PRACTICAL DIFFERENTIALLY PRIVATE HYPERPARAMETER TUNING WITH SUBSAMPLING

Antti Koskela
Nokia Bell Labs
antti.h.koskela@nokia-bell-labs.com

Tejas Kulkarni
Nokia Bell Labs
tejas.kulkarni@nokia-bell-labs.com

ABSTRACT

Tuning the hyperparameters of differentially private (DP) machine learning (ML) algorithms often requires use of sensitive data and this may leak private information via hyperparameter values. Recently, Papernot & Steinke (2022) proposed a certain class of DP hyperparameter tuning algorithms, where the number of random search samples is randomized itself. Commonly, these algorithms still considerably increase the DP privacy parameter ε over non-tuned DP ML model training and can be computationally heavy as evaluating each hyperparameter candidate requires a new training run. We focus on lowering both the DP bounds and the computational cost of these methods by using only a random subset of the sensitive data for the hyperparameter tuning and by extrapolating the optimal values to a larger dataset. We provide a Rényi differential privacy analysis for the proposed method and experimentally show that it consistently leads to better privacy-utility trade-off than the baseline method by Papernot & Steinke (2022).

1 INTRODUCTION

Our aim is two-fold: to decrease the computational cost as well as the privacy cost of hyperparameter tuning of DP ML models. The reasons for this are clear. As the dataset sizes grow and models get more complex, blackbox optimization of hyperparameters becomes more expensive since evaluation of a single set of hyperparameters often requires retraining a new model. On the other hand, tuning the hyperparameters often depends on the use of sensitive data, so it requires privacy protection as well, as illustrated by the counterexample by Papernot & Steinke (2022). Intuitively, the leakage from hyperparameters is much smaller than from the model parameters, however providing tuning algorithms with low additional DP cost has turned out challenging. Current best algorithms (Papernot & Steinke, 2022) still come with a considerable DP cost overhead.

Although our methods and results are applicable to general DP mechanisms, we will in particular focus on tuning of the DP stochastic gradient descent (DP-SGD) (Song et al., 2013; Bassily et al., 2014; Abadi et al., 2016). Compared to plain SGD, DP brings additional hyperparameters to tune: the noise level σ and the clipping constant C. We use the results by Papernot & Steinke (2022) as building blocks of our methods. Their work was based on the analysis of Liu & Talwar (2019) who provided the first results for DP black-box optimization of hyperparameters, where, if the base training algorithm is $(\varepsilon, 0)$-DP, then the tuned model is approximately $(3\varepsilon, 0)$-DP. Mohapatra et al. (2022) showed that for reasonable numbers of adaptively chosen private candidates a naive RDP accounting (i.e., release all models) often leads to lower DP bounds than the methods by Liu & Talwar (2019). Papernot & Steinke (2022) gave Rényi differential privacy (RDP) analysis for black-box tuning algorithms where the guarantees grow only logarithmically w.r.t. number of model evaluations. As the privacy bounds are in terms of RDP and assume only RDP bounds about the candidate model training algorithms, they are particularly suitable to tuning DP-SGD. However, still, running these algorithms increases the ε-values two or three-fold or more, and they can be computationally heavy as evaluating each candidate model requires training a new model.

Our novelty is to consider using only a random subset of the sensitive data for the tuning part and use the output hyperparameter values and the corresponding model for training subsequent models. This automatically leads to both lower DP privacy leakage and lower computational cost. We mention that in non-DP setting, small random subsets of data have been used in Bayesian optimization of hyperparameters, see e.g. (Swersky et al., 2013; Klein et al., 2017). The full version of this paper can be found in (Koskela & Kulkarni, 2023).

1.1 Our Contributions

- We propose a subsampling strategy to lower the privacy and compute cost of DP hyperparameter tuning. We provide a tailored RDP analysis for the proposed strategy. Our analysis is in terms of RDP and we use existing results for tuning Papernot & Steinke (2022) and DP-SGD (Zhu & Wang, 2019) as building blocks.
- We propose algorithms to tune hyperparameters that affect the RDP guarantees of the base model training algorithms. We provide a rigorous RDP analysis for these algorithms.
- We carry out experiments on several standard datasets, where we are able to improve upon the baseline tuning method by a clear margin. While our experiments focus mainly on training of deep learning models with DP-SGD and DP-Adam, our framework is currently applicable to any computation that involves selecting the best among several alternatives (consider e.g., DP model selection, Thakurta & Smith, 2013).

2 Background: DP, DP-SGD and DP Hyperparameter Tuning

We first give the basic definitions. An input dataset containing n data points is denoted as $X = (x_1, \ldots, x_n) \in \mathcal{X}^n$, where $x_i \in \mathcal{X}$, $1 \leq i \leq n$. We say X and Y are neighbours if we get one by adding or removing one data element to or from the other (denoted $X \sim Y$). Consider a randomized mechanism $\mathcal{M} : \mathcal{X}^n \rightarrow \mathcal{O}$, where $\mathcal{O}$ denotes the output space. The (ε, δ)-definition of DP can be given as follows (Dwork, 2006).

Definition 1. *Let $\varepsilon > 0$ and $\delta \in [0, 1]$. We say that a mechanism $\mathcal{M}$ is (ε, δ)-DP, if for all neighbouring datasets X and Y and for every measurable set $E \subset \mathcal{O}$ we have:*

$$\Pr(\mathcal{M}(X) \in E) \leq \mathrm{e}^{\varepsilon} \Pr(\mathcal{M}(Y) \in E) + \delta.$$

We will also use the Rényi differential privacy (RDP) (Mironov, 2017) which is defined as follows. Rényi divergence of order $\alpha > 1$ between two distributions P and Q is defined as

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \int \left(\frac{P(t)}{Q(t)} \right)^\alpha Q(t) \, \mathrm{d}t. \tag{2.1}$$

Definition 2. *We say that a mechanism $\mathcal{M}$ is (α, ε)-RDP, if for all neighbouring datasets X and Y, the output distributions $\mathcal{M}(X)$ and $\mathcal{M}(Y)$ have Rényi divergence of order α at most ε, i.e.,*

$$\max_{X \sim Y} D_\alpha\big(\mathcal{M}(X)||\mathcal{M}(Y)\big) \leq \varepsilon.$$

We can convert from Rényi DP to approximate DP using, for example, the following formula:

Lemma 3 (Canonne et al. 2020). *Suppose the mechanism $\mathcal{M}$ is $\big(\alpha, \varepsilon'\big)$-RDP. Then $\mathcal{M}$ is also $(\varepsilon, \delta(\varepsilon))$-DP for arbitrary $\varepsilon \geq 0$ with*

$$\delta(\varepsilon) = \frac{\exp\big((\alpha - 1)(\varepsilon' - \varepsilon)\big)}{\alpha} \left(1 - \frac{1}{\alpha} \right)^{\alpha - 1}. \tag{2.2}$$

As is common, in practice we carry out the RDP accounting such that we do bookkeeping of total $\varepsilon(\alpha)$-values for a list of RDP-orders (e.g. integer α's) and in the end convert to (ε, δ)-guarantees by minimizing over the values given by equation 2.2. RDP accounting for compositions of DP mechanisms is carried using standard RDP composition results (Mironov, 2017).

DP-SGD differs from SGD such that sample-wise gradients of a random mini-batch are clipped to have L_2-norm at most C and normally distributed noise with variance σ^2 is added to the sum of the gradients of the mini-batch (Abadi et al., 2016). One iteration is given by

$$\theta_{j+1} = \theta_j - \eta_j \Big(\frac{1}{|B|} \sum_{x \in B_j} \mathrm{clip}(\nabla f(x, \theta_j), C) + Z_j \Big), \tag{2.3}$$

where $Z_j \sim \mathcal{N}(\mathbf{0}, \frac{C^2\sigma^2}{|B|^2}\mathbb{I}_d)$, and η_j denotes the learning rate hyperparameter and $|B|$ is the expected batch size (if we carry out Poisson subsampling of mini-batches, $|B_j|$ varies). There are several results that enable the RDP analysis of DP-SGD iterations (Abadi et al., 2016; Balle et al., 2018; Zhu & Wang, 2019). The following result by Zhu & Wang (2019) is directly applicable to analyzing DP-SGD, however we also use it for analyzing a variant of our hyperparameter tuning method.

Theorem 4 (Zhu & Wang 2019)**.** *Suppose $\mathcal{M}$ is a $\big(\alpha, \varepsilon(\alpha)\big)$-RDP mechanism, w.r.t. to the add/remove neighbourhood relation. Consider the subsampled mechanism $(\mathcal{M} \circ \mathrm{subsample}_{\mathrm{Poisson}(\gamma)})(X)$. If $\mathcal{M}$ is $\big(\alpha, \varepsilon(\alpha)\big)$-RDP then $\mathcal{M} \circ \mathrm{subsample}_{\mathrm{Poisson}(q)}$ is $\big(\alpha, \varepsilon'(\alpha)\big)$-RDP ($\alpha \geq 2$ is an integer), where*

$$\varepsilon'(\alpha) = \frac{1}{\alpha - 1} \log \Bigg((1-\gamma)^{\alpha-1}(\alpha\gamma - \gamma + 1) + \binom{\alpha}{2} \gamma^2 (1-\gamma)^{\alpha-2} \exp(\varepsilon(2)) + 3 \sum_{j=3}^{\alpha} \binom{\alpha}{j} \gamma^j (1-\gamma)^{\alpha-j} \exp((j-1)\varepsilon(j)) \Bigg).$$

We remark that the recent works (Koskela et al., 2020; Gopi et al., 2021; Zhu et al., 2022) give methods to carry out (ε, δ)-analysis of DP-SGD tightly.

