private candidates (Liu and Talwar 2019). They provide two methods: one outputs a candidate with accuracy greater than a given threshold, and the other randomly stops and outputs the best candidate seen so far. The first approach is of limited utility in practice as it requires a prior accuracy bound for the dataset. The second variant incurs a considerable overhead in privacy cost. We study the second approach and compare it with a naive approach based on Moments Accountant (Abadi et al. 2016) which would scale as the square root of the number of candidates.

## Problem Setup

Consider a sensitive dataset $D$ which lies beyond a privacy firewall and has $n$ points of the form $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $x_i \in \mathcal{X}$ is the feature vector of the $i$th point and $y_i \in \mathcal{Y}$ is its desired output. Though our experiments are carried out in the supervised setting, all results can be translated to unsupervised setting. The dataset has been partitioned into the training set and the validation set. A trusted curator wants to train a ML model by making queries on the dataset with a total end-to-end training privacy budget of $(\varepsilon_f, \delta_f)$ such that the model can perform with high accuracy on the validation set. The curator wants to try multiple hyperparameter candidates for the model to figure out which candidate gives the maximum accuracy. However, as the model is private, each candidate requires multiple queries made on the dataset and all of them need to be accounted in the total privacy budget of $(\varepsilon_f, \delta_f)$.

Note that any validation accuracy must also be measured privately. Since this accuracy is a low-sensitivity query with a scalar output, and must be computed once per choice of hyperparameters, the cost of this procedure is a lower order term versus the main training procedure. Thus for simplicity, we do not noise these validation accuracy queries. As we will see later, some optimizers require more candidates to tune and hence require more privacy budget than others.

## Privately Tuning DP Optimizers

Effective hyperparameter tuning is crucial in extracting good utility from an optimizer. Unlike the non-private setting, DP optimizers typically i) have more hyperparameters to tune; ii) require additional privacy budget for tuning. Existing work on DP optimizers acknowledge this problem (e.g., (Abadi et al. 2016)), but do not address the privacy cost incurred during hyperparameter tuning (Abadi et al. 2016; Yu et al. 2019; Tramer and Boneh 2021). There are two main prior general-purpose approaches for private hyperparameter selection. The first performs composition via Moments Accountant (Abadi et al. 2016), and the second is the algorithm of Liu and Talwar (2019) (LT). We investigate the privacy cost of these two techniques in practice and discuss situations in which each method is preferred.

**Tuning cost via LT** Liu and Talwar (2019) propose a random stopping algorithm (LT) to output a 'good' hyperparameter candidate from a pool of $K$ candidates, $\{x_1, \ldots, x_K\}$. They assume sampling access to a randomized mechanism $Q(D)$ which samples $i \sim [K]$, and returns the $i$-th candidate $x_i$, and a score $q_i$ for this candidate. It is a random stopping algorithm, in which at every iteration, a candidate is picked from $Q$ i.i.d. with replacement and a $\gamma$-biased coin is flipped to randomly stop the algorithm. When

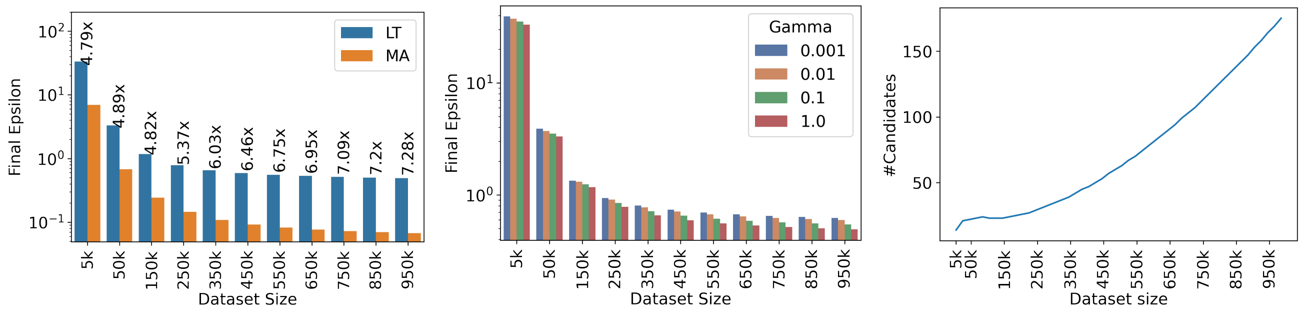

Figure 1: Comparing the privacy cost of LT versus Moments Accountant. The minimal privacy overhead incurred by LT is at least $\approx$ 5x, and increases with the dataset size (left). However, as we allow LT to sample and test more candidate hyperparameters, the privacy cost barely increases (middle). Moments Accountant is able to test a significant number of candidates at the same cost as the minimal privacy overhead of LT (right).

the algorithm stops, the candidate with the maximum score seen so far is outputted. In the approximate DP version of this algorithm, an extra parameter $\Upsilon$ is set to limit the total of number of iterations.

