A Preliminaries

Definition 1 (Differential Privacy)**.** *[DKM$^+$06, DMNS06] A randomized algorithm $\mathcal{M}$ achieves (ε, δ)-DP if for all $\mathcal{S} \subseteq$ Range($\mathcal{M}$) and for any two database instances $D, D' \in \mathcal{D}$ that differ only in one tuple:*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta.$$

The privacy cost is measured by the parameters (ε, δ) also referred to as the privacy budget. Smaller values of ε correspond to stricter privacy guarantees, and it is standard in literature to set $\delta \ll \frac{1}{n}$, where n is the size of the database. We set the δ_f in our work to $\frac{1}{n}$ scaled down to the nearest power of 10. Complex DP algorithms can be built from the basic algorithms following two important properties of differential privacy: 1) Post-processing states that for any function g defined over the output of the mechanism $\mathcal{M}$, if $\mathcal{M}$ satisfies (ε, δ)-DP, so does $g(\mathcal{M})$; 2) Basic composition states that if for each $i \in [k]$, mechanism $\mathcal{M}_i$ satisfies $(\varepsilon_i, \delta_i)$-DP, then a mechanism sequentially applying $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ satisfies $(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i)$-DP.

Given a function $f : \mathcal{D} \to \mathbb{R}^d$, the *Gaussian mechanism* adds noise drawn from a normal distribution $\mathcal{N}(0, S_f^2\sigma^2)$ to each dimension of the output, where S_f is the ℓ_2-sensitivity of f, defined as $S_f = \max_{D,D' \text{differ in a row}} \|f(D) - f(D')\|_2$. For $\varepsilon \in (0, 1)$, if $\sigma \geq \sqrt{2\ln(1.25/\delta)}/\varepsilon$, then the Gaussian mechanism satisfies (ε, δ)-DP.

The Gaussian mechanism is used to privatize optimization algorithms. In contrast to non-private optimizers where batches are sliced from the training dataset, DP optimizers at each iteration work by sampling "lots" from the training with probability L/n, where L is the (expected) lot size and n is the total data size. A set of queries are computed over those samples. These queries include gradient computation, updates to batch normalization or accuracy metric calculations. As there is not any a priori bound on these query outputs, the sensitivity S_f is set by clipping the maximum ℓ_2 norm of the gradient to a user-defined parameter C. The gradient of each point is then noised and published. All DP optimizers follow the same framework in which they take steps on the computed noisy gradient as in its non-private counterpart [MAE$^+$18]. The privacy cost of the whole training procedure is calculated by advanced composition techniques such as the Moments accountant [ACG$^+$16].

A.1 DP Optimizers

DP-SGD: The most popular private optimizer is the differentially private stochastic gradient descent (DPSGD) [WM10, BST14, SCS13, ACG$^+$16]. DPSGD takes individual steps for each point in the sampled lot just like in SGD. Due to these individual steps, SGD is more locally unstable and empirically generalizes better than other optimizers [ZFM$^+$20]. However, SGD requires the learning rate to be properly tuned when changing architectures or datasets, without which SGD may show subpar performance.

There are five main hyperparameters involved in DPSGD. We start with those also present in the non-private setting, highlighting any differences that arise due to privacy.

- Training iterations (T) - In the private setting, more iterations results in a larger privacy cost.
- Lot size (L) - Lot size factors into the privacy calculation, due to amplification by subsampling [BBG18].
- Learning rate (α) - Learning rate has an important interplay with the clipping threshold C, discussed in Section 4.1.

The following hyperparameters are new in the private setting.

- Clipping threshold (C) - To limit sensitivity, per-example gradients are clipped to have ℓ_2-norm bounded by C.
- Noise scale (σ) - Scale of the noise added, as a multiple of C. A larger value gives higher privacy but (typically) lower accuracy.

DPMomentum: The private counterpart of SGD-Momentum [RHW86, Qia99], which adds the momentum parameter to the update rule of DPSGD [GAYB17]. This optimizer adds an extra hyperparameter to tune as no default value for momentum is known.

