Fine-Tuning Large Language Models with User-Level Differential Privacy

Zachary Charles [1] **Arun Ganesh** [1] **Ryan McKenna** [1] **H. Brendan McMahan** [1] **Nicole Mitchell** [1]
Krishna Pillutla [2] **Keith Rush** [1]

Abstract

We investigate practical and scalable algorithms for training large language models (LLMs) with user-level differential privacy (ULDP). We study variants of DP-SGD that use example-level sampling (ELS) and user-level sampling (ULS). We derive a novel ULDP accountant that computes provably tight privacy guarantees for ELS, and use it to show that while ELS outperforms ULS in specific settings, ULS performs better when users have diverse collections of examples. We validate our findings in realistic LLM fine-tuning tasks under fixed compute budgets. Our results show that ULS is significantly better when (1) strong privacy guarantees are required, or (2) the compute budget is large. Our focus on LLM-compatible training algorithms allows us to scale to models with hundreds of millions of parameters and datasets with hundreds of thousands of users.

1. Introduction

Fully realizing the promise of large language models (LLMs) in a variety of domains may require fine-tuning on domain-specific data (Scao & Rush, 2021; Lester et al., 2021; Bhatia et al., 2023). In-domain data of users can be highly sensitive (Chen et al., 2019; Xi et al., 2023; Xu et al., 2023b), and without safeguards, using such data comes with major privacy risks, especially since LLMs can memorize and leak their training data (Carlini et al., 2021; 2023).

Differential privacy (DP) (Dwork et al., 2006) can mitigate such privacy risks: it gives rigorous guarantees on privacy leakage that can eliminate data leakage from LLMs (Carlini et al., 2019). DP is often used to protect individual examples (*example-level* DP). However, when training on user data, each user may contribute multiple correlated examples. Thus, example-level DP can fail to protect user-level privacy (Song & Shmatikov, 2019; Kandpal et al., 2023). We therefore study practical approaches to training LLMs with user-level DP (ULDP).

Many prior works on learning with ULDP are theoretical (Levy et al., 2021; Bassily & Sun, 2023; Ghazi et al., 2023a; Asi & Liu, 2024), and rely on subroutines such as outlier removal. Scaling such algorithms to LLM training across clusters of accelerators remains challenging. Empirical work on ULDP has largely focused on federated learning (McMahan et al., 2018; Wei et al., 2021; Xu et al., 2023b) for training small models on edge devices. By contrast, developing efficient ULDP training methods for LLMs requires vastly different system considerations.

Setting. We focus on two scalable variants of DP-SGD: DP-SGD-ELS (ELS), which applies DP-SGD with example-level sampling and gradient clipping to pooled user data (with each user contributing at most G_{ELS}); and DP-SGD-ULS (ULS), which applies DP-SGD to user-level gradients averaged over G_{ULS} examples per sampled user. Following LLM scaling law principles (Kaplan et al., 2020), we compare these methods under fixed compute budgets.

Contributions. We provide theoretical insights into LLM training with ULDP. We develop a novel, provably tight DP accountant for ELS under ULDP, significantly outperforming generic reductions. We show that while ELS may be better in specific cases, ULS excels when user gradients are diverse. We validate our findings through extensive experiments, including a mean estimation task and realistic LLM fine-tuning tasks. Our focus on scalable algorithms allows us to train models with hundreds of millions of parameters on datasets with hundreds of thousands of users.

2. Algorithms and Privacy Accounting

To define (ε, δ)-DP, we use the hockey-stick divergence H_α between distributions P, Q and its symmetrization H_α^{sym}:

$$H_\alpha(P, Q) := \max_S \{P(S) - \alpha Q(S)\},$$

$$H_\alpha^{\text{sym}}(P, Q) = \max\{H_\alpha(P, Q), H_\alpha(Q, P)\}.$$

*Equal contribution [1]Google Research [2]IIT Madras, Chennai India. Correspondence to: Zachary Charles <zachcharles@google.com>.

Work presented at TF2M workshop at ICML 2024, Vienna, Austria. PMLR 235, 2024.