**Theorem 1** ((Liu and Talwar 2019), Theorem 3.4)**.** *Fix any* $\gamma \in [0,1]$, $\delta_2 > 0$ *and let* $\Upsilon = \frac{1}{\gamma} \log \frac{1}{\delta_2}$. *If Q is* $(\varepsilon_1, \delta_1)$*-DP, then the LT algorithm is* $(\varepsilon_f, \delta_f)$*-DP for* $\varepsilon_f = 3\varepsilon_1 + 3\sqrt{2\delta_1}$ *and* $\delta_f = \sqrt{2\delta_1}\Upsilon + \delta_2$.

Theorem 1 expresses the privacy cost of the algorithm in terms of the privacy cost of individual learners, and parameters of the algorithm itself. The $\delta_2$ parameter does not significantly affect the final epsilon $\varepsilon_f$ of the algorithm and in practice, one can set it to a very small value ($10^{-20}$). Though a small value of $\delta_2$ has little effect on $\delta_f$, it increases the hard stopping time of the algorithm, $\Upsilon$.

To understand the minimum privacy cost overhead incurred in the LT algorithm, we first measure its final privacy cost $\varepsilon_f, \delta_f$ budget for training a single hyperparameter candidate ($\gamma = 1$), against the privacy cost $\varepsilon_1, \delta_1$ of the underlying individual learner. Later we show how the privacy cost changes for multiple candidates (varying $\gamma$).

To use LT, one needs to figure out the $\varepsilon_1, \delta_1$ via Theorem 1 from the $\varepsilon_f, \delta_f$ values. Theorem 1 implies that $\delta_1$ needs to be much smaller than $\delta_f$. This change in $\delta_1$ results in a blowup of $\varepsilon_1$ and hence, the final privacy cost of the LT algorithm ($3\varepsilon_1 + 3\sqrt{2\delta_1}$), is much larger than what it would have been for learning one candidate without LT. We call this increase the *blowup* of privacy. We report this blowup in Figure 1(left), for the setting of $\sigma = 4$, $L$=250, $T$=10,000 with varying dataset sizes ($n$). It can be seen that for $n$=5,000, the blowup is 4.8x whereas for for $n$=950,000, the blowup is almost 7.3x. Qualitatively similar trends persist for other choices of noise multiplier, lot size and iterations.

Furthermore, we show that although LT entails a privacy blowup, decreasing $\gamma$ (corresponding to training more individual learners with $\varepsilon_1, \delta_1$) doesn't result in a significant difference in the final epsilon guaranteed by LT. In Figure 1(middle), we show the final epsilon cost for different dataset sizes and varying values of $\gamma \in [0.001, 0.01, 0.1, 1]$. It is interesting to note here that with smaller $\gamma$ values, one can train many candidates (in expectation, $\frac{1}{\gamma}$) for negligible additional privacy cost. The blowup to train 1 candidate ($\gamma = 1$) versus 1,000 candidates ($\gamma = 0.001$) increases from 33 to 39 for $n = 5,000$ and increases from 0.49 to 0.69, for $n = 950,000$. This increase is minimal in comparison to advanced composition, which grows proportional to $\mathcal{O}(\sqrt{k})$.

In summary, the LT algorithm is effective if an analyst has the privacy budget to afford the initial blowup, as the privacy cost of testing additional hyperparameters is insignificant.

**Tuning cost via MA** We now compare LT with tuning using Moments Accountant (MA); for tuning via MA, each hyperparameter candidate is trained by adding necessary Gaussian noise at each iteration, and the best hyperparameter candidate is selected at the end. MA is used for arriving at the final privacy cost of this process. We notice that with the same initial privacy blowup of the LT algorithm, MA is able to compose a considerable number of hyperparameter candidates. In Figure 1(right), we show the number of candidates that can be composed using MA with the minimum privacy cost for running the LT algorithm ($\gamma$=1), for the setting of $\sigma$=4, $L$=250, $T$=10,000 and varying dataset size ($n$). As $T$ and $L$ are set constant, bigger $n$ values in this graph correspond to fewer epochs of training and hence, worse utility. MA can compose 14 candidates for $n$=5000 and up to 175 candidates when $n$=$10^5$. It is surprising how well a standard composition technique performs versus LT. This information can be highly valuable to a practitioner who has limited privacy budget. Qualitatively similar trends persist for other choices of batch size, noise multiplier, and iterations. We note that both MA and LT can leverage the recent PRV accountant (Gopi, Lee, and Wutschitz 2021), which is faster and estimates a tighter bound for the final privacy cost.