DP-Adam: Adam [KB14] is an adaptive optimizer that combines the advantages from AdaGrad [DHS11] and RMSProp [HSS12]. At the core of Adam, exponentially averaged first and second moment estimates of the gradients are used to take a step. Converting Adam to its differentially private counterpart DPAdam can be done trivially by replacing the standard gradients with their clipped and noised counterparts. Adam adds two extra hyperparameters (β_1, β_2) to tune in the DP setting. However, default values of these parameters are known in the non-private setting. We will tune these parameters to the private setting in Section 4.2. The adaptivity of these optimizers imply they need not be tuned across learning rates, hence reducing a hyperparameter to tune.

ADADP: This DP adaptive optimizer finds the best learning rate at every alternate iteration [KH20]. It does so by leveraging the ℓ_2 error of taking a full step and taking two half steps. If the error computed is greater than a threshold τ, the learning rate is updated using a closed form expression. As suggested by the authors, for all our experiments using ADADP, we use the threshold $\tau = \sqrt{\frac{d}{2T}}$, where d is the model dimension and T is the total number of iterations.

B Dataset details

Table 2: Datasets used in experiment

Dataset	Type	#Samples	#Dims	#Classes
MNIST	Image	70000	784	10
Gisette	Image	6000	5000	2
Adult	Structured	45222	202	2
ENRON	Structured	5172	5512	2

C Parameter grid for DPSGD and DPAdam comparison

Table 3: Parameter grid for comparing DPSGD and DPAdam

Optimizer	Parameter	Values
DPSGD	α	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	α	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, 0.95, 0.99
DPAdam	C	0.1, 0.2, 0.5, 1

D Proof of Theorem

Theorem 4. *Let f be a convex and β-smooth function, and let $x^* = \underset{x \in \mathcal{S}}{\arg\min} f(x)$. 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 $R = \mathbb{E}[\|x_0 - x^*\|]$.*

Proof.

$$\begin{aligned}
\mathbb{E}[\|x_{t+1} - x^*\|^2 \,\Big|\, x_t] &= \mathbb{E}[\|x_t - \alpha(g_t + z_t) - x^*\|^2 \,\Big|\, x_t] \\
&= \mathbb{E}[\|x_t - x^*\|^2 - 2\alpha(g_t + z_t)(x_t - x^*) + \alpha^2\|(g_t + z_t)\|^2 \,\Big|\, x_t] \\
&= \|x_t - x^*\|^2 - 2\alpha\,\mathbb{E}[(g_t + z_t) \,\Big|\, x_t]^T(x_t - x^*) + \alpha^2\,\mathbb{E}[\|(g_t + z_t)\|^2 \,\Big|\, x_t] \\
&\leq \|x_t - x^*\|^2 - 2\alpha[f(x_t) - f(x^*)] + \alpha^2\,\mathbb{E}[\|(g_t + z_t)\|^2 \,\Big|\, x_t]
\end{aligned}$$

The inequality is due to convexity of the loss function and $E[(g_t + z_t)] = g_t$ due to 0-mean noise. Taking expectation on both sides and reordering,

$$\begin{aligned}
2\alpha[f(x_t) - f(x^*)] &\leq \mathbb{E}[\|x_{t+1} - x^*\|^2] - \mathbb{E}[\|x_t - x^*\|^2] + \alpha^2\,\mathbb{E}[\|(g_t + z_t)\|^2 \\
&\leq \mathbb{E}[\|x_{t+1} - x^*\|^2] - \mathbb{E}[\|x_t - x^*\|^2] + \alpha^2(C^2 + C^2\sigma^2)
\end{aligned}$$

Summing for T steps and dividing both sides by $2\alpha T$,

$$\mathbb{E}[f(\frac{1}{T}\sum_i^T x_t) - f(x*)] \leq \frac{R^2}{2\alpha T} + \frac{\alpha C^2(1+\sigma^2)}{2} \tag{1}$$