Intuitively, the leakage from hyperparameters is much smaller than from the model parameters, however considering it in the final accounting is needed to ensure rigorous DP guarantees. In the results of Papernot & Steinke (2022) it is important that the number of candidate models K is randomized. They analyze various distributions for drawing K, however we focus on using the Poisson distribution as it is the most concentrated around the mean among all the alternatives. The corresponding hyperparameter tuning algorithm and its privacy guarantees are given as follows.

First recall: K is distributed according to a Poisson distribution with mean $\mu > 0$, if for all non-negative integer values k: $\mathbb{P}(K = k) = \mathrm{e}^{-\mu} \cdot \frac{\mu^k}{k!}$.

Theorem 5 (Papernot & Steinke 2022)**.** *Let $Q : \mathcal{X}^N \to \mathcal{Y}$ be a randomized algorithm satisfying $\big(\alpha, \varepsilon(\alpha)\big)$-RDP and $(\widehat{\varepsilon}, \widehat{\delta})$-DP for some $\alpha \in (1, \infty)$ and $\varepsilon, \widehat{\varepsilon}, \widehat{\delta} \geq 0$. Assume $\mathcal{Y}$ is totally ordered. Let the Poisson distribution parameter $\mu > 0$. Define the hyperparameter tuning algorithm $A : \mathcal{X}^N \to \mathcal{Y}$ as follows. Draw K from a Poisson distribution with mean μ. Run $Q(X)$ for K times. Then $A(X)$ returns the best value of those K runs (both the hyperparameters and the model parameters). If $K = 0$, $A(X)$ returns some arbitrary output. If $\mathrm{e}^{\widehat{\varepsilon}} \leq 1 + \frac{1}{\alpha - 1}$, then A satisfies $\big(\alpha, \varepsilon'(\alpha)\big)$-RDP, where $\varepsilon'(\alpha) = \varepsilon(\alpha) + \mu \cdot \widehat{\delta} + \frac{\log \mu}{\alpha - 1}$.*

3 DP Hyperparameter Tuning with a Random Subset

We next consider our main tool: we carry out the private hyperparameter tuning on a random subset, and if needed, extrapolate the found hyperparameter values to larger datasets that we use for training subsequent models. In our approach described below, the subset of data used for tuning is generally smaller than the data used for training the final model and thus we extrapolate the hyperparameter values.

3.1 Our method: Small Random Subset for Tuning

Our method works as below:

1. Use Poisson subsampling to draw $X_1 \subset X$: draw a random subset X_1 such that each $x \in X$ is included in X_1 with probability q.
2. Compute $(\theta_1, t_1) = \mathcal{M}_1(X_1)$, where $\mathcal{M}_1$ is a hyperparameter tuning algorithm (e.g., method by Papernot & Steinke, 2022) that outputs the vector of optimal hyperparameters t_1 and the corresponding model θ_1.
3. If needed, extrapolate the hyperparameters t_1 to the dataset $X \setminus X_1$: $t_1 \to t_2$.
4. Compute $\theta_2 = \mathcal{M}_2(t_2, X \setminus X_1)$, where $\mathcal{M}_2$ is the base mechanism (e.g., DP-SGD).

Denote the whole mechanism by $\mathcal{M}$. Then, we may write

$$\mathcal{M}(X) = \big(\mathcal{M}_1(X_1), \mathcal{M}_2\big(\mathcal{M}_1(X_1), X \setminus X_1\big). \tag{3.1}$$

Additionally, we consider a variation of our method in which we use consider the full data set X instead of $X \setminus X_1$ in step 3 onwards.

$$\mathcal{M}(X) = \big(\mathcal{M}_1(X_1), \mathcal{M}_2\big(\mathcal{M}_1(X_1), X\big). \tag{3.2}$$

We call these methods variant 1 and variant 2 respectively. The RDP bounds for variant 2 can be obtained with standard subsampling and composition result (e.g. Thm 4). We provide a new tailored analysis for this method in Section 3.3.

3.2 Extrapolating the DP-SGD Hyperparameters

When using DP-SGD, to transfer the optimal hyperparameter to a larger dataset, we use the heuristics also used by van der Veen et al. (2018): we scale the learning rate η with the dataset size. I.e., if we carry out the hyperparameter tuning using a subset of size m and find an optimal value η^*, we multiply η^* by n/m when transferring to the dataset of size n. Clipping constant C, the noise level σ and the subsampling ratio γ are kept constant in this transfer. This can be also motivated as follows. Consider T steps of the DP-SGD. With the above rules, the distribution of the noise that gets injected into the second model is approximately

$$\sum\nolimits_{j=1}^{T} Z_j \sim \mathcal{N}\big(0, \tfrac{T \cdot \left(\frac{n}{m}\eta^*\right)^2 \sigma^2 C^2}{(\gamma \cdot n)^2}\big) \sim \mathcal{N}\big(0, \tfrac{T \cdot {\eta^*}^2 \sigma^2 C^2}{(\gamma \cdot m)^2}\big)$$

which is the distribution of the noise added to the model trained with the subsample of size m.

Training of certain models benefits from use of adaptive optimizers such as Adam (Kingma & Ba, 2014) or RMSProp, e.g., due to sparse gradients. Then the above extrapolation rules for DP-SGD are not meaningful anymore. In our experiments, when training a neural network classifier for the IMDB dataset and using Adam with DP-SGD gradients, we found that keeping the value of learning rate fixed in the transfer to the larger dataset lead to better results. Recently Sander et al. (2022) proposed hyperparameter scaling laws using a similar guiding principle.

3.3 Privacy Analysis

We first consider the variant 2 given in equation 3.2. Using the RDP values given by Thm. 5 for $\mathcal{M}_1$ and a subsampling amplification result such as Thm. 4, we obtain RDP bounds for $(\mathcal{M}_1 \circ \text{subsample}_{\text{Poisson}(\gamma)})$. Using RDP bounds for $\mathcal{M}_2$ (e.g., DP-SGD) and composition results, we further get RDP bounds for the mechanism $\mathcal{M}$ of equation 3.2 where we use full data for the subsequent model.

Tailored RDP-Analysis. When we only use the rest of the data $X \backslash X_1$ for $\mathcal{M}_2$ in variant 1 (equation 3.1), we can get even tighter RDP bounds. The following theorem gives tailored RDP bounds for the mechanism of equation 3.1. Similarly to the analysis of Zhu & Wang (2019) for the Poisson subsampled Gaussian mechanism, we obtain RDP bounds using the RDP bounds of the non-subsampled mechanisms and by using binomial expansions (proof given in the Appendix C.2).

Theorem 6. *Let $X \in \mathcal{X}^n$ and $Y = X \cup \{x'\}$ for some $x' \in \mathcal{X}$. Let $\mathcal{M}(X)$ be the mechanism described in Section 3.1, such that X_1 is sampled with sampling ratio q, $0 \leq q \leq 1$. Let $\alpha > 1$. Denote by $\varepsilon_P(\alpha)$ and $\varepsilon_Q(\alpha)$ the RDP-values of mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. We have that*

$$\begin{aligned} D_\alpha\big(\mathcal{M}(Y) || \mathcal{M}(X)\big) \leq \frac{1}{\alpha - 1} \log \bigg(& q^\alpha \cdot \exp\big((\alpha-1)\varepsilon_P(\alpha)\big) + (1-q)^\alpha \cdot \exp\big((\alpha-1)\varepsilon_Q(\alpha)\big) \\ & + \sum_{j=1}^{\alpha-1} \binom{\alpha}{j} \cdot q^{\alpha-j} \cdot (1-q)^j \cdot \exp\big((\alpha-j-1)\varepsilon_P(\alpha-j)\big) \exp\big((j-1)\varepsilon_Q(j)\big) \bigg) \end{aligned} \tag{3.3}$$

and

$$\begin{aligned} D_\alpha\big(\mathcal{M}(X)||\mathcal{M}(Y)\big) \leq \frac{1}{\alpha-1}\log\Bigg(& (1-q)^{\alpha-1}\cdot\exp\big((\alpha-1)\varepsilon_Q(\alpha)\\ & + \sum_{j=1}^{\alpha-1}\binom{\alpha-1}{j}\cdot q^j\cdot(1-q)^{\alpha-1-j}\cdot\exp\big(j\cdot\varepsilon_P(j+1)\big)\cdot\exp\big((\alpha-j-1)\varepsilon_Q(\alpha-j)\big)\Bigg). \end{aligned} \tag{3.4}$$

***Remark* 7.** *We can initialize the subsequent model training $\mathcal{M}_2$ using the model θ_1. This adaptivity is included in all the RDP analyses we consider.*

3.4 Computational Savings

The expected number of required gradient evaluations for our approach is bounded by $(\mu\cdot q\cdot n + n)\cdot \mathrm{epochs}$, whereas the baseline requires in expectation $\mu\cdot n\cdot \mathrm{epochs}$ evaluations. For example, in our experiments with $\mu = 10$ and $q = 0.1$, the baseline requires $\frac{\mu}{\mu\cdot q+1}\approx 5$ times more gradient evaluations than our method.

4 Dealing with DP-SGD Hyperparameters that Affect the DP Guarantees

Theorem 5 gives RDP-parameters of order α for the tuning algorithm, assuming the underlying candidate picking algorithm is $\big(\alpha,\varepsilon(\alpha)\big)$-RDP. In case of DP-SGD, if we are tuning the learning rate η or clipping constant C, and fix rest of the hyperparameters, these $\big(\alpha,\varepsilon(\alpha)\big)$-RDP bounds are fixed for all the hyperparameter candidates. However, if we are tuning hyperparameters that affect the DP guarantees, i.e., the subsampling ratio γ, the noise level σ or the length of the training T, it is less straightforward to determine suitable uniform $\varepsilon(\alpha)$-upper bounds. As is common practice, we consider a grid Λ of α-orders for RDP bookkeeping (e.g. integer values of α's).