From these experiments, we conclude that while tuning with LT entails an initial privacy blowup, the additional privacy cost for trying more candidates (smaller $\gamma$) is minimal. Even though this has an additional computation cost, it can be appealing when an analyst wants to try numerous hyperparameters. On the other hand, for the same overall privacy cost, MA can be used to compose a significant number of hyperparameter candidates. Additionally, MA allows access to all intermediate learners, whereas LT allows access to only the final output parameters. In the sequel, this conclusion will be useful in making the naive MA approach a more appealing tool for some settings (e.g., tighter privacy budgets).

## Tuning DP Optimizers

We detail aspects of tuning both non-adaptive and adaptive optimizers. We start with tuning non-adaptive optimizers. While prior work (Andrew et al. 2021) hints at the existence of a relationship between clip and optimal learning rate, they provide no evidence of it. We theoretically and empirically demonstrate this connection between the learning rate and clipping threshold. We also establish that non-adaptive optimizers inevitably require searching over a large LR-clip grid to extract performant models. Adaptive optimizers forego this problem as they do not need to tune the hyperparameter dimension of learning rate. However, they introduce other hyperparameters that have known good choices in the non-private setting, and later we empirically show that these choices perform well in the private setting as well.

**Tuning DP non-adaptive optimizers** While many hyperparameters are restricted due to computational and privacy/utility targets, the learning rate $\alpha$ and the clipping threshold $C$ have no a priori bounds. In the following theorem, we show an interplay between these parameters by first theoretically analyzing the convergence of DPSGD. We derive a bound on the expected excess risk of DPSGD and while doing so, show that the optimal learning rate, $\alpha_{opt}$, is proportional to the inverse of $C$. We provide the proof in our full version (Mohapatra et al. 2021).

**Theorem 2.** *Let $f$ be a convex and $\beta$-smooth function, and let $x^* = \arg\min_{x \in \mathcal{S}} f(x)$. Let $x_0$ be an arbitrary point in $\mathcal{S}$, and $x_{t+1} = \Pi_{\mathcal{S}}(x_t - \alpha(g_t + z_t))$, where $g_t = \min(1, \frac{C}{\|\nabla f(x)\|^2})\nabla f(x)$ and $z_t \sim \mathcal{N}(0, \sigma^2 C^2)$ is the noise due to privacy. After $T$ iterations, the optimal learning rate is $\alpha_{opt} = \frac{R}{CT\sqrt{1+\sigma^2}}$, where $\mathbb{E}[f(\frac{1}{T}\sum_i^T x_t) - f(x^*)] \leq \frac{RC\sqrt{1+\sigma^2}}{\sqrt{T}}$ and $R = \mathbb{E}[\|x_0 - x^*\|]$.*

Though Theorem 2 gives a closed-form expression for the optimum learning rate for the convergence bound to hold, it is a function of the parameter $R$, which is unknown a priori to the analyst. Given constant $T$ and $\sigma$, the optimal learning rate $\alpha_{opt}$ is inversely proportional to the clipping norm $C$. This is crucial information in practice because these parameters vary among datasets and are unbounded. This unboundedness requires us to search over very large ranges of $C$ and $\alpha$ when we have no prior knowledge of the dataset. It is natural to ask if we can fix the clipping norm $C$ and search only over a wide range for the learning rate $\alpha$ (or vice versa). We explore this relationship experimentally via simulations on a synthetic dataset as well as on the ENRON dataset, showing that fixing one of these two hyperparameters may often but not always result in an optimal model.

We train a linear regression model on a 10-dimensional synthetic dataset of input-label pairs $(x, y)$ sampled from a distribution $\mathcal{D}$ as follows: $x_1, \ldots, x_d \sim \mathcal{U}(0, 1), y = x \cdot w^*, w^* = 10 \cdot \mathbf{1}^d$. We use the initialization $w_0 = \mathbf{0}^d$ and train for 100 iterations. In the non-private setting, this model converges quickly with any reasonable learning rate, but in the private setting, we notice that the training loss depends heavily on the choice of $\alpha$ and $C$. Figure 2(a) shows a heat map for the log training loss when trained on ($\alpha$,$C$) pairs taken from a large grid consisting of $[1, 2, 4, 5, 8]$ at scales of $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^0, 10^1]$. The best training is observed when the loss is close to 0 (white pixels).

From Figure 2(a), We first observe that to achieve the best accuracy, $\alpha$ and $C$ need to be tuned on a large grid spanning several orders of magnitude for each of these parameters. Second, multiple ($\alpha$,$C$) pairs achieve the best accuracy and all lie on the same diagonal, validating our theory for an inverse relation between $\alpha$ and $C$. However, it is not possible to set the clipping norm $C$ constant and tune $\alpha$ (corresponding to a vertical line in Figure 2(c)) or vice versa in order to eliminate a tuning hyperparameter, because not all $C$ or $\alpha$ values can obtain the lowest loss. This phenomenon is evident by noticing that not all vertical or horizontal lines on this figure have white pixels. This happens, for example, at the extremes (e.g., at the top-right corner), but also for several intermediate and standard choices (e.g., $C = 0.1$ or $0.2$). Again, the analyst has no way of knowing this a priori.