Taking derivative and finding best value of α,

$$\alpha_{opt} = \frac{R}{C\sqrt{1+\sigma^2 T}}$$

Plugging α_{opt} to Eq. 1,

$$\mathbb{E}[f(\frac{1}{T}\sum_i^T x_t) - f(x*)] \leq \frac{RC\sqrt{1+\sigma^2}}{\sqrt{T}}$$

□

E LT Algorithm

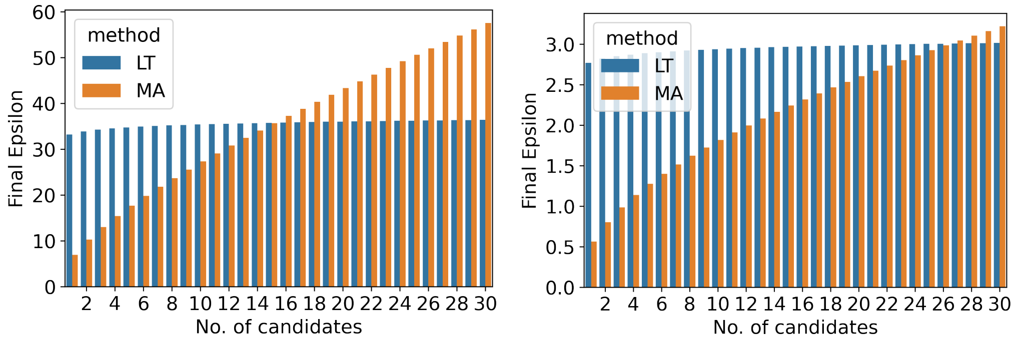
Algorithm 1 Hard stopping private selection algorithm for (ε, δ)-DP input algorithms

Require: $\gamma \leq 1$, $\delta_2 > 0$, and sampling access to $Q(D)$

```
Initialize the list S = ∅
Initialize ϒ = (1/γ) log (1/δ₂)
for j ∈ [1, ϒ] do
    Draw (x, q) ~ Q(D)
    S ← S ∪ (x, q)
    Flip a γ-biased coin, output highest scored candidate from S and halt;
end for
Output highest scored candidate from S
```

F LT vs MA with varying candidate size

Continuing from Section 3.1, in this section we show an additional experiment in which we compare the LT (Liu and Talwar) and MA (Moments Accountant) algorithms with varying number of hyper-parameter candidates. In Figure 6, we run the LT and MA algorithms for $T = 10000$ with $\sigma = 4$ and $L = 250$ with varying candidate size and compare the final privacy costs. The γ value for the LT algorithm is set to $1/k$, where the k is the number of candidates. It can be seen that the privacy cost of LT (blue) remains almost constant for with increasing number of candidates. Figure 6 also demonstrates the exact number of candidates when the cost of MA (orange) remains below the LT cost. This insight is valuable in practice to a practitioner to decide the which algorithm to choose for hyperparameter tuning with respect to the number of candidates.



(a) (b)

Figure 6: Comparing LT vs MA with varying number of candidates at setting $\sigma = 4$, $L = 250$, $T = 10000$. MA can compose upto 14 and 26 candidates for the same cost of LT for dataset sizes 5k (left) and 60k (right) respectively.

G Pruning hyperparameter grid for SGD

Figure 7 demonstrates a heat map plot of the candidate hyperparameter pairs for DPSGD. Each point on this heatmap is assigned a score (totalling 2400) that reflects how many times that (α, C) pair has performed the best among all the candidates, and we score across all iterations (at a granularity of every 100 iterations) of training.

We justify this as a fair metric of 'goodness', for candidates as one could in practice stop training at any iteration. Furthermore this metric is quite critical of quality, in that it only awards a hyperparameter set a point, if it appeared as the best candidate at one of the intervals. Hence we deem this to be a generous pruning of the search space, which will imbue the best possible advantage to DPSGD with regards to a pruned hyperparameter search space.