Algorithm 1 DP-SGD-ELS

Additional Inputs: group size G_{ELS}, noise multiplier σ_{ELS}, example sampling probability p

$D_{\text{sub}} = \emptyset$ {*Limit user contributions*}

for each user $u \in [N]$ **do**

  Sample $S_u \subseteq D_u$ of size $|S_u| \leq G_{\text{ELS}}$

  $D_{\text{sub}} = D_{\text{sub}} \sqcup S_u$

end for

$B = p|D_{\text{sub}}|$ {*Expected batch size*}

for $t = 0, 1, \ldots, T-1$ **do**

  ▷ *Include each* ***example*** *w.p. p*

  Sample a batch of examples $S^t \subseteq D_{\text{sub}}$

  ▷ *Clip & noise* ***per-example*** *gradients*

  $g^t_{\text{sum}} = \sum_{z \in S^t} \text{clip}(\nabla f(\theta^t, z), C)$

  $g^t = \frac{1}{B}\left(g^t_{\text{sum}} + \mathcal{N}(0, C^2\sigma^2_{\text{ELS}} I_d)\right)$

  $\theta^{t+1} = \theta^t - \eta g^t$

end for

Algorithm 2 DP-SGD-ULS

Additional Inputs: group size G_{ULS}, noise multiplier σ_{ULS}, user sampling probability q

$M = qN$ {*Expected user cohort size*}

for $t = 0, 1, \ldots, T-1$ **do**

  ▷ *Include each* ***user*** *w.p. q*

  Sample users $U^t \subseteq [N]$

  for each **user** $u \in U^t$ **do**

    Sample $D^t_u \subseteq D_u$ of size $|D^t_u| \leq G_{\text{ULS}}$

    $g^t_u = \frac{1}{|D^t_u|} \sum_{z \in D^t_u} \nabla f(\theta^t, z)$

  end for

  ▷ *Clip & noise* ***per-user*** *gradients*

  $g^t_{\text{sum}} = \sum_{u \in U^t} \text{clip}(g^t_u, C)$

  $g^t = \frac{1}{M}\left(g^t_{\text{sum}} + \mathcal{N}(0, C^2\sigma^2_{\text{ULS}} I_d)\right)$

  $\theta^{t+1} = \theta^t - \eta g^t$

end for

Let $\mathcal{M}$ be a randomized algorithm that takes in a dataset D and returns a distribution $\mathcal{M}(D)$. We say $\mathcal{M}$ satisfies (ε, δ)-DP under an adjacency relation "$\sim$" if

$$\sup_{D \sim D'} H^{\text{sym}}_{e^\varepsilon}(\mathcal{M}(D), \mathcal{M}(D')) \leq \delta. \tag{1}$$

We study *user-partitioned* datasets. Let $\mathcal{U}$ be a set of *users*. Each $u \in \mathcal{U}$ is associated to a non-empty, finite multiset D_u. A user-partitioned dataset is a tuple (D, U) where $U \subseteq \mathcal{U}$, $|U| < \infty$, and $D = \sqcup_{u \in U} D_u$ is the multiset sum of the user datasets. We define $(D, U) \sim_{\text{user}} (D', U')$ if $U' = U \cup \{u\}$ or $U' = U \backslash \{u\}$ for some u. We say that $\mathcal{M}$ satisfies (ε, δ)-ULDP if $\mathcal{M}$ satisfies (1) w.r.t. $\sim_{\text{user}}$.

Algorithms. We obtain ULDP guarantees with two generalizations of DP-SGD. The first, DP-SGD with example-level sampling (DP-SGD-ELS, abbreviated ELS) forms a subdataset D_{sub} to which each user contributes at most G_{ELS} examples and applies DP-SGD to D_{sub}, with example-level sampling and gradient clipping. The second, DP-SGD with user-level sampling (DP-SGD-ULS, abbreviated ULS), applies DP-SGD to user-level gradients averaged over G_{ULS} examples per sampled user, We present ELS and ULS in Algorithms 1 and 2. They share an initial model θ_0, loss function $f(\theta, z)$, user-level dataset $D = \sqcup_{u=1}^N D_u$, learning rate η, clip norm C, and number of steps T. While we present an SGD model update in both, any first-order optimization technique can be applied.

ULDP accountants. Prior state-of-the-art accounting for Algorithm 1 uses black-box user-to-example reductions based on group privacy (Vadhan, 2017). As a result, prior work (e.g. Levy et al., 2021) suggests that Algorithm 1 is worse than Algorithm 2 as its formal ε grows quickly with G_{ELS}. We instead tailor the reduction to DP-SGD. By leveraging the Mixture-of-Gaussians (MoG) mechanism (Choquette-Choo et al., 2023), we derive the optimal ULDP accounting for ELS.