Figure 2(b) details the results for the same simulation experiment with DPAdam (with default Adam hyperparameters) as the underlying optimizer. There we notice that the inverse relation between $\alpha$ and $C$ no longer hold as we intuited earlier. Moreover, for a given choice of $\alpha$, there are several clip choices that result in lowest losses. We repeat the same experiment over the ENRON dataset and observe similar trends (Figures 2(c) and 2(d)). We conclude that privately tuning non-adaptive optimizers, requires a larger grid of hyperparameter options than their adaptive counterparts.

**Tuning DP adaptive optimizers** Adaptive optimizers automatically adapt over the learning rate $\alpha$, requiring us to tune only over the clipping norm $C$. But recall our key question: can we train models that perform competitively with the fine-tuned counterparts from DPSGD?

Adam introduces two new hyperparameters, first and second moment exponential decay parameters ($\beta_1$ and $\beta_2$). In the non-private setting, these parameters are relatively insensitive, and default values of $\alpha$=0.001, $\beta_1$=0.9, and $\beta_2$=0.999 are recommended based on empirical findings. Hence before we compare DPAdam and DPSGD, we first find and establish such recommended values for this hyperparameter triple in the DP setting next, and then show that DPAdam with a small hyperparameter space performs competitively with DPSGD in Section .

To establish default choices of $\alpha$, $\beta_1$, and $\beta_2$ for DPAdam, we evaluate this private optimizer over four diverse datasets and two learning models including logistic regression and a neural network with one 100 neurons hidden layer (TLNN). These selected datasets include both low-dimensional data (where the number of samples greatly outnumbers the dimensionality) and high-dimensional data (where the number of samples and dimensionality are at same scale). Since we still have a large hyperparameter space to tune over, for the rest of this work, we fix a constant lot size ($L$=250),

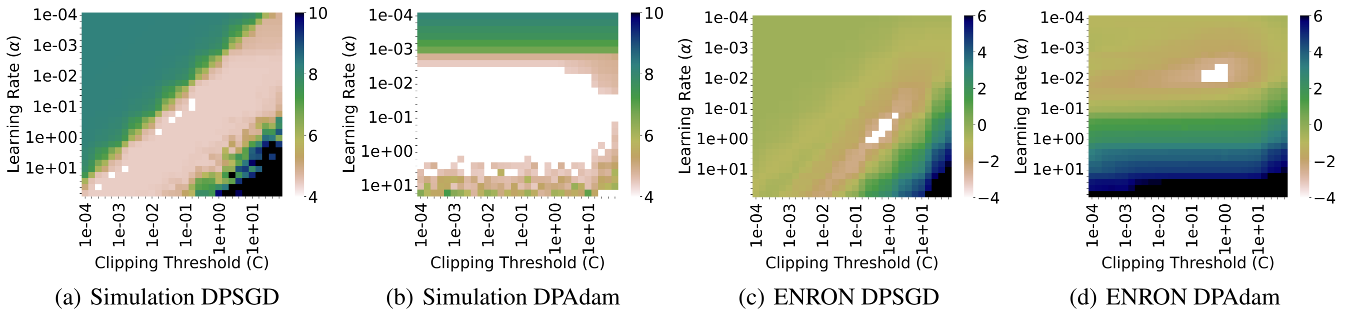

Figure 2: Log of training loss with $\sigma = 4$. In the DPSGD experiments (a,c) the white pixels mark the points with training losses below the 1st percentile. Note that best loss values lie on a diagonal expressing the inverse connection between $\alpha$ and $C$ for DPSGD experiments . Meanwhile white pixels in the DPAdam experiments (b,d) correspond to points that have lower losses than the 1 percentile threshold of the DPSGD counterpart experiments. We see that DPAdam is robust to a wide range of $C$ for a select range of initial $\alpha$.