4.1 Grid Search with Randomization

To deal with this problem, we first set an approximative DP target value (ε,δ) that we use to adjust some of the hyperparameters. For example, if we are tuning the subsampling ratio γ and noise level σ, we can, for each choice of (γ,σ), adjust the length of the training T so that the resulting training iteration is at most (ε,δ)-DP. Vice versa, we may tune γ and T, and take minimal value of σ such that the resulting training iteration is at most (ε,δ)-DP.

More specifically,we first fix $\varepsilon,\delta>0$ which represent the target approximative DP bound for each candidate model. Denote by $\varepsilon(T,\delta,\gamma,\sigma)$ the ε-value of the subsampled Gaussian mechanism with parameter values γ, σ and T and for fixed δ. To each value of (γ,σ), we attach a number of iterations $T_{\gamma,\sigma}$ such that it is the largest number with which the resulting composition is (ε,δ)-DP:

$$T_{\gamma,\sigma} = \max\{T\in\mathbb{N} \,:\, \varepsilon(T,\delta,\gamma,\sigma)\leq\varepsilon\}.$$

As the RDP values increase monotonously w.r.t. number of compositions, it is straightforward to find $T_{\gamma,\sigma}$, e.g., using the bisection method.

Alternatively, we could fix a few values of T, and to each combination of (γ,T), attach the smallest σ (denoted $\sigma_{\gamma,T}$) such that the target (ε,δ)-guarantee holds. We prefer this option in our experiments to suit our computational constraints. By the data-processing inequality, the privacy parameters decrease monotonously w.r.t. σ, so that again, e.g., the bisection method can be used to find σ.

We consider a finite grid Γ of possible hyperparameter values t (e.g., $t=(\gamma,\sigma,T)$, where T is adjusted to γ and σ). Then, for all $t\in\Gamma$, we compute the corresponding RDP value $\varepsilon_t(\alpha)$ for each $\alpha\in\Lambda$. Finally, for each $\alpha\in\Lambda$, we set

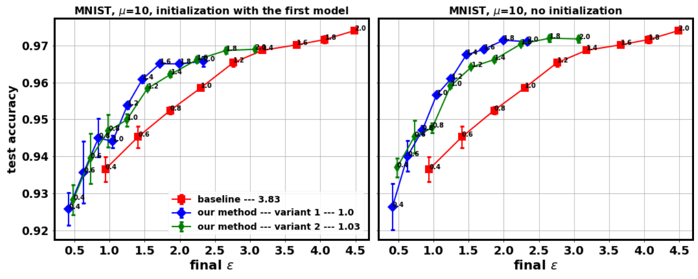
$$\varepsilon(\alpha)=\max_{t\in\Gamma}\varepsilon_t(\alpha).$$

Then, since for each random draw of t, the DP-SGD trained candidate model is $\varepsilon(\alpha)$-RDP, by Lemma A.1 given in Appendix, the candidate picking algorithm Q is also $\varepsilon(\alpha)$-RDP. This approach is used in the experiments of Appendix 5.2, where we jointly tune σ, γ and η.

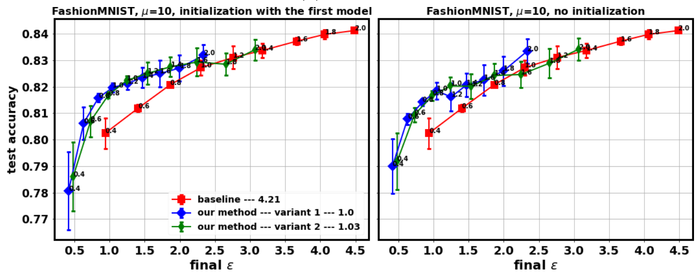
5 Experiments

5.1 Learning Rate Tuning

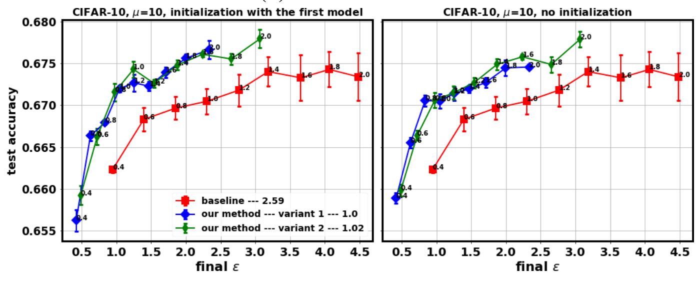
Figure 1 describes the results for learning rate tuning using our methods and the baseline method by Papernot & Steinke (2022) when training neural networks on standard benchmark datasets for classification (MNIST (LeCun et al., 1998), FashionMNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky & Hinton, 2009) and IMDB (Maas et al., 2011)). Full details of the experiments are are given in Appendix B. We set here $\mu = 10$ (the expected number of candidate models the tuning algorithm trains). We notice that both variants of our method provide better privacy-utility trade-off and have a lower computational cost than the baseline method.



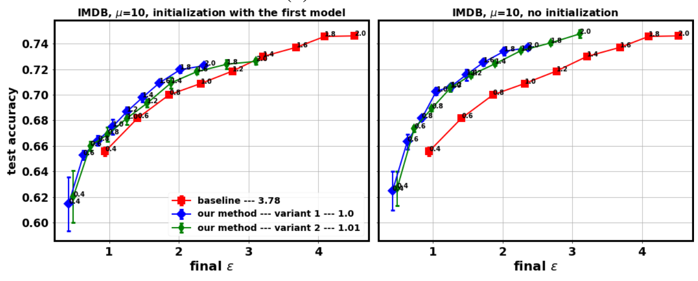
(a) MNIST



(b) FashionMNIST



(c) CIFAR-10

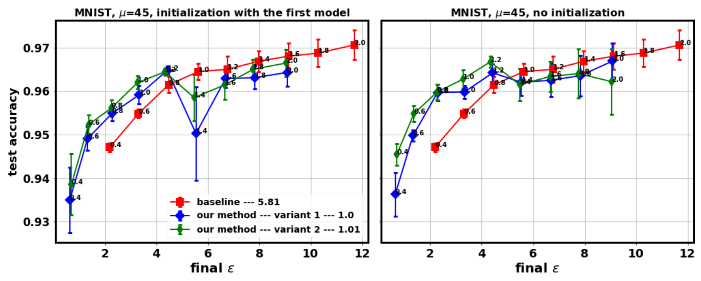


(d) IMDB

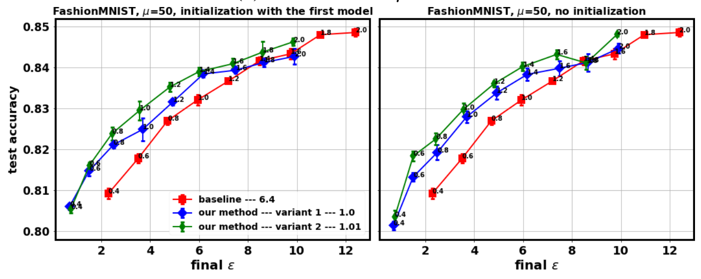
Figure 1: Tuning only the learning rate. Test accuracies averaged across 5 independent runs. The numbers in the legend refer to the mean training timings scaled with respect to variant 1 (blue curves). The error bars denote the std. error of the mean. Each curve contains 9 points, one for each target $\varepsilon \in \{0.4, 0.6, .., 2.0\}$ (for $\delta = 10^{-5}$) used to adjust σ.

5.2 Tuning All Hyperaparameters

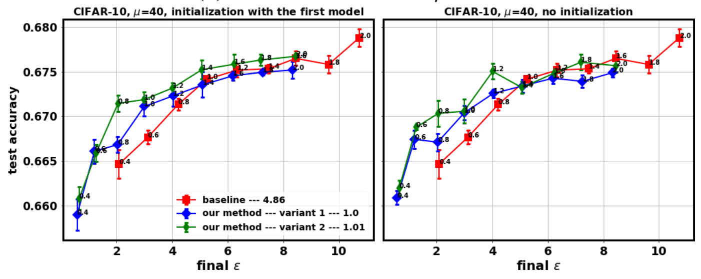
Next we jointly optimize the noise level σ, the batch size, and the learning rate η. The remaining setup is the same as in the previous experiment. The hyperparameter candidates are listed in Table 2 (Section B.2 in Appendix). Figure 2 shows the same quantities as Figure 1, however higher values of μ are used to accommodate to increased hyperaparameter spaces. Our method is more accurate compared to the baseline for low target ε-values and there is slight degradation in accuracy for higher target ε's. We also note that for final ε-value 2.0, our test accuracies are at least as good as those of Abadi et al. (2016) in case of MNIST (96% in Abadi et al., 2016) for and approximately the same in case of CIFAR-10 (67% in Abadi et al., 2016), although the results of Abadi et al. (2016) do not include the DP cost of hyperparameter tuning.



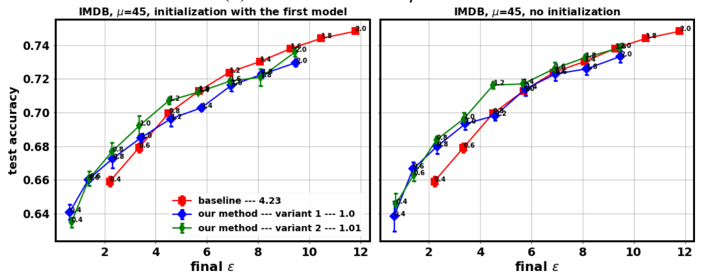
(a) MNIST with $\mu = 45$



(b) FashionMNIST with $\mu = 50$



(c) CIFAR-10 with $\mu = 40$



(d) IMDB with $\mu = 45$

Figure 2: Tuning of subsampling ratio, training length, and learning rate. Test accuracies averaged across 5 independent runs. The numbers in the legend refer to the mean training timings scaled with respect to variant 1 (blue curves). The error bars denote the std. error of the mean. Each curve contains 9 points, one for each target $\varepsilon \in \{0.4, 0.6, .., 2.0\}$.