Theorem 1 (Informal version of Thm. 3)**.** *$\forall \varepsilon \geq 0$, Alg. 1 satisfies $(\varepsilon, \delta(\varepsilon))$-ULDP with*

$$\delta(\varepsilon) = H^{\text{sym}}_{e^\varepsilon}(\mathcal{N}(0, \sigma^2_{ELS})^{\otimes T}, \mathcal{N}(\mathcal{B}, \sigma^2_{ELS})^{\otimes T}), \tag{2}$$

where $\mathcal{B} = \text{Binom}(G_{ELS}, p)$ and $P^{\otimes T}$ denotes the product of a distribution P with itself T times. For any $\delta' < \delta(\varepsilon)$, there is a setting where Alg. 1 does not satisfy (ε, δ')-ULDP.

The $\delta(\varepsilon)$ function is easily computable using open-source DP accounting libraries (see Appendix B.1). We use this in Fig. 1 to compare the ε computed by Theorem 1 to the ε given by black-box reductions. **Our accountant gives near-linear scaling** of ε in G_{ELS} where generic user-to-example reductions give **exponential scaling**.

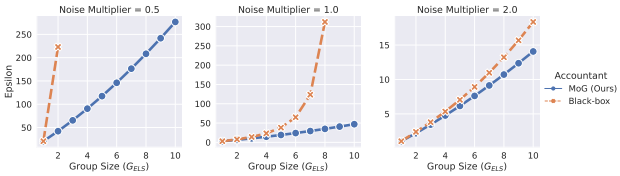


Figure 1: Upper bound on ε for ELS using our Mixture-of-Gaussians (MoG) accountant and prior state-of-the-art black-box accounting (Vadhan, 2017). We set $T = 2000$, $p = 10^{-2}$, $\delta = 10^{-6}$, and vary σ_{ELS} and G_{ELS}. The black-box accountant diverged for G_{ELS} sufficiently large.

Since Alg. 2 applies a straightforward subsampled Gaussian mechanism, its formal privacy guarantees are well understood and easy to compute (e.g. Koskela et al., 2021). To directly compare ELS and ULS, we will use the following result, which follows from Theorem 1 by setting $G_{\text{ELS}} = 1$.

Theorem 2. *$\forall \varepsilon \geq 0$, Alg. 2 satisfies $(\varepsilon, \delta(\varepsilon))$-ULDP with*

$$\delta(\varepsilon) = H_{e^\varepsilon}^{\mathrm{sym}}(\mathcal{N}(0, \sigma_{ULS}^2)^{\otimes T}, \mathcal{N}(\mathrm{Ber}(q), \sigma_{ULS}^2)^{\otimes T}), \quad (3)$$

where Ber *is Bernoulli. For any $\delta' < \delta(\varepsilon)$, there is a setting where Alg. 2 does not satisfy (ε, δ') ULDP.*

Compute budgets. Algs. 1 and 2 vary greatly in how they process data. Motivated by work on LLM scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022), we focus on their performance under *fixed compute budgets*. For simplicity, we assume computation cost equals the number of gradient computations. We do not consider gradient averaging, clipping, and noise generation, or communication. The expected per-iteration compute of ELS is its expected batch size B, while that of ULS equals $G_{\text{ULS}}M$, where $M = qN$ is the expected user cohort size (see Algorithm 2).

3. Comparing Noise Variances

We now compare ELS and ULS under Lipschitz losses via their noise variance per example gradient. For DP-SGD, this quantity empirically correlates with learning utility (Ponomareva et al., 2023). Consider Algs. 1 and 2 on a dataset $D = \sqcup_{u=1}^{N} D_u$, where each user has $|D_u| = K$ examples. To meet a compute budget of B gradients per iteration, we let $p = B/G_{\text{ELS}}N$ for ELS and $q = M/N = B/G_{\text{ULS}}N$ for ULS. We analyze the impact of the group sizes $G_{\text{ELS}}, G_{\text{ULS}}$.