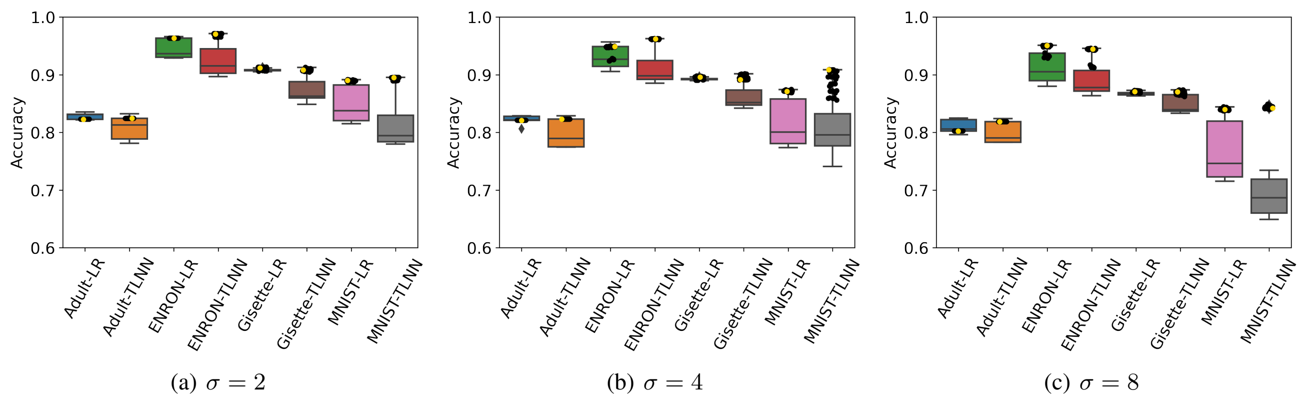

Figure 3: Ranking hyperparameter candidates across datasets. The black points refer to the candidates with $\alpha = 0.001$ (with all permutations of $\beta_1, \beta_2$ from our searchgrid); the gold refers to the candidate with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$

and consider tuning over three different noise levels, $\sigma \in [2, 4, 8]$, so that we can study the effects of tuning the other hyperparameters more thoroughly. All experiments are repeated three times and averaged before reporting. Additionally, in this particular experiment since we focus on $\alpha, \beta_1$, and $\beta_2$, we also fix the clipping threshold $C$=0.5, and $T$=2500 iterations of training. For each dataset and model, we run DPAdam three times with hyperparameters $(\alpha, \beta_1, \beta_2)$ from the grids, $\alpha \in [0.001, 0.05, 0.01, 0.2, 0.5]$, $\beta_1, \beta_2 \in [0.8, 0.85, 0.9, 0.95, 0.99.0.999]$.

| Optimizer | Parameter | Values |
|---|---|---|
| DPSGD | $\alpha$ | 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1 |
| | $C$ | 0.1, 0.2, 0.5, 1 |
| DPMomentum | $\alpha$ | 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1 |
| | $C$ | 0.1, 0.2, 0.5, 1 |
| | $m$ | 0.5, 0.6, 0.7, 0.8, 0.9, 0.99 |
| DPAdam | $C$ | 0.1, 0.2, 0.5, 1 |

Table 1: Hyperparameter Grid

Figure 3 shows the boxplots of testing accuracies of DPAdam over the different hyperparameter choices. When $\alpha$ is 0.001 (same as in the non-private setting), all the datasets and models have final testing accuracies (marked in black) close to the best possible (and in most cases it is the best) accuracy. Furthermore, we also highlight the accuracy of the suggested default choice of Adam ($\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$) using gold dots, which also work well for DPAdam. Hence, we suggest the non-private default values for these parameters in the private setting.

## Advantages of Tuning Using DPAdam

In the non-private setting, adaptive optimizers like Adam enjoy a smaller hyperparameter tuning space than SGD. We ask two questions here. First, can DPAdam (with little tuning) achieve accuracy comparable to a well-tuned DPSGD? Second, what is the privacy-accuracy tradeoff one incurs when using either of the two hyperparameter selection methods detailed in Section .

To answer both questions, we compare DPSGD and DPAdam over the same set of datasets and models and