References

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*, pp. 308–318, 2016.

Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pp. 464–473. IEEE, 2014.

Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pp. 635–658. Springer, 2016.

Clément Canonne, Gautam Kamath, and Thomas Steinke. The discrete gaussian for differential privacy. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Jinshuo Dong, Aaron Roth, Weijie J Su, et al. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B*, 84(1):3–37, 2022.

Cynthia Dwork. Differential privacy. In *Proc. 33rd Int. Colloq. on Automata, Languages and Prog. (ICALP 2006), Part II*, pp. 1–12, 2006.

Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *International Conference on Artificial Intelligence and Statistics*, pp. 528–536. PMLR, 2017.

Antti Koskela and Tejas Kulkarni. Practical differentially private hyperparameter tuning with subsampling. *arXiv preprint arXiv:2301.11989*, 2023.

Antti Koskela, Joonas Jälkö, and Antti Honkela. Computing tight differential privacy guarantees using FFT. In *International Conference on Artificial Intelligence and Statistics*, pp. 2560–2569. PMLR, 2020.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

Friedrich Liese and Igor Vajda. On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory*, 52(10):4394–4412, 2006.

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

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30, 2009.

Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pp. 263–275. IEEE, 2017.

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

Shubhankar Mohapatra, Sajin Sasy, Xi He, Gautam Kamath, and Om Thakkar. The role of adaptive optimizers for honest private hyperparameter selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7806–7813, 2022.

Nicolas Papernot and Thomas Steinke. Hyperparameter tuning with Rényi differential privacy. In *International Conference on Learning Representations*, 2022.

Tom Sander, Pierre Stock, and Alexandre Sablayrolles. Tan without a burn: Scaling laws of dp-sgd. *arXiv preprint arXiv:2210.03403*, 2022.

Josh Smith, Hassan Jameel Asghar, Gianpaolo Gioiosa, Sirine Mrabet, Serge Gaspers, and Paul Tyler. Making the most of parallel composition in differential privacy. *Proceedings on Privacy Enhancing Technologies*, 1:253–273, 2022.

Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, pp. 245–248. IEEE, 2013.

Thomas Steinke. Composition of differential privacy & privacy amplification by subsampling. *arXiv preprint arXiv:2210.00597*, 2022.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Abhradeep Guha Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Conference on Learning Theory*, pp. 819–850. PMLR, 2013.

Koen Lennart van der Veen, Ruben Seggers, Peter Bloem, and Giorgio Patrini. Three tools for practical differential privacy. *NeurIPS 2018 Privacy Preserving Machine Learning workshop, arXiv:1812.02890*, 2018.

Tim Van Erven and Peter Harremos. Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. Technical report, 2017.

Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. Opacus: User-friendly differential privacy library in pytorch. In *NeurIPS 2021 Workshop Privacy in Machine Learning*, 2021.

Yuqing Zhu and Yu-Xiang Wang. Poisson subsampled Rényi differential privacy. In *International Conference on Machine Learning*, pp. 7634–7642, 2019.

Yuqing Zhu, Jinshuo Dong, and Yu-Xiang Wang. Optimal accounting of differential privacy via characteristic function. In *International Conference on Artificial Intelligence and Statistics*, pp. 4782–4817. PMLR, 2022.

APPENDIX

A DEALING WITH HYPERPARAMETERS THAT AFFECT THE DP GUARANTEES

A.1 RANDOM SEARCH

Here, we assume we are given some distributions of the hyperparameter candidates and the algorithm Q draws hyperparameters using them. In order to adjust the number of iterations for each candidate, we take a α-line as an RDP upper bound. More specifically, we require that the candidate models are $\big(\alpha, c \cdot \alpha\big)$-RDP for some $c > 0$ and for all $\alpha \in \Lambda$. Then the number of iterations $T_{\gamma,\sigma}$ for each draw of (γ, σ) is the maximum number of iterations for which the $\big(\alpha, c \cdot \alpha\big)$-RDP bound holds, i.e.,

$$T_{\gamma,\sigma} = \max\{T \in \mathbb{N} \;:\; T \cdot \varepsilon_{\gamma,\sigma}(\alpha) \leq c \cdot \alpha \text{ for all } \alpha \in \Lambda\}.$$

Similarly, we can find the minimal σ based on T and γ such that the mechanism is $\big(\alpha, c \cdot \alpha\big)$-RDP for all $\alpha \in \Lambda$.

Again, by Lemma A.1 below, the candidate picking algorithm Q is then $c \cdot \alpha$-RDP and we may use Thm. 5 to obtain RDP bounds for the tuning algorithm.

The following result gives a rigorous proof for RDP bounds for algorithms presented in Section 4.1 and Section A.1. Let's call them Algorithm 1 and 2.

A.2 RDP ANALYSIS OF ALGORITHMS 1 AND 2

Lemma A.1. *Denote by C the random variable of which outcomes are the hyperparameter candidates (drawing either randomly from a grid or from given distributions). Consider an algorithm Q, that first randomly picks hyperparameters $t \sim C$, then runs a randomised mechanism $\mathcal{M}(t, X)$. Suppose $\mathcal{M}(t, X)$ is $\big(\alpha, \varepsilon(\alpha)\big)$-RDP for all t. Then, Q is $\big(\alpha, \varepsilon(\alpha)\big)$-RDP.*

Proof. Suppose the hyperparameters t are outcomes of a random variable C. Let X and Y be neighbouring datasets. Then, if $p(t, s)$ and $q(t, s)$ (as functions of s) give the density functions of $\mathcal{M}(t, X)$ and $\mathcal{M}(t, Y)$, respectively, we have that

$$Q(X) \sim \mathbb{E}_{t\sim C}\, p(t, s) \quad \text{and} \quad Q(Y) \sim \mathbb{E}_{t\sim C}\, q(t, s).$$

Since for any distributions p and q, and for any $\alpha > 1$,

$$\exp\big((\alpha - 1) D_\alpha(p||q)\big) = \int \left(\frac{p(t)}{q(t)}\right)^\alpha q(t)\, \mathrm{d}t$$

gives an f-divergence (for $f(x) = x^\alpha$), it is also jointly convex w.r.t. p and q (Liese & Vajda, 2006). Thus, using Jensen's inequality, we have

$$\begin{aligned}
\exp\big((\alpha - 1) D_\alpha\big(Q(X)||Q(Y)\big)\big) &= \int \left(\frac{\mathbb{E}_{t\sim C}\, p(t, s)}{\mathbb{E}_{t\sim C}\, q(t, s)}\right)^\alpha \cdot \mathbb{E}_{t\sim C}\, q(t, s)\, \mathrm{d}s \\
&\leq \mathbb{E}_{t\sim C} \int \left(\frac{p(t, s)}{q(t, s)}\right)^\alpha \cdot q(t, s)\, \mathrm{d}s \\
&= \mathbb{E}_{t\sim C} \exp\big((\alpha - 1) D_\alpha\big(\mathcal{M}(t, X)||\mathcal{M}(t, Y)\big)\big) \\
&\leq \mathbb{E}_{t\sim C} \exp\big((\alpha - 1)\varepsilon(\alpha)\big) \\
&= \exp\big((\alpha - 1)\varepsilon(\alpha)\big)
\end{aligned}$$

from which the claim follows. □

A.3 ADJUSTING THE PARAMETERS T AND σ FOR DP-SGD

We next discuss the reasons for the success of Algorithm 1 and 2. It is often a good approximation to say that the RDP-guarantees of the Poisson subsampled Gaussian mechanism are lines as functions of the RDP order α, i.e., that the guarantees are those a Gaussian mechanism with some sensitivity and noise level values. For example, Mironov et al. (Thm. 11, 2019) show that the Poisson subsampled Gaussian mechanism is $\big(\alpha, 2\gamma^2\alpha/\sigma^2\big)$-RDP when α is sufficiently small. Also, (Thm. 38, Steinke, 2022) show that if the underlying mechanism is ρ-zCDP, then the Poisson subsampled version with subsampling ratio γ is $\big(\alpha, 10\gamma^2\rho\alpha\big)$-RDP when α is sufficiently small. Notice that the Gaussian mechanism with L_2-sensitivity Δ and noise level σ is $(\Delta^2/2\sigma^2)$-zCDP (Bun & Steinke, 2016).

We numerically observe, that the larger the noise level σ and the smaller the subsampling ratio γ, the better the line approximation of the RDP-guarantees (see Figure 3).

In case the privacy guarantees (either (ε,δ)-DP or RDP) are approximately those of a Gaussian mechanisms with some sensitivity and noise level values, both of the approaches for tuning the hyperparameters γ, σ and T described in Section 4 would lead to very little slack. This is because for the Gaussian mechanism, both the RDP guarantees (Mironov, 2017) and (ε,δ)-DP guarantees (Dong et al., 2022) depend monotonously on the scaled parameter

$$\widetilde{\sigma} = \frac{\sigma}{\Delta \cdot \sqrt{T}}.$$

This means that if we adjust the training length T based on values of σ by having some target (δ,ε)-bound for the candidate model (Algorithm 1 of Section 4), the resulting RDP upper bounds of different candidates will not be far from each other (and similarly for adjusting σ based on value of T). Similarly, for Algorithm 2 of Section 4, when adjusting T based on values of σ, the RDP guarantees of all the candidate models would be close to the upper bound ($c \cdot \alpha$, $c > 0$), i.e., they would not be far from each other.