Suppose the loss $f(\cdot, z)$ is Lipschitz for each z. Let $L_{\text{ELS}}, L_{\text{ULS}}$ be the smallest constants such that for all θ, $\max_{z \in D} \|\nabla f(\theta, z)\|_2 \leq L_{\text{ELS}}$ and

$$\max_{u \in [N]} \max_{S \subset D_u, |S| = G_{\text{ULS}}} \left\| \frac{1}{G_{\text{ULS}}} \sum_{z \in S} \nabla f(\theta, z) \right\|_2 \leq L_{\text{ULS}}. \quad (4)$$

L_{ELS} and L_{ULS} bound the norms of the per-example and per-user gradients[1], and $L_{\text{ULS}} \leq L_{\text{ELS}}$ by the triangle inequality. For simplicity of analysis, we set the clip norm C to L_{ELS} and L_{ULS} for ELS and ULS respectively, so the clip operation in both is a no-op.

In this setting, ELS and ULS produce gradient estimates of the form $\overline{g} + \zeta$ where $\overline{g}$ is (in expectation) an average over B example gradients, and ζ is Gaussian satisfying:

$$\zeta_{\text{ELS}} \sim \mathcal{N}\left(0, \left(\sigma_{\text{ELS}} L_{\text{ELS}}/B\right)^2 I_d\right), \quad (5)$$

$$\zeta_{\text{ULS}} \sim \mathcal{N}\left(0, \left(\sigma_{\text{ULS}} L_{\text{ULS}}/M\right)^2 I_d\right). \quad (6)$$

The noise multipliers $\sigma_{\text{ELS}}, \sigma_{\text{ULS}}$ are determined by fixing a ULDP guarantee (ε, δ) and using Thms 1 and 2. Comparing the noise variances $\mathrm{var}(\zeta_{\text{ELS}})$ and $\mathrm{var}(\zeta_{\text{ULS}})$,

$$\mathrm{var}(\zeta_{\text{ELS}}) \leq \mathrm{var}(\zeta_{\text{ULS}}) \iff \frac{L_{\text{ELS}}\sigma_{\text{ELS}}}{L_{\text{ULS}}\sigma_{\text{ELS}}} \leq G_{\text{ULS}}. \quad (7)$$

[1] L_{ELS} and L_{ULS} are data-dependent. It is common (albeit non-private) practice to tune the clip norm on the dataset. The problem of privately choosing clip norms is beyond the scope of this work.

While (7) is not entirely predictive of the relative performance of ELS and ULS (since they sample data differently), it is a useful way to compare them. The ratio of $L_{\text{ELS}}/L_{\text{ULS}}$ is related to *gradient diversity* (Yin et al., 2018). In general, (7) shows that when $L_{\text{ELS}}/L_{\text{ULS}}$ is large, ULS adds noise with smaller variance. To illustrate this, we plot $\mathrm{var}(\zeta_{\text{ELS}})$ and $\mathrm{var}(\zeta_{\text{ULS}})$ in Fig. 2, fixing $K = 32$, $T = 1000$, and $N = 1024$. ULS has smaller variance when $L_{\text{ELS}} < L_{\text{ULS}}$ and either ε is small, or the compute budget B is large. Appendix C contains details and more results.

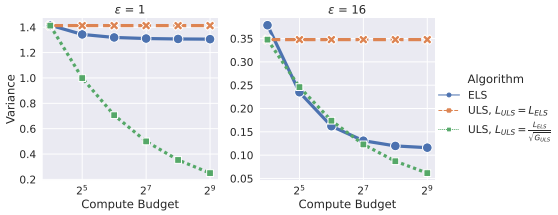


Figure 2: Noise variance of ELS and ULS. For ULS, we fix the cohort size M and vary G_{ULS}. We compare settings in which $L_{\text{ULS}} = L_{\text{ELS}}$ (no user gradient diversity), and one in which $L_{\text{ULS}} = L_{\text{ELS}}/\sqrt{G_{\text{ULS}}}$ (maximal gradient diversity).

We now let $G_{\text{ELS}} = G_{\text{ULS}} = K$ (the maximum user-dataset size). This represents the setting in which example/user sampling are maximally different. We conjecture that when $L_{\text{ELS}} = L_{\text{ULS}}$, $\mathrm{var}(\zeta_{\text{ELS}}) \leq \mathrm{var}(\zeta_{\text{ULS}})$. This is worst-case for ULS, as all gradients within a user point in the same direction. This conjecture is empirically supported by Figure 2 by comparing variance at the largest compute budget for ELS and ULS with $L_{\text{ELS}} = L_{\text{ULS}}$.