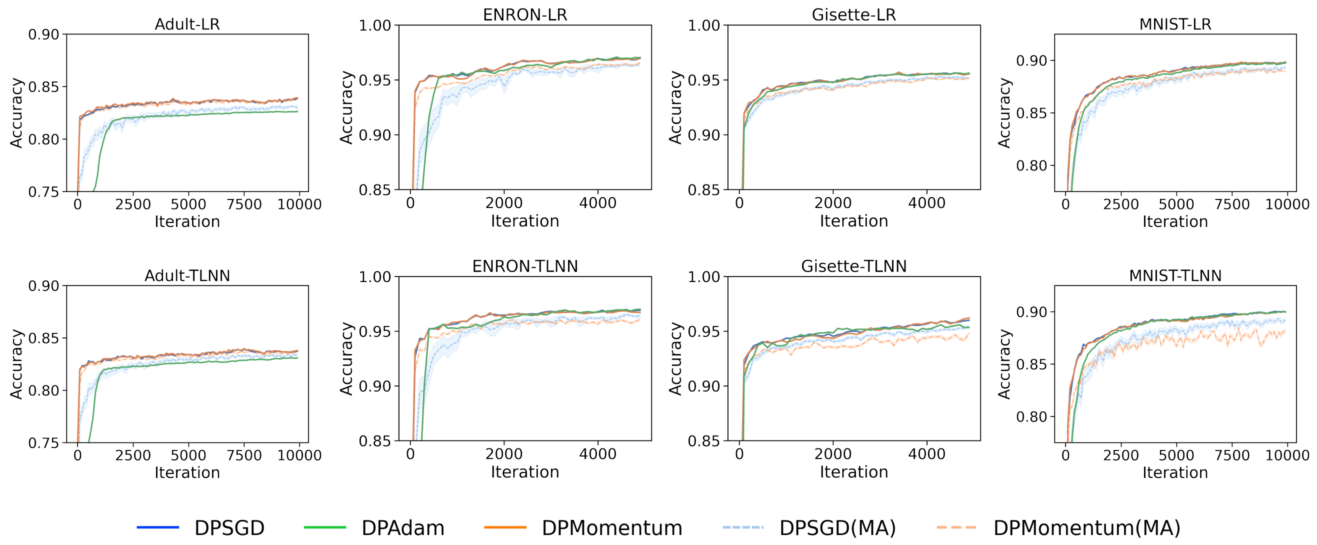

Figure 4: Comparing the testing accuracy curves of DPAdam, DPSGD and DPMomentum models across their hyperparameter tuning grids with $\sigma = 4$. The limits for y-axis are adjusted based on the dataset while maintaining a 15% range for all.

the hyperparameter grids learned in Section . The grids for each optimizer are shown in Table 1, where DPSGD has 40 candidates to tune over and DPAdam has 4 with fixed $\alpha = 0.001, \beta_1 = 0.9$, $\beta_2 = 0.999$. This is because Section shows that DPSGD needs a wide grid to obtain the best accuracy when data distributions are unknown. Additionally, we consider the DPMomentum optimizer. Similar to how we searched for default tuning choices for DPAdam in Section , we investigate if there exists a qualitatively good choice for the momentum hyperparameter, and unfortunately our results show that there is no such choice.

To compare the privacy-accuracy tradeoffs of the three optimizers, we show i) their privacy cost when extracting the best accuracy from these optimizers, and ii) the accuracy one would obtain from them under the tight privacy constraints.

**Prioritizing Accuracy** For brevity, we show experiments for $\sigma = 4$ in Figure 4. Results for other values of $\sigma$ are in the full version (Mohapatra et al. 2021). For each dataset and model, we train each hyperparameter candidate thrice and report the max accuracy every 100 iterations, corresponding to the dark lines for each optimizer. We note that their max accuracies are extremely similar. However, Table 2 shows the final privacy costs incurred by each of these max accuracy lines, and reflects our claims from Section that using fewer hyperparameter candidates and composing privacy via MA gives a much tighter privacy guarantee.

**Prioritizing Privacy** Additionally in Figure 4, DPSGD and DPMomentum have pastel dotted lines corresponding to their mean accuracy attained using the MA composition that provides the tightest privacy guarantees for DPAdam. These pastel lines are the mean accuracy (with 95% CI) from 100 repetitions of this experiment. Since DPAdam has only 4 hyperparameter candidates, for this experiment, we randomly sample 4 candidates for DPSGD and DPMomentum so that they all incur the same privacy cost. Since the candidate pool

| Dataset | DPSGD | DPMomentum | DPAdam |
|---|---|---|---|
| Adult | 5.01 | 5.23 | 1.91 |
| ENRON | 30.86 | 32.31 | 12.80 |
| Gisette | 26.40 | 27.64 | 10.76 |
| MNIST | 3.01 | 3.14 | 1.14 |

Table 2: Final $\varepsilon$ (at $\delta = 10^{-6}$) for optimizers for the LR Models (Figure 4). DPSGD and DPMomentum use LT for privacy accounting; DPAdam uses MA.

is significantly larger for DPSGD and DPMomentum, we additionally scrutinize their parameter grids and prune learning rates that perform poorly. Our pruning process [1] is quite generous, and favours minimizing the hyperparameter space of DPSGD and DPMomentum, but these optimizers perform subpar than DPAdam when constrained with privacy.

## DPAdamWOSM

Besides a decaying average of past gradient updates, DPAdam maintains a decaying average of their second moments. In this section, we design DPAdamWOSM (DPAdam Without Second Moments), a new DP optimizer that operates only using a decaying average of past gradients and eliminates the need to tune the learning rate. We achieve this by analyzing the convergence behavior of the second-moment decaying average in DPAdam in regimes where the scale of noise added is much higher than the scale of the clipped gradients. Setting the *effective step size* (ESS) of DPAdam to the converged constant and removing all computations related to the second-moment up-

[1]Note, pruning itself is unfair; the intent was to design a DP optimizer for any datasets with unknown prior distributions. To do so with DPSGD one would have to consider a significantly wide range of $(\alpha, C)$ to cover 'good' candidates as illustrated in Section .