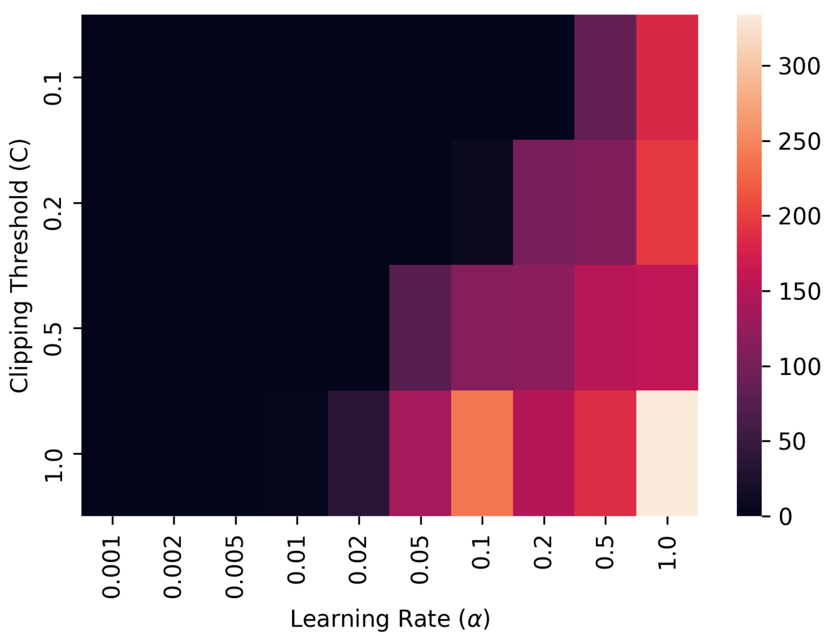


Figure 7: Pruning for DPSGD. Each (α, C) point on the heatmap shows how many times it has performed best among all candidates

H Implementation Details

The code for our paper is written in Python3.6 using the PyTorch library. The implementation of all private optimizers are done using the Pyvacy library[2]. We run our code on ComputeCanada servers. Each allocation of the server includes 2 CPU cores, 8 GB RAM and 1 GPU from the list – P100, V100, K80. We report results from all our experiments after averaging over 3 runs. The code is attached with our supplementary material submission.

All datasets used in our experiments are publicly available. We split all datasets into 80% training and 20% validation sets. For our experiments, we assume that all our datasets start in its preprocessed state, i.e. the numerical features are scaled to the range [0,1], as is standard practice in machine learning. However when considering an end-to-end private algorithm, this preprocessing itself may need to be performed in a privacy-preserving fashion. In this work, we do not account for privacy in this step. Note that for our work this only effects the ENRON and Adult datasets, where scaling the values does require computing the maximum possible values of features in a differentially-private fashion, whereas the max values for image datasets (Gisette and MNIST) are known a priori due to max pixel value and does not involve any privacy cost.

I Additional experiment results for Section 5 and Section 6

In Figures 8 and 9, we display our results for the same experiments described in Section 5, with $\sigma = 2$, and $\sigma = 8$ respectively. Similarly Figure 10 and 11 displays our results of the experiments detailed in Section 6 with $\sigma = 2$, and $\sigma = 8$.

J Omitted Pseudocode for DPAdamWOSM

Algorithm 2 Optimization using DPAdamWOSM

Require: Training set $A : \{x_1, ..., x_n\}$, Loss function $\mathcal{L}(\theta)$, Parameters: Lot size L, Learning rate α, Gradient norm bound C, Noise scale σ, Total number of iterations T, Exponential decay rate β_1

Initialize model with θ_0 randomly
Initialize first moment vector $m_0 = 0$
Set learning rate to ESS $\alpha = \frac{10^{-3}}{(\sigma C/L)+10^{-8}}$;
for $t \in [1, T]$ **do**
  Sample a random subset $L_t \subseteq A$, by independently including each element of A with probability L/n
  Compute gradient $\forall x_i \in L_t$
  $g_t(x_i) = \nabla_\theta \mathcal{L}(\theta_t, x_i)$
  Clip each gradient in ℓ_2 norm to C $\bar{g}_t(x_i) = g_t(x_i)/\max(1, \frac{\|g_t(x_i)\|_2}{C})$
  Add noise $\tilde{g}_t = \frac{1}{|L|}(\sum_i \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 I))$
  Exponentially average the first moment
  $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \tilde{g}_t$
  Perform bias correction
  $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$
  Update model $\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t$
end for
Compute privacy cost using Moments Accountant.