Conjecture 1. *Let $G_{ULS} = G_{ELS} = K$, and $p = q$. For all $\varepsilon, \delta, T, p$, and K, $\sigma_{ELS} \leq K\sigma_{ULS}$.*

While this conjecture is challenging to prove for (ε, δ)-DP, we show a weaker version of the conjecture holds for "one-sided" α-RDP where α is an integer. Recall that for $\alpha > 1$, the α-Rényi divergence between distributions P, Q is defined by $R_\alpha(P, Q) = (1/\alpha - 1) \log \mathbb{E}_{x \sim Q}\left[(P(x)/Q(x))^\alpha\right]$.

Lemma 1. *Let $P_K(\sigma) = \mathcal{N}(\mathrm{Binom}(K, p), \sigma^2)$ and $Q(\sigma) = \mathcal{N}(0, \sigma^2)$. For integers $\alpha > 1$, $K \geq 1$:*

$$R_\alpha\big(P_K(K\sigma), Q(K\sigma)\big) \leq R_\alpha\big(P_1(\sigma), Q(\sigma)\big).$$

The RHS is approximately the Rényi-DP parameter of one iteration of ULS with noise multiplier σ and the LHS is the Rényi-DP parameter of one iteration of ELS with noise multiplier $K\sigma$. We give the proof in Appendix D.

Mean estimation task. We corroborate our findings above on a synthetic mean estimation task in Appendix E. Although the loss is not globally Lipschitz, we see similar findings to that of the above: ULS performs significantly better than ULS in settings with high user gradient diversity, especially when ε is small or the compute budget is large.

4. Language Model Results

We validate our findings above, especially the improvement of ULS over ELS when ε is small or the compute budget is large, on realistic LLM fine-tuning tasks. Our results suggest that for LLM fine-tuning, users often have high gradient diversity, and that this often leads ULS to outperform ELS.

Experimental setup. We fine-tune a 350 million parameter decoder-only transformer model on two datasets: Stack Overflow and CC-News. The former has over 300,000 users, while the latter has nearly 10,000. We pre-train our model on C4. In order to minimize information leakage between pre-training and fine-tuning, we use near-duplicate detection (Lee et al., 2022) and URL filtering to minimizer dataset overlap. We pre-train on this de-duplicated version of C4. We pre-train and fine-tune using Adafactor (Shazeer & Stern, 2018), and use varying compute budgets and privacy levels ε. See Appendix F for details.

Selecting group sizes. $G_{\text{ELS}}, G_{\text{ULS}}$ can be crucial for optimal performance of ELS and ULS. In Appendix H and I, we give empirically validated heuristics for choosing them. We show that letting G_{ELS} be the median user dataset size works well across tasks. While G_{ULS} is more complicated, we show that G_{ULS} can be selected by estimating L_{ULS} in (4). We can estimate the variance reduction by increasing G_{ULS} by using (6) and choose the right value for a compute budget. See Appendix I for details. We use these heuristics to set $G_{\text{ELS}}, G_{\text{ULS}}$ in the sequel.

Privacy-utility-compute trade-offs. We now apply ELS and ULS for a variety of compute budgets and ε values, using the heuristics above to select $G_{\text{ELS}}, G_{\text{ULS}}$. We fine-tune on Stack Overflow and CC-News, and compute the test loss on the final iterate. In Figures 3 and 4, we plot privacy-loss trade-offs for three distinct compute budgets.

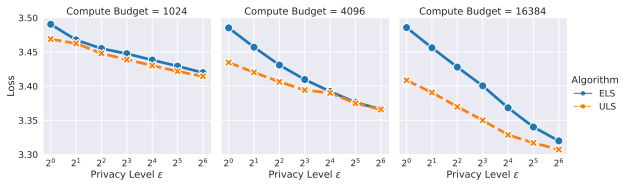


Figure 3: Privacy-loss trade-offs on Stack Overflow, for varying compute budgets.

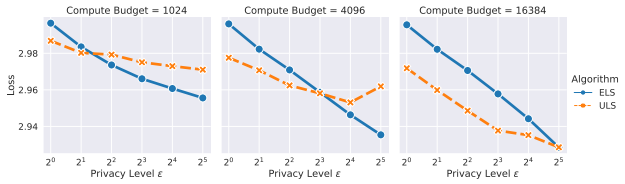


Figure 4: Privacy-loss trade-offs on CC-News, for varying compute budgets.