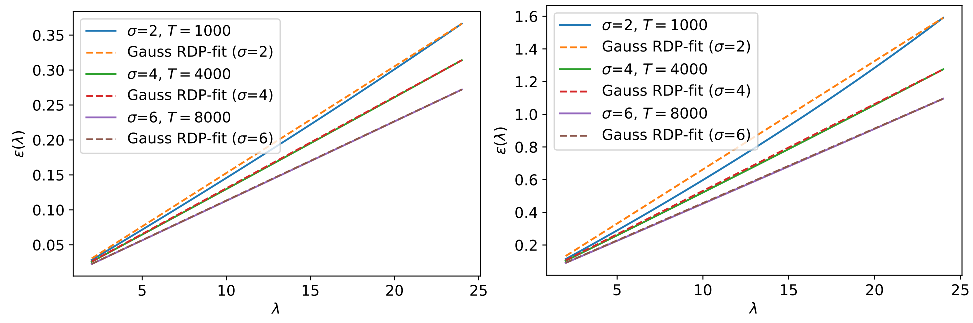


Figure 3: DP-SGD RDP curves for different values of noise level σ and number of compostions T. Left: $\gamma = 1/100$, right: $\gamma = 1/50$ and the corresponding lines with the smallest slope that give upper bounds for the RDP orders up to $\alpha = 24$.

B Full Description of Experiments

Quality Metric and Evaluation. In all of our experiments, we partition the available data into train and test sets and choose the best model based on the test accuracy. Training and test sets are disjoint, and the quality metric is usually a low sensitivity function. Therefore, even for a private test set, parallel composition (for an RDP bound of parallel compositions, we refer to Appendix D) can accommodate DP evaluation of a quality metric in the training budget itself. However, we assume that only the training dataset is private and the test data is public for simplicity. This assumption of test data set being public and the approach of making only two (train and test) partitions of available data instead of three (train, validation, and test) has been considered in many prior works (Mohapatra et al., 2022; Papernot & Steinke, 2022) to study the proposed method in isolation. This relaxation also allows us to take the best checkpoint over all epochs for each model.

Our plots show the test accuracy of the final model against the final approximate DP ε of the tuning process for several target ε's, varying from $\{0.4, 0.6, .., 2.0\}$. We fix $q = 0.1$ for our methods. We mention that in several cases smaller value of q would have lead to better privacy-accuracy tradeoff (see Appendix E for additional experiments), however we use the same value $q = 0.1$ in all experiments. The carry out RDP accounting using RDP orders $\{2, \ldots, 64\}$ and use $\delta = 10^{-5}$ in all experiments.

Methods. We consider the both variants of our proposed approach in our experiments. The final $\varepsilon(\alpha)$'s for the variant 2 and 1 are obtained by combining Thm. 5 with Thm. 4 and Thm. 6, respectively. We expect the second variant to be more accurate for a larger final ε compared to the first, since the final models obtained with the mechanism equation 3.2 use slightly more data than the one obtained with the mechanism equation 3.1. As discussed in Section 3.2, we scale the best learning rate obtained from the first model by the dataset size for training the final model. The method by Papernot & Steinke (2022) described in Thm. 5 is the baseline.

Datasets and Models. We carry out our experiments on the following standard benchmark datasets for classification: CIFAR-10 (Krizhevsky & Hinton, 2009), MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017) and IMDB (Maas et al., 2011). For MNIST and IMDB, we use the convolutional neural networks from the examples provided in the Opacus library Yousefpour et al. (2021). For FashionMNIST, we consider a simple feedforward 3-layer ReLU network with hidden layers of width 120. For CIFAR-10, we use a Resnet20 pretrained on CIFAR-100 (Krizhevsky & Hinton, 2009) dataset so that only the last fully connected layer is trained. We minimize the cross-entropy loss in all models. Following the Opacus example, we optimize with DP-Adam for IMDB dataset, but use DP-SGD for others.

Hyperparameters. For these datasets, in one of the experiments we tune only the learning rate, and in the other one the learning rate, batch size, and the number of epochs, while fixing the clipping constant C. The number of trainable parameters and the hyperparameter ranges used are provided in Table 2 (Appendix B.2). The numbers of epochs are chosen to suit our computational constraints. Following the procedure from Section 4.1, we compute the smallest σ satisfying a target (ε, δ) bound for each (γ, epoch) pair. The only purpose of target ε's is to facilitate the mapping from epochs to σ, and retain comparability across methods.

Initializing with the First Model. We are free to use the first model beyond hyperparameter extraction, since its privacy cost is already accounted for in the privacy analysis. Therefore, we use it to initialize the final model in both variants.

Implementation. For the implementation of DP-SGD, we use the Opacus library (Yousefpour et al., 2021). For scalability, we explore the hyperparameter spaces with Ray Tune (Liaw et al., 2018) on a multi-GPU cluster. We spell out additional details in the corresponding sections.

B.1 Tuning Learning Rate

The learning rate is among the most critical hyperaparameters, and thus we start with a learning rate optimization experiment. We fix the subsampling ratio γ and the number of epochs to the values given in Table 1 (Appendix B.2) for all models. For example, for $q = 0.1$, $\gamma = 0.0213$ on MNIST dataset, the Poisson subsampling of DP-SGD gives in expectation batch sizes of 128 and 1150 for our method and 1280 for the baseline method. The learning rate grid size is either 5 or 6, and we use $\mu = 10$, which is sufficiently large to include a good candidate with a large probability.

Plot Description. Figure 1 and Figure 2 plots the test accuracy against the final ε for all 4 datasets. The labels with 'variant 1' and 'variant 2' refer to the mechanism given in equation 3.1 and equation 3.2. The label 'baseline' refers to the method by (Papernot & Steinke, 2022) described in Thm. 5. The left panel for each dataset considers the case when the first model output by the tuning algorithm is used to initialize the subsequent model and the right panel shows the results when the first model is not used for initialization. Each plot contains 9 points for each method, one for each target ε ($\varepsilon \in \{0.4, 0.6, .., 2.0\}$) for each model training run.

B.2 Hyperparameter Tables

Tables 1 and 2 gives the hyperaparameters for the experiments of figures 1 and 2.

	MNIST	FashionMNIST	CIFAR-10	IMDB
$\gamma = \frac{B}{N}$	0.0213	0.0213	0.0256	0.0256
epochs	40	40	40	110

Table 1: Tuning η: rest of the hyperparameters fixed to these values.

	train/test set	parameters	C	B	learning rate	epochs
MNIST	60k/10k	$\sim$26k	1	$\{128, 256\}$	$\{10^{-i}\}_{i\in\{2,1.5,1,0.5,0\}}$	{10,20,30,40}
FashionMNIST	60k/10k	$\sim$109k	3	$\{128, 256\}$	$\{10^{-i}\}_{i\in\{2,1.5,1,0.5,0,-0.5\}}$	{10,20,30,40}
CIFAR-10	50k/10k	0.65k	3	$\{64, 128\}$	$\{10^{-i}\}_{i\in\{2,1.5,1,0.5,0,-0.5\}}$	{20,30,40}
IMDB	25k/25k	$\sim$464k	1	$\{64, 128\}$	{0.02,0.1,0.2,1,1.1}	{50,70,90,110}

Table 2: Tuning σ, η and T: datasets used and the corresponding hyperparameter grids.

C Proof of Theorem 6

C.1 Auxiliary Lemma

We will need the following inequality for the proof of Theorem 6.

Lemma C.1 (Lemma 35, Steinke 2022). *For all $p \in [0,1]$ and $x \in (0,\infty)$,*

$$\frac{1}{1-p+\frac{p}{x}} \leq 1-p+p\cdot x.$$

***Remark* C.2.** *An alternative proof for this result can be obtained using so called Bergström's inequality, which states that for all $x_k \in \mathbb{R}$, $a_k > 0$, $k \in [n]$,*

$$\frac{(x_1+\ldots+x_n)^2}{a_1+\ldots+a_n} \leq \frac{x_1^2}{a_1}+\ldots+\frac{x_n^2}{a_n}. \tag{C.1}$$

In particular, for $n=2$ and $a_1=\frac{q}{x}$, $a_2=1-q$, $x_1=q$, $x_2=1-q$, this gives

$$\frac{1}{1-q+\frac{q}{x}} \leq 1-q+q\cdot x.$$

As the following proof shows, the statement of of Theorem 6 could be generalized to the case of random k-way split ($k>2$) using the inequality of equation C.1.

C.2 Proof of Thm. 6

Proof. We first consider the case $D_\alpha\big(\mathcal{M}(Y)||\mathcal{M}(X)\big)$. Let $X \in \mathcal{X}^n$ and $x' \in \mathcal{X}$. Denote

$$\varepsilon_1(\alpha) = D_\alpha\big(\mathcal{M}(X \cup \{x'\})||\mathcal{M}(X)\big).$$

Looking at our approach which uses Poisson subsampling with subsampling ratio q to obtain the dataset X_1, and conditioning the output on the randomness in choosing X_1, we can write the mechanism as a mixture over all possible choices of X_1 as

$$\mathcal{M}(X) = \sum_{X_1} p_{X_1} \cdot \big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1)\big), \tag{C.2}$$

where p_{X_1} is the probability of sampling X_1. Since each data element is in X_1 with probability q, we can furthermore write $\mathcal{M}(X \cup \{x'\})$ as a mixture

$$\begin{aligned}\mathcal{M}(X \cup \{x'\}) = \sum_{X_1} p_{X_1} \cdot \Big(& q \cdot \big(\mathcal{M}_1(X_1 \cup \{x'\}), \mathcal{M}_2(\mathcal{M}_1(X_1 \cup \{x'\}), X\backslash X_1)\big) \\ &+ (1-q) \cdot \big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1 \cup \{x'\})\big)\Big).\end{aligned} \tag{C.3}$$

From the quasi-convexity of the Rényi divergence (Van Erven & Harremos, 2014) and the expressions equation C.2 and equation C.3, it follows that

$$\begin{aligned}D_\alpha\big(\mathcal{M}(X \cup \{x'\})||\mathcal{M}(X)\big) \leq \sup_{X_1} D_\alpha\big(& q \cdot \big(\mathcal{M}_1(X_1 \cup \{x'\}), \mathcal{M}_2(\mathcal{M}_1(X_1 \cup \{x'\}), X\backslash X_1)\big) \\ &+ (1-q) \cdot \big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1 \cup \{x'\})\big)||\big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1)\big)\big).\end{aligned} \tag{C.4}$$

Our aim is to express the right-hand side of equation C.4 in terms of RDP parameters of $\mathcal{M}_1$ and $\mathcal{M}_2$. To this end, take an arbitrary $X_1 \subset X$, and denote by

- $\widetilde{P}(t)$ the density function of $\mathcal{M}_1(X_1 \cup \{x'\})$,
- $P(t)$ the density function of $\mathcal{M}_1(X_1)$,
- $\widetilde{Q}(t,s)$ the density function of $\mathcal{M}_2(t, X\backslash X_1 \cup \{x'\})$ for auxiliary variable t (the output of $\mathcal{M}_1$),

- $Q(t,s)$ the density function of $\mathcal{M}_2(t, X\backslash X_1)$ for auxiliary variable t.