Broader Impacts

Our work points out a false sense of security afforded by prior work in the space of differentially private machine learning, as true privacy losses are much larger than what is typically reported in papers. That said, regardless, differential privacy is a very difficult topic to properly deploy and genuinely provide its theoretical guarantees, rather than just a mirage of privacy. These issues can

[2] https://github.com/ChrisWaites/pyvacy

be avoided via training and/or consultation with data privacy experts, although this may be more challenging for smaller, resource-constrained organizations.

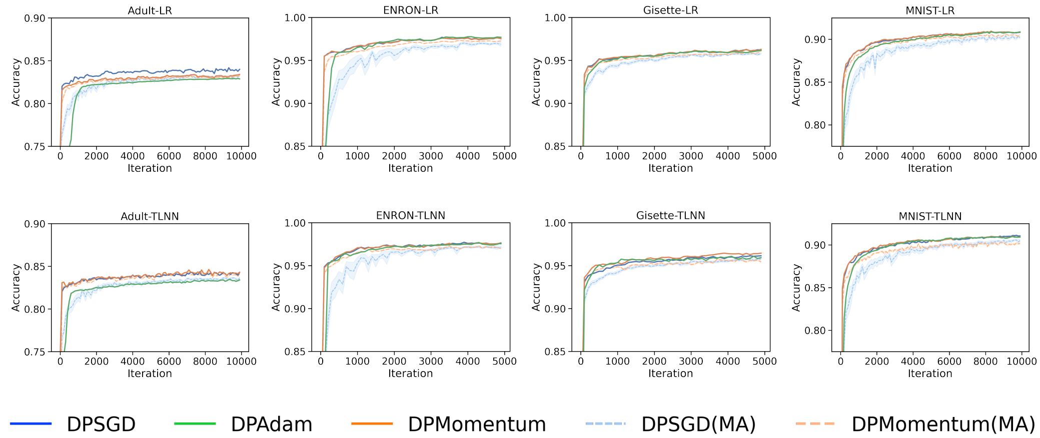


Figure 8: Comparing the testing accuracy curves of DPAdam and DPSGD models across hyperparameter tuning grid from Table 3 with $\sigma = 2$.

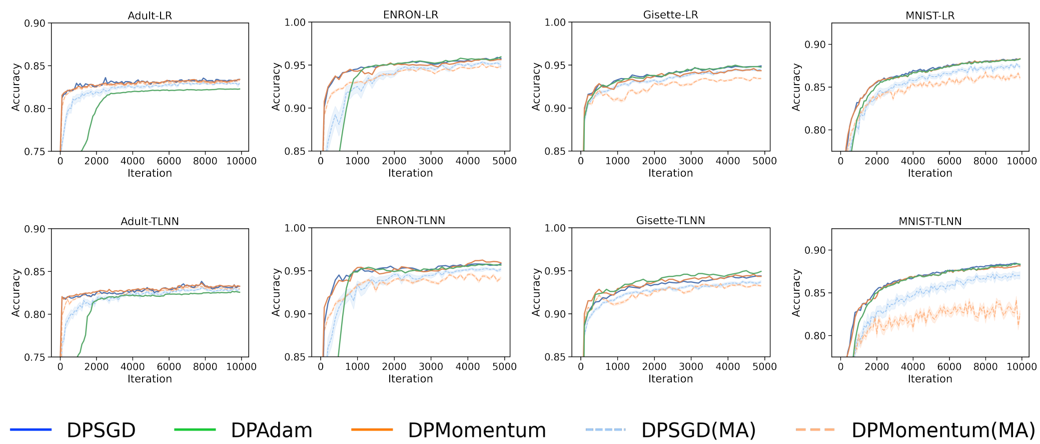


Figure 9: Comparing the testing accuracy curves of DPAdam and DPSGD models across hyperparameter tuning grid from Table 3 with $\sigma = 8$.