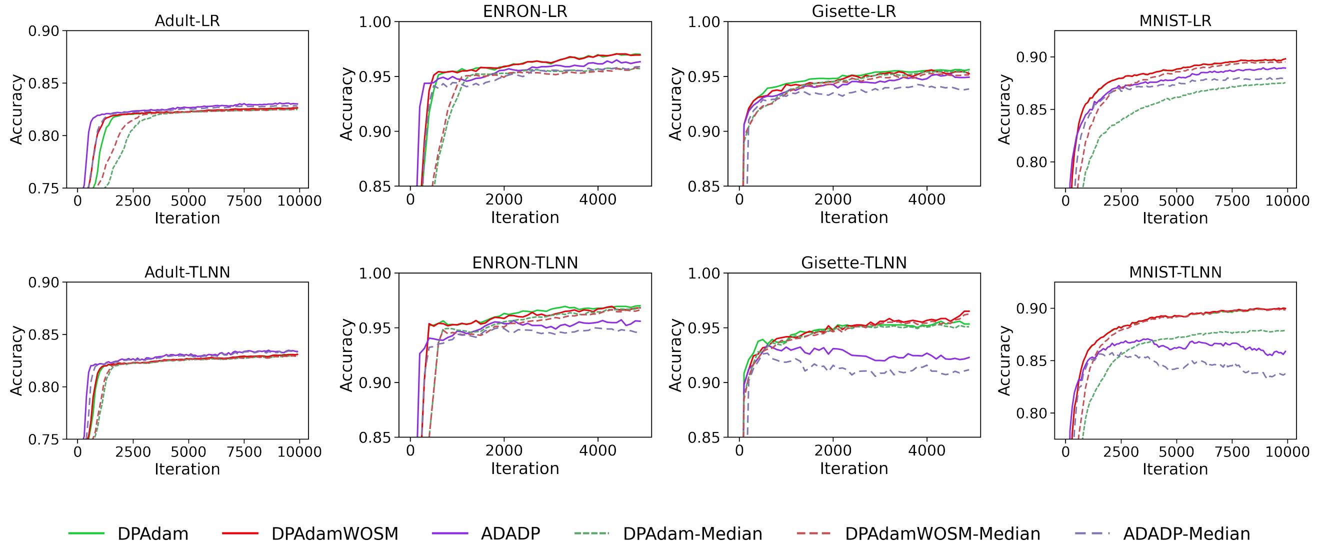

Figure 5: Comparing the testing accuracy curves of DPAdam, ADADP and DPAdamWOSM models across hyperparameter tuning grid from Table 1 with $\sigma = 4$. The y-axes limits are adjusted based on the dataset while maintaining a 15% range for all.

dates, results in DPAdamWOSM. We empirically show that DPAdamWOSM matches the utility of DPAdam while requiring less computation than DPAdam.

Observe that removing the second-moment updates from DPAdam reduces it to DPMomentum with one additional feature: bias-correction to the first-moment decaying average, which DPAdam does to account for its initialization at the origin. While the resulting optimizer still requires tuning the learning rate and other hyperparameters like the clipping threshold, DPAdamWOSM can be viewed as self-tuning the learning rate by fixing it to the converged ESS in DPAdam.

**Effective step size (ESS) in DPAdam** DPAdam results have less variance than DPSGD due to its adaptive learning rate. To understand this phenomenon better, we inspect DPAdam's update step. DPAdam being an adaptive optimizer picks per-parameter ESS as $\frac{\alpha}{\sqrt{\hat{v}_t}+\xi}$, which is the base learning rate $\alpha$ scaled by the second moment of the individual parameter gradients. We notice that when $g \rightarrow 0$, the ESS for DPAdam converges for the first moment gradient, which innately accounts for the clip bound one is training with. This may happen at later iterations, when the model is close to its minima and the gradients get close to zero.

**Theorem 3.** *The effective step size (ESS) for DPAdam with* $g \rightarrow 0$ *converges to* $ESS^* = \frac{\alpha}{(\sigma C/L)+\xi}$.

The proof for Theorem 3 and the pseudo-code for DPAdamWOSM are provided in our full version (Mohapatra et al. 2021). Theorem 3 gives the converged ESS a closed form expression that can be used in place of $\frac{\alpha}{\sqrt{\hat{v}_t}+\xi}$ in the update step from the inception of the learning. Since the second-moment updates (e.g., $\hat{v}_t$) are not used anymore, removing them results in our new optimizer DPAdamWOSM.

**Comparing Adaptive Optimizers** The new optimizer is compared with DPAdam and ADADP. For brevity, we show experiments on $\sigma = 4$ and others appear in the full version (Mohapatra et al. 2021). In Figure 5, we show the maximum and median accuracy curves for all the optimizers. The median accuracy curves (in dotted) are displayed as a quality indicator over the entire hyperparameter grid for a given optimizer; which in this case is strictly over the choices of clip. The max lines for ADADP lies beneath DPAdam and DPAdamWOSM for all dataset except Adult. Also, the max accuracy line for DPAdamWOSM runs alongside DPAdam, implying it can perform as good as DPAdam throughout training. The median line for DPAdamWOSM also performs alongside DPAdam and in some cases is able to beat it (e.g, the median for DPAdamWOSM for MNIST-LR and MNIST-TLNN lies above the median line of DPAdam).