Then, we see that

$$\mathbb{P}\big(\big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1)\big) = (t,s)\big) = P(t)\cdot Q(t,s)$$

and similarly that

$$\begin{aligned}
&\mathbb{P}\big(q \cdot \big(\mathcal{M}_1(X_1 \cup \{x'\}), \mathcal{M}_2(\mathcal{M}_1(X_1 \cup \{x'\}), X\backslash X_1)\big) \\
&+ (1-q)\cdot \big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1 \cup \{x'\})\big) = (t,s)\big) \\
&= q \cdot \mathbb{P}\big(\big(\mathcal{M}_1(X_1 \cup \{x'\}), \mathcal{M}_2(\mathcal{M}_1(X_1 \cup \{x'\}), X\backslash X_1)\big) = (t,s)\big) \\
&\quad + (1-q)\cdot \mathbb{P}\big(\big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1 \cup \{x'\})\big) = (t,s)\big) \\
&= q \cdot \widetilde{P}(t)\cdot Q(t,s) + (1-q)\cdot P(t)\cdot \widetilde{Q}(t,s).
\end{aligned}$$

By the definition of the Rényi divergence, we have that

$$\begin{aligned}
&\exp\Bigg((\alpha-1)D_\alpha\big(q\cdot\big(\mathcal{M}_1(X_1 \cup \{x'\}), \mathcal{M}_2(\mathcal{M}_1(X_1 \cup \{x'\}), X\backslash X_1)\big) \\
&\qquad + (1-q)\cdot \big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1 \cup \{x'\})\big) || \big(\mathcal{M}_1(X_1), \mathcal{M}_2(\mathcal{M}_1(X_1), X\backslash X_1)\big)\big)\Bigg) \\
&= \int\int \left(\frac{q\cdot \widetilde{P}(t)\cdot Q(t,s) + (1-q)\cdot P(t)\cdot \widetilde{Q}(t,s)}{P(t)\cdot Q(t,s)}\right)^\alpha \cdot P(t)\cdot Q(t,s)\,\mathrm{d}t\,\mathrm{d}s.
\end{aligned} \tag{C.5}$$

which can be expanded as

$$\begin{aligned}
&\int\int \left(\frac{q\cdot \widetilde{P}(t)\cdot Q(t,s) + (1-q)\cdot P(t)\cdot \widetilde{Q}}{P(t)\cdot Q(t,s)}\right)^\alpha P(t)\cdot Q(t,s)\,\mathrm{d}t\,\mathrm{d}s \\
&= \int\int \left(q\cdot\frac{\widetilde{P}(t)}{P(t)} + (1-q)\cdot\frac{\widetilde{Q}(t,s)}{Q(t,s)}\right)^\alpha P(t)\cdot Q(t,s)\,\mathrm{d}t\,\mathrm{d}s \\
&= \int\int q^\alpha \left(\frac{\widetilde{P}(t)}{P(t)}\right)^\alpha P(t)\cdot Q(t,s)\,\mathrm{d}t\,\mathrm{d}s \\
&+ \int\int (1-q)^\alpha \left(\frac{\widetilde{Q}(t,s)}{Q(t,s)}\right)^\alpha P(t)\cdot Q(t,s)\,\mathrm{d}t\,\mathrm{d}s \\
&+ \int\int \alpha\cdot q^{\alpha-1}\cdot(1-q)\cdot\left(\frac{\widetilde{P}(t)}{P(t)}\right)^{\alpha-1} P(t)\cdot\widetilde{Q}(t,s)\,\mathrm{d}t\,\mathrm{d}s \\
&+ \int\int \alpha\cdot q\cdot(1-q)^{\alpha-1}\cdot\left(\frac{\widetilde{Q}(t,s)}{Q(t,s)}\right)^{\alpha-1} Q(t,s)\cdot\widetilde{P}(t)\,\mathrm{d}t\,\mathrm{d}s \\
&+ \int\int \sum_{j=2}^{\alpha-2}\binom{\alpha}{j}\cdot q^{\alpha-j}\cdot(1-q)^j\cdot\left[\left(\frac{\widetilde{P}(t)}{P(t)}\right)^{\alpha-j} P(t)\right]\left[\left(\frac{\widetilde{Q}(t,s)}{Q(t,s)}\right)^j Q(t,s)\right]\,\mathrm{d}t\,\mathrm{d}s.
\end{aligned} \tag{C.6}$$

We next bound five integrals on the right hand side of equation C.6. For the first two integrals, we use the RDP-bounds for $\mathcal{M}_1$ and $\mathcal{M}_2$ to obtain

$$\begin{aligned}
\int\int \left(\frac{\widetilde{P}(t)}{P(t)}\right)^\alpha P(t)Q(t,s)\,\mathrm{d}t\,\mathrm{d}s &= \int \left(\frac{\widetilde{P}(t)}{P(t)}\right)^\alpha P(t)\,\mathrm{d}t \\
&\leq \exp\big((\alpha-1)\varepsilon_P(\alpha)\big).
\end{aligned} \tag{C.7}$$

and

$$\begin{aligned} \int\int \left(\frac{\widetilde{Q}(t,s)}{Q(t,s)} \right)^{\alpha} Q(t,s)P(t)\, \mathrm{d}s\, \mathrm{d}t &\leq \int \exp\big((\alpha-1)\varepsilon_Q(\alpha) \big) P(t)\, \mathrm{d}t \\ &= \exp\big((\alpha-1)\varepsilon_Q(\alpha) \big), \end{aligned} \tag{C.8}$$

where ε_P and ε_Q give the RDP-parameters of order α for $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. The third and fourth integral can be bounded analogously. In the second inequality we have also used the fact that the RDP-parameters of $\mathcal{M}_2$ are independent of the auxiliary variable t. Similarly, for the third integral, we have

$$\begin{aligned} &\int\int \left[\left(\frac{\widetilde{P}(t)}{P(t)} \right)^{\alpha-j} P(t) \right] \left[\left(\frac{\widetilde{Q}(t,s)}{Q(t,s)} \right)^{j} Q(t,s) \right] \mathrm{d}s\, \mathrm{d}t \\ &\leq \int \left[\left(\frac{\widetilde{P}(t)}{P(t)} \right)^{\alpha-j} P(t) \right] \exp\big((j-1)\varepsilon_Q(j) \big)\, \mathrm{d}t \\ &\leq \exp\big((\alpha-j-1)\varepsilon_P(\alpha-j) \big) \cdot \exp\big((j-1)\varepsilon_Q(j) \big). \end{aligned} \tag{C.9}$$

Substituting equation C.7, equation C.8 (and similar expressions for the third and fourth integral) and equation C.9 to equation C.6, we get a bound for equation C.5. Since $X_1 \subset X$ was arbitrary, we arrive at the claim via equation C.4.

Next, we consider bounding $D_\alpha\big(\mathcal{M}(X)||\mathcal{M}(Y)\big)$. The proof goes similarly as the one for $D_\alpha\big(\mathcal{M}(Y)||\mathcal{M}(X)\big)$. Denote

$$\varepsilon_2(\alpha) = D_\alpha\big(\mathcal{M}(X)||\mathcal{M}(X \cup \{x'\})\big).$$

With the notation of proof of Thm. 6, we see that, instead of equation C.5, we need to bound

$$\begin{aligned} &\exp\big((\alpha-1)\varepsilon_2(\alpha) \big) \\ &= \int\int \left(\frac{P(t) \cdot Q(t,s)}{q \cdot \widetilde{P}(t) \cdot Q(t,s) + (1-q) \cdot P(t) \cdot \widetilde{Q}(t,s)} \right)^{\alpha} (q \cdot \widetilde{P}(t) \cdot Q(t,s) + (1-q) \cdot P(t) \cdot \widetilde{Q}(t,s))\, \mathrm{d}t\, \mathrm{d}s. \end{aligned}$$