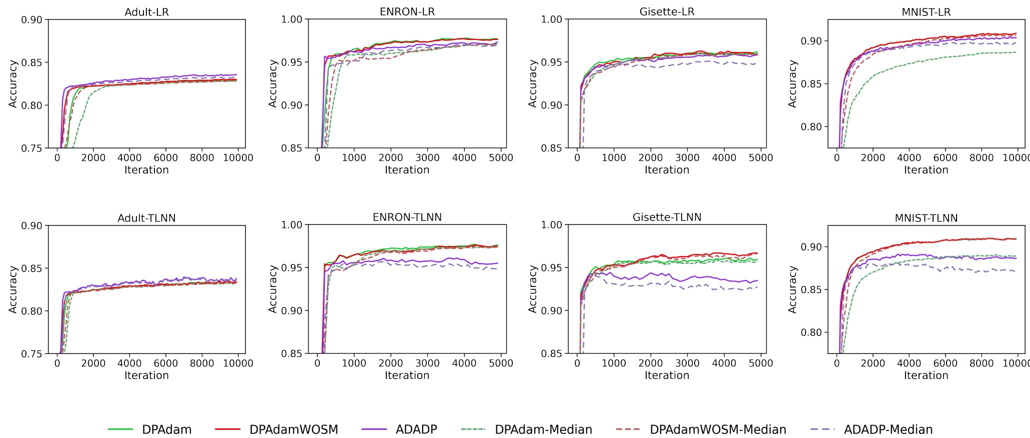


Figure 10: Comparing the testing accuracy curves of DPAdam, ADADP and DPAdamWOSM models across hyperparameter tuning grid from Table 3 with $\sigma = 2$. The limits for the y-axes are adjusted based on the dataset while maintaining a 15% range for all.

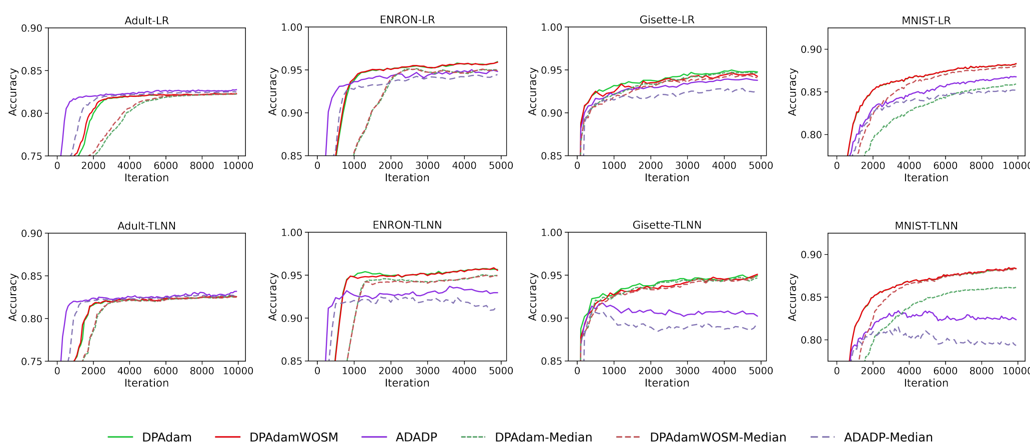


Figure 11: Comparing the testing accuracy curves of DPAdam, ADADP and DPAdamWOSM models across hyperparameter tuning grid from Table 3 with $\sigma = 8$. The limits for the y-axes are adjusted based on the dataset while maintaining a 15% range for all.