For Stack Overflow, ULS performs at least as well as ELS in all settings. The improvement is largest for small ε and large compute budgets. For CC-News, ELS outperforms ULS in some settings, especially when the compute budget is small or ε is small. For both datasets, ULS improves more with increased compute budgets: if we fix a privacy level ε, the loss for ULS drops more significantly as the compute budget increases than for ELS. We demonstrate this in greater detail in Appendix J.1.

Impact of dataset size. We investigate, for a fixed compute budget, how ELS and ULS compare as we vary the number of users. To test this, we use CC-News. We fix all other factors, but vary the number of users *only* in the privacy accounting. We perform the same experiments, but instead assume there are $1\times, 10\times$, and $100\times$ more users in the dataset in the accounting. The results for $\varepsilon = 4$ are in Figure 5. Throughout, the loss of ULS decreases more as compute budget increases, but the loss of ELS decreases more as the number of users increases.

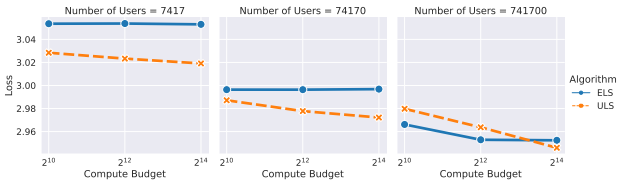


Figure 5: Compute versus loss on CC-News, as the number of users in the privacy accounting varies.

Model personalization. In some settings, we may wish to personalize the model after fine-tuning on user data. We measure the ability of the fine-tuned models to personalize to downstream user data. We find that both algorithms seem to benefit comparably from personalization, which does not change the general trade-offs discussed above. For details and results, see Appendix J.2.

5. Discussion

Our work highlights two general findings. First, by introducing tight group-level accounting, we can make ELS a practical method for ULDP that is scalable to LLM settings and serves as a useful baseline. Second, despite this accounting improvement, ULS often outperforms ELS in practical LLM fine-tuning. While we are able to scale ULS to models with hundreds of millions of parameters, ULS uses user-level sampling that is distinct from most LLM training algorithms. Future work is needed to determine which algorithms best balance scalability and performance when training with formal ULDP guarantees.

References

Paxml. https://github.com/google/paxml. Accessed: 2024-05-20.

Praxis. https://github.com/google/praxis. Accessed: 2024-05-20.

Adnan, M., Kalra, S., Cresswell, J. C., Taylor, G. W., and Tizhoosh, H. R. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1): 1953, 2022.

Agarwal, N., Suresh, A. T., Yu, F. X. X., Kumar, S., and McMahan, B. cpSGD: Communication-efficient and differentially-private distributed SGD. *NeurIPS*, 31, 2018.

Amin, K., Kulesza, A., Munoz, A., and Vassilvtiskii, S. Bounding User Contributions: A Bias-Variance Trade-off in Differential Privacy. In *ICML*, pp. 263–271. PMLR, 2019.

Asi, H. and Liu, D. User-level Differentially Private Stochastic Convex Optimization: Efficient Algorithms with Optimal Rates. In *International Conference on Artificial Intelligence and Statistics*, pp. 4240–4248. PMLR, 2024.

Authors, T. T. F. TensorFlow Federated Stack Overflow dataset, 2019. URL https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data.

Bassily, R. and Sun, Z. User-level Private Stochastic Convex Optimization with Optimal Rates. In *International Conference on Machine Learning*, pp. 1838–1851. PMLR, 2023.

Bhatia, K., Narayan, A., De Sa, C. M., and Ré, C. Tart: A plug-and-play transformer module for task-agnostic reasoning. *Advances in Neural Information Processing Systems*, 36:9751–9788, 2023.

Carlini, N., Liu, C., Erlingsson, Ú., Kos, J., and Song, D. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security*, pp. 267–284, 2019.

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 2633–2650, 2021.

Carlini, N., Ippolito, D., Jagielski, M., Lee, K., Tramer, F., and Zhang, C. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TatRHT_1cK.

Charles, Z., Mitchell, N., Pillutla, K., Reneer, M., and Garrett, Z. Towards federated foundation models: Scalable dataset pipelines for group-structured learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Chen, M., Zhang, Z., Wang, T., Backes, M., and Zhang, Y. FACE-AUDITOR: Data auditing in facial recognition systems. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 7195–7212, Anaheim, CA, August 2023. USENIX Association. ISBN 978-1-939133-37-3. URL https://www.usenix.org/conference/usenixsecurity23/presentation/chen-min.