In order to use here the series approach, we need to use Lemma C.1:

$$\begin{aligned} &\left(\frac{P \cdot Q}{q \cdot \widetilde{P} \cdot Q + (1-q) \cdot P \cdot \widetilde{Q}} \right)^{\alpha} (q \cdot \widetilde{P} \cdot Q + (1-q) \cdot P \cdot \widetilde{Q}) \\ &= \left(\frac{P \cdot Q}{q \cdot \widetilde{P} \cdot Q + (1-q) \cdot P \cdot \widetilde{Q}} \right)^{\alpha-1} \cdot P \cdot Q \\ &= \left(q \cdot \frac{\widetilde{P}}{P} + (1-q) \cdot \frac{\widetilde{Q}}{Q} \right)^{1-\alpha} \cdot P \cdot Q \\ &= \left(q \cdot \frac{\widetilde{P}}{P} \frac{Q}{\widetilde{Q}} + (1-q) \right)^{1-\alpha} \cdot \left(\frac{\widetilde{Q}}{Q} \right)^{1-\alpha} \cdot P \cdot Q \\ &= \left(q \cdot \frac{\widetilde{P}}{P} \frac{Q}{\widetilde{Q}} + (1-q) \right)^{1-\alpha} \cdot \left(\frac{Q}{\widetilde{Q}} \right)^{\alpha-1} \cdot P \cdot Q \\ &\leq \left(q \cdot \frac{P}{\widetilde{P}} \frac{\widetilde{Q}}{Q} + (1-q) \right)^{\alpha-1} \cdot \left(\frac{Q}{\widetilde{Q}} \right)^{\alpha-1} \cdot P \cdot Q \\ &= \left(q \cdot \frac{P}{\widetilde{P}} \frac{\widetilde{Q}}{Q} + (1-q) \right)^{\alpha-1} \cdot \left(\frac{Q}{\widetilde{Q}} \right)^{\alpha} \cdot P \cdot \widetilde{Q}, \end{aligned} \tag{C.10}$$

where in the inequality we have used Lemma C.1. Now we can expand $\left(q \cdot \frac{P}{\widetilde{\widetilde{P}}} \frac{\widetilde{Q}}{Q} + 1 - q\right)^{\alpha-1}$:

$$\begin{aligned}
&\left(1 - q + q \cdot \frac{P}{\widetilde{\widetilde{P}}} \frac{\widetilde{Q}}{Q}\right)^{\alpha-1} \cdot \left(\frac{Q}{\widetilde{\widetilde{Q}}}\right)^{\alpha} \cdot P \cdot \widetilde{Q} \\
&= \left(\sum_{j=0}^{\alpha-1} \binom{\alpha-1}{j} q^j \cdot (1-q)^{\alpha-1-j} \cdot \left(\frac{P}{\widetilde{\widetilde{P}}}\right)^j \left(\frac{\widetilde{Q}}{Q}\right)^j\right) \cdot \left(\frac{Q}{\widetilde{\widetilde{Q}}}\right)^{\alpha} \cdot P \cdot \widetilde{Q} \\
&= \left(\sum_{j=0}^{\alpha-1} \binom{\alpha-1}{j} q^j \cdot (1-q)^{\alpha-1-j} \cdot \left(\frac{P}{\widetilde{\widetilde{P}}}\right)^j \left(\frac{Q}{\widetilde{\widetilde{Q}}}\right)^{\alpha-j}\right) \cdot P \cdot \widetilde{Q} \\
&= \sum_{j=0}^{\alpha-1} \binom{\alpha-1}{j} q^j \cdot (1-q)^{\alpha-1-j} \cdot \left(\frac{P}{\widetilde{\widetilde{P}}}\right)^{j+1} \widetilde{P} \cdot \left(\frac{Q}{\widetilde{\widetilde{Q}}}\right)^{\alpha-j} \widetilde{Q}.
\end{aligned}$$

Then, we use the known $\varepsilon_P(\alpha)$ and $\varepsilon_Q(\alpha)$-values as in equation C.9 to arrive at the claim.

$\square$

D f-DIVERGENCE OF PARALLEL COMPOSITIONS

We first formulate the parallel composition result for general f-divergences (Lemma D.1). We then obtain the RDP bound for parallel compositions as a corollary (Cor. D.2).

Our Lemma D.1 below can be seen as an f-divergence version of the $(\varepsilon, 0)$-DP result given in (Thm. 4 McSherry, 2009). Corollary 2 by Smith et al. (2022) gives the corresponding result in terms of μ-Gaussian differential privacy (GDP), and it is a special case of our Lemma D.1 as μ-GDP equals the (ε, δ)-DP (i.e., the hockey-stick divergence) of the Gaussian mechanism with a certain noise scale (Cor. 1, Dong et al., 2022).

We define f-divergence for distributions on $\mathbb{R}^d$ as follows. Consider two probability densities P and Q defined on $\mathbb{R}^d$, such that if $Q(x) = 0$ then also $P(x) = 0$, and a convex function $f : [0, \infty) \to \mathbb{R}$. Then, an f-divergence (Liese & Vajda, 2006) is defined as

$$D_f(P||Q) = \int f\left(\frac{P(t)}{Q(t)}\right) Q(t)\ \mathrm{d}t.$$

In case the data is divided into disjoint shards and separate mechanisms are applied to each shard, the f-divergence upper bound for two neighbouring datasets can be obtained from the individual f-divergence upper bounds:

Lemma D.1. *Suppose a dataset $X \in \mathcal{X}^N$ is divided into k disjoint shards X_i, $i \in [k]$, and mechanisms $\mathcal{M}_i$, $i \in [k]$, are applied to the shards, respectively. Consider the mechanism*

$$\mathcal{M}(X) = \big(\mathcal{M}_1(X_1), \ldots, \mathcal{M}_k(X_k)\big).$$

Then, we have that

$$\max_{X \sim Y} D_f\big(\mathcal{M}(X)||\mathcal{M}(Y)\big) \leq \max_{i \in [k]} \max_{X \sim Y} D_f\big(\mathcal{M}_i(X)||\mathcal{M}_i(Y)\big).$$

Proof. Let X be divided into k shards as described above and suppose Y is a neighbouring dataset such that $X_1 \sim Y_1$ and $Y = \{Y_1, X_2, \ldots, X_k\}$.

Then, we see that

$$\begin{aligned}
&\frac{\mathbb{P}\big(\mathcal{M}(X) = (a_1, \ldots, a_k)\big)}{\mathbb{P}\big(\mathcal{M}(Y) = (a_1, \ldots, a_k)\big)} \\
&= \frac{\mathbb{P}\big(\mathcal{M}_1(X_1) = a_1\big) \cdot \mathbb{P}\big(\mathcal{M}_1(X_2, a_1) = a_2\big) \cdots \mathbb{P}\big(\mathcal{M}_k(X_k, a_1, \ldots, a_{k-1}) = a_k\big)}{\mathbb{P}\big(\mathcal{M}_1(Y_1) = a_1\big) \cdot \mathbb{P}\big(\mathcal{M}_1(X_2, a_1) = a_2\big) \cdots \mathbb{P}\big(\mathcal{M}_k(X_k, a_1, \ldots, a_{k-1}) = a_k\big)} \\
&= \frac{\mathbb{P}\big(\mathcal{M}_1(X_1) = a_1\big)}{\mathbb{P}\big(\mathcal{M}_1(Y_1) = a_1\big)}.
\end{aligned}$$

and furthermore, denoting $a = (a_1, \ldots, a_k)$,

$$\begin{aligned}
D_f\big(\mathcal{M}(X)||\mathcal{M}(Y))\big) &= \int f\left(\frac{\mathbb{P}\big(\mathcal{M}(X) = a\big)}{\mathbb{P}\big(\mathcal{M}(Y) = a\big)}\right) \mathbb{P}\big(\mathcal{M}(Y) = a\big)\ \mathrm{d}a \\
&= \int f\left(\frac{\mathbb{P}\big(\mathcal{M}_1(X_1) = (a_1)\big)}{\mathbb{P}\big(\mathcal{M}_1(Y_1) = (a_1)\big)}\right) \mathbb{P}\big(\mathcal{M}(Y) = a\big)\ \mathrm{d}a \\
&= \int f\left(\frac{\mathbb{P}\big(\mathcal{M}_1(X_1) = (a_1)\big)}{\mathbb{P}\big(\mathcal{M}_1(Y_1) = (a_1)\big)}\right) \mathbb{P}\big(\mathcal{M}(Y_1) = (a_1)\big)\ \mathrm{d}a_1 \\
&= D_f(\mathcal{M}_1(X_1)||\mathcal{M}_1(Y_1))\big).
\end{aligned}$$

Thus,

$$D_f(\mathcal{M}(X)||\mathcal{M}(Y)) = D_f(\mathcal{M}_1(X_1)||\mathcal{M}_1(Y_1)) \leq \max_{X \sim Y} D_f(\mathcal{M}_1(X)||\mathcal{M}_1(Y)).$$

Similarly, if $X_i \sim Y_i$, $i = 2, \ldots, k$ and

$$Y = \big(X_1, \ldots X_{i-1}, Y_i, X_{i+1}, \ldots, X_k\big),$$

we see that

$$D_f(\mathcal{M}(X)||\mathcal{M}(Y)) = D_f(\mathcal{M}_i(X_i)||\mathcal{M}_i(Y_i)) \leq \max_{X\sim Y} D_f(\mathcal{M}_i(X)||\mathcal{M}_i(Y)).$$

Thus, we have that

$$\max_{X\sim Y} D_f(\mathcal{M}(X)||\mathcal{M}(Y)) = \max_{i\in[k]} \max_{X\sim Y} D_f(\mathcal{M}_i(X)||\mathcal{M}_i(Y)).$$

□

Corollary D.2. *Suppose a dataset $X \in \mathcal{X}^N$ is divided into k disjoint shards X_i, $i \in [k]$, and mechanisms $\mathcal{M}_i$, $i \in [k]$, are applied to the shards, respectively. Consider the mechanism*

$$\mathcal{M}(X) = \big(\mathcal{M}_1(X_1), \ldots, \mathcal{M}_k(X_k)\big).$$

Suppose each $\mathcal{M}_i$ is $\big(\alpha, \varepsilon_i(\alpha)\big)$-RDP, respectively. Then, $\mathcal{M}$ is $\big(\alpha, \max_{i\in[k]} \varepsilon_i(\alpha)\big)$-RDP.