## Conclusion

We thoroughly investigated honest hyperparameter selection for DP optimizers. We compared two existing private methods (LT and MA) to search for hyperparameter candidates, and showed that LT incurs a significant privacy cost but can compose over many candidates, while MA is effective when the number of candidates is small. Next, we explored connections between the clipping norm and the step size to show an inverse relationship between them. Additionally, we compared non-adaptive and adaptive optimizers, demonstrating that the latter typically achieves more consistent performance over a variety of hyperparameter settings. Finally, we brought to light that DPAdam converges to a static learning rate when the noise dominates the gradients. This insight allowed us to derive a novel optimizer DPAdamWOSM, a variant of DPAdam which avoids the second-moment computation and enjoys better accuracy especially at earlier iterations. Future work remains to investigate further implications of these results to provide tuning-free end-to-end private ML optimizers.

## 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 (CCS'16)*.

Andrew, G.; Thakkar, O.; Ramaswamy, S.; and McMahan, H. B. 2021. Differentially Private Learning with Adaptive Clipping. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Avent, B.; González, J.; Diethe, T.; Paleyes, A.; and Balle, B. 2020. Automatic Discovery of Privacy–Utility Pareto Fronts. *Proceedings on Privacy Enhancing Technologies*, 2020(4): 5–23.

Bassily, R.; Smith, A. D.; and Thakurta, A. 2014. Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *FOCS*, 464–473.

Carlini, N.; Liu, C.; Erlingsson, Ú.; Kos, J.; and Song, D. 2019. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *28th USENIX Security Symposium (USENIX Security 19)*, 267–284. Santa Clara, CA: USENIX Association. ISBN 978-1-939133-06-9.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3).

Chaudhuri, K.; and Vinterbo, S. A. 2013. A stability-based validation procedure for differentially private machine learning. *Advances in Neural Information Processing Systems*, 26: 2652–2660.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC '06, 265–284.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. 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)*.

Gopi, S.; Lee, Y. T.; and Wutschitz, L. 2021. Numerical Composition of Differential Privacy. *arXiv preprint arXiv:2106.02848*.

He, X.; Zhao, K.; and Chu, X. 2021. AutoML: A Survey of the State-of-the-Art. *Knowledge-Based Systems*, 212: 106622.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koskela, A.; and Honkela, A. 2020. Learning Rate Adaptation for Differentially Private Learning. In *International Conference on Artificial Intelligence and Statistics*, 2465–2475. PMLR.

Kusner, M.; Gardner, J.; Garnett, R.; and Weinberger, K. 2015. Differentially private Bayesian optimization. In *International conference on machine learning*, 918–927. PMLR.

Liu, J.; and Talwar, K. 2019. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, 298–309.

Mohapatra, S.; Sasy, S.; He, X.; Kamath, G.; and Thakkar, O. 2021. The Role of Adaptive Optimizers for Honest Private Hyperparameter Selection. arXiv:2111.04906.

Nasr, M.; Shokri, R.; and Houmansadr, A. 2019. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *2019 IEEE Symposium on Security and Privacy (SP'19)*.

Papernot, N.; Chien, S.; Song, S.; Thakurta, A.; and Erlingsson, U. 2020. Making the Shoe Fit: Architectures, Initializations, and Tuning for Learning with Privacy.

Papernot, N.; and Steinke, T. 2021. Hyperparameter Tuning with Renyi Differential Privacy. *arXiv preprint arXiv:2110.03620*.

Ramaswamy, S.; Thakkar, O.; Mathews, R.; Andrew, G.; McMahan, H. B.; and Beaufays, F. 2020. Training production language models without memorizing user data. *arXiv preprint arXiv:2009.10031*.

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)*.

Song, C.; Ristenpart, T.; and Shmatikov, V. 2017. Machine Learning Models That Remember Too Much. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*.

Song, S.; Chaudhuri, K.; and Sarwate, A. D. 2013. Stochastic gradient descent with differentially private updates. In *GlobalSIP*, 245–248.

Swersky, K.; Snoek, J.; and Adams, R. P. 2013. Multi-Task Bayesian Optimization. *Advances in Neural Information Processing Systems*, 26: 2004–2012.

Tramer, F.; and Boneh, D. 2021. Differentially Private Learning Needs Better Features (or Much More Data). In *International Conference on Learning Representations*.

Williams, O.; and McSherry, F. 2010. Probabilistic Inference and Differential Privacy. In *NIPS*, 2451–2459.

Yu, D.; Zhang, H.; Chen, W.; Yin, J.; and Liu, T. 2021. Large Scale Private Learning via Low-rank Reparametrization. *International Conference of Machine Learning (ICML) 2021*.

Yu, L.; Liu, L.; Pu, C.; Gursoy, M. E.; and Truex, S. 2019. Differentially Private Model Publishing for Deep Learning. In *2019 IEEE Symposium on Security and Privacy (SP'19)*.