Chen, M. X., Lee, B. N., Bansal, G., Cao, Y., Zhang, S., Lu, J., Tsay, J., Wang, Y., Dai, A. M., Chen, Z., et al. Gmail Smart Compose: Real-Time Assisted Writing. In *KDD*, pp. 2287–2295, 2019.

Choquette-Choo, C. A., Ganesh, A., Steinke, T., and Thakurta, A. G. Privacy amplification for matrix mechanisms. In *The Twelfth International Conference on Learning Representations*, 2023.

Choquette-Choo, C. A., Ganesh, A., Steinke, T., and Thakurta, A. G. Privacy amplification for matrix mechanisms. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xUzWmFdglP.

Cummings, R., Feldman, V., McMillan, A., and Talwar, K. Mean Estimation with User-level Privacy under Data Heterogeneity. *NeurIPS*, 35:29139–29151, 2022.

De, S., Berrada, L., Hayes, J., Smith, S. L., and Balle, B. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

Dong, W., Luo, Q., and Yi, K. Continual Observation under User-level Differential Privacy. In *IEEE Symposium on Security and Privacy*, pp. 2190–2207. IEEE, 2023.

Doroshenko, V., Ghazi, B., Kamath, P., Kumar, R., and Manurangsi, P. Connect the dots: Tighter discrete approximations of privacy loss distributions. *Proceedings on Privacy Enhancing Technologies*, 2022:552–570, 10 2022. doi: 10.56553/popets-2022-0122.

DP Team. Google's differential privacy libraries., 2022. https://github.com/google/differential-privacy.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proc. of the Third Conf. on Theory of Cryptography (TCC)*, pp. 265–284, 2006.

Epasto, A., Mahdian, M., Mao, J., Mirrokni, V., and Ren, L. Smoothly Bounding User Contributions in Differential Privacy. *NeurIPS*, 33:13999–14010, 2020.

Fang, J. and Yi, K. Privacy Amplification by Sampling under User-level Differential Privacy. *Proceedings of the ACM on Management of Data*, 2(1):1–26, 2024.

George, A. J., Ramesh, L., Singh, A. V., and Tyagi, H. Continual Mean Estimation Under User-Level Privacy. *IEEE Journal on Selected Areas in Information Theory*, 2024.

Geyer, R. C., Klein, T., and Nabi, M. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

Ghazi, B., Kamath, P., Kumar, R., Manurangsi, P., Meka, R., and Zhang, C. User-Level Differential Privacy With Few Examples Per User. *Advances in Neural Information Processing Systems*, 36, 2023a.

Ghazi, B., Kamath, P., Kumar, R., Manurangsi, P., Meka, R., and Zhang, C. On User-level Private Convex Optimization. In *ICML*, pp. 11283–11299. PMLR, 2023b.

Girgis, A. M., Data, D., and Diggavi, S. Distributed User-Level Private Mean Estimation. In *IEEE International Symposium on Information Theory*, pp. 2196–2201. IEEE, 2022.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Huang, R., Zhang, H., Melis, L., Shen, M., Hejazinia, M., and Yang, J. Federated Linear Contextual Bandits with User-level Differential Privacy. In *ICML*, pp. 14060–14095, 2023.

Kairouz, P., Mcmahan, B., Song, S., Thakkar, O., Thakurta, A., and Xu, Z. Practical and Private (Deep) Learning Without Sampling or Shuffling. In *ICML*, volume 139, pp. 5213–5225, 2021.

Kandpal, N., Pillutla, K., Oprea, A., Kairouz, P., Choquette-Choo, C. A., and Xu, Z. User inference attacks on large language models. *arXiv preprint arXiv:2310.09266*, 2023.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kapp, A., Nuñez von Voigt, S., Mihaljević, H., and Tschorsch, F. Towards mobility reports with user-level privacy. *Journal of Location Based Services*, 17(2):95–121, 2023.

Kato, F., Xiong, L., Takagi, S., Cao, Y., and Yoshikawa, M. ULDP-FL: Federated learning with across silo user-level differential privacy. *arXiv preprint arXiv:2308.12210*, 2023.

Koskela, A., Jalko, J., Prediger, L., and Honkela, A. Tight differential