Proof. This follows from Lemma D.1 since

$$\exp\big((\alpha-1)D_\alpha(\mathcal{M}(X)||\mathcal{M}(Y))\big) = \int \left(\frac{\mathbb{P}\big(\mathcal{M}(X)=a\big)}{\mathbb{P}\big(\mathcal{M}(Y)=a\big)}\right)^{\alpha} \mathbb{P}\big(\mathcal{M}(Y)=a\big)\,\mathrm{d}a$$

is an f-divergence for $f(x) = x^\alpha$. Thus, by Lemma D.1 we have that

$$\max_{X\sim Y} \exp\big((\alpha-1)D_\alpha(\mathcal{M}(X)||\mathcal{M}(Y))\big) \leq \max_{i\in[k]} \max_{X\sim Y} \exp\big((\alpha-1)D_\alpha(\mathcal{M}_i(X)||\mathcal{M}_i(Y))\big)$$

from which it follows that

$$\max_{X\sim Y} D_\alpha(\mathcal{M}(X)||\mathcal{M}(Y)) \leq \max_{i\in[k]} \max_{X\sim Y} D_\alpha(\mathcal{M}_i(X)||\mathcal{M}_i(Y)) = \max_{i\in[k]} \varepsilon_i(\alpha).$$

□

E Additional Experiments

E.1 Comparison of (ε, δ)-Bounds

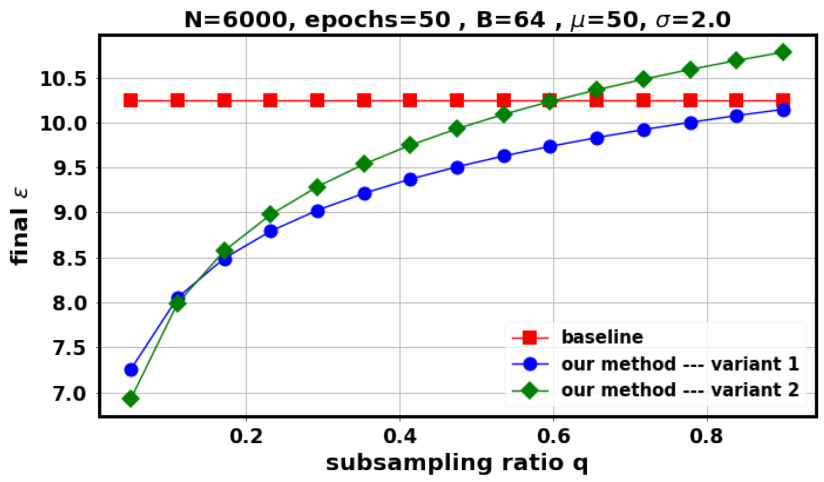
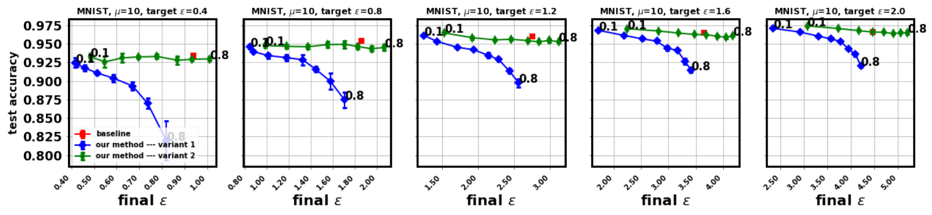


Figure 4: Final ε as a function of q for $\mu = 50$, when the base mechanism is the subsampled Gaussian mechanism with $\gamma = 64/6000$, number of epochs = 50 and $\sigma = 2.0$. The value of q varies from 0.05 to 0.9. Compared to the baseline, our method has considerably smaller ε's for small values of q. However, with $q \leq 0.1$, it may not provide good utility on small data sets (e.g., IMDB).

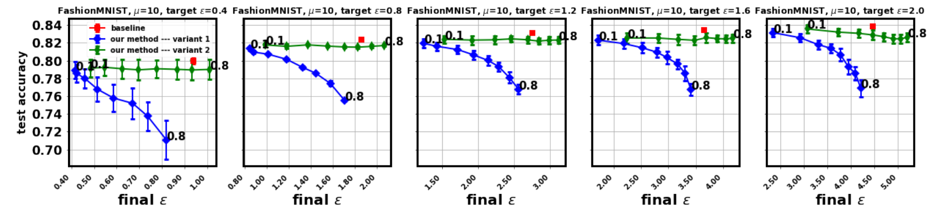
E.2 Vary q

When q increases, the hyperparameter tuning mechanism trains with a larger dataset which means also weaker final privacy guarantees. Additionally, the best learning rate from the first model is also scaled with a smaller factor in the final model. We perform for an experiment to compute the final test accuracies and ε-values as a function of q.

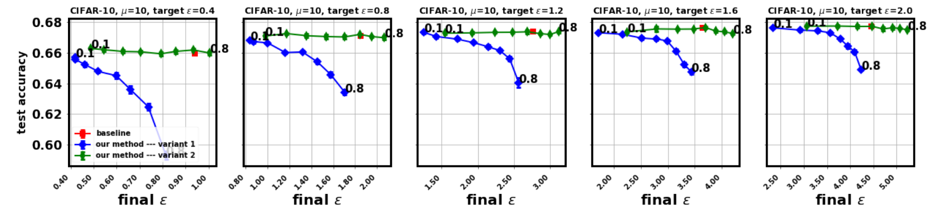
Figure 5 shows the test accuracy as a function of target ε for the base mechanism (DP-SGD), q, final ε. The value of q in each plot varies from 0.1 to 0.8. As expected, the accuracy for the variant 1 drops when q increases. The performance of variant 2 that trains the final model with the full dataset remains relatively steady for all models. In IMDB, we use DP-Adam to train the IMDB model and do not scale the best initial learning rate.



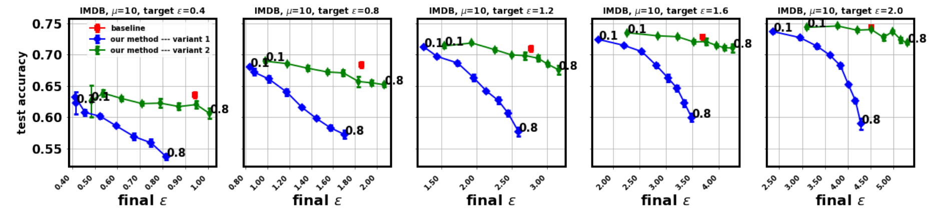
(a) MNIST with $\mu = 10$



(b) FashionMNIST with $\mu = 10$

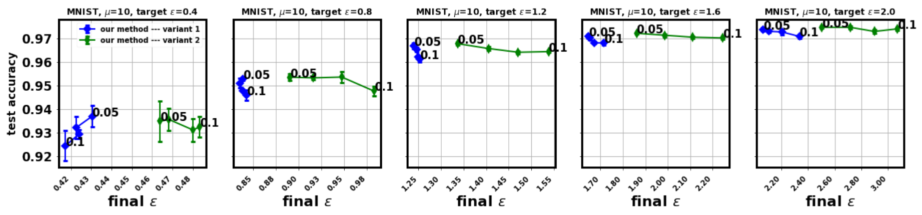


(c) CIFAR-10 with $\mu = 10$

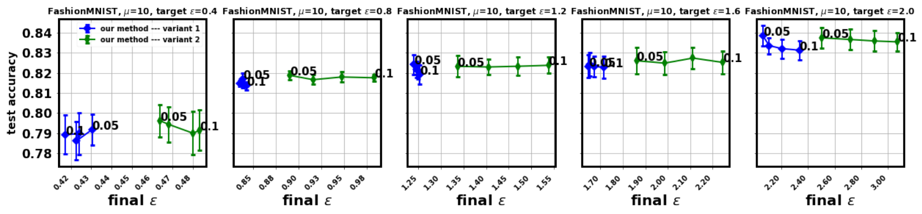


(d) IMDB with $\mu = 10$

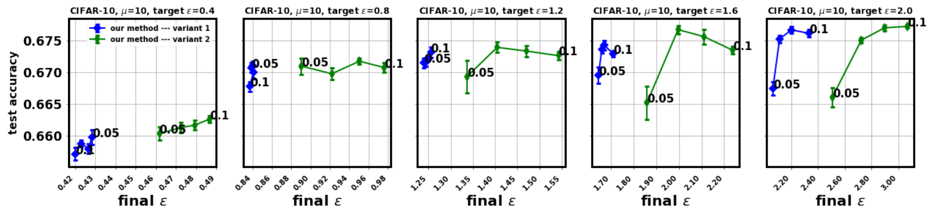
Figure 5: Tuning learning rate: We keep the batch size, the number of epochs fixed and only tune the learning rate with our method for various values of q'. Test accuracies are averaged across 5 independent runs. The error bars denotes the std. error of the mean. Each plot contains 8 points for both variants of our method, one for each $q \in \{0.1, 0.2, .., 0.8\}$ (left to right). We also add the baseline method for comparison. The $q = 0.1$ case generally yields the peak accuracy on all datasets.



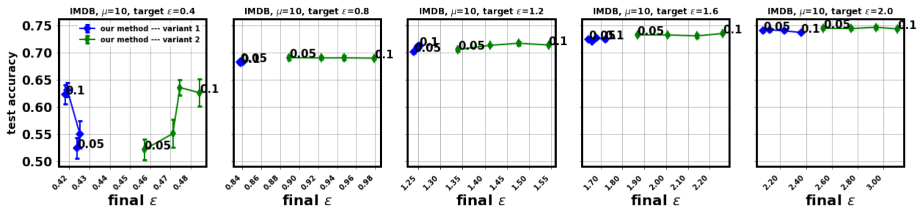
(a) MNIST with $\mu = 10$



(b) FashionMNIST with $\mu = 10$



(c) CIFAR-10 with $\mu = 10$



(d) IMDB with $\mu = 10$

Figure 6: Tuning learning rate for $q \leq 0.1$: We keep the batch size and the number of epochs fixed and only tune the learning rate with our methods for various values of $q \leq 0.1$. Test accuracies are averaged across 5 independent runs. The error bars denotes the std. error of the mean. Each plot contains 4 points for both variants, i.e. when $q \in \{0.05, 0.067, 0.083, 0.1\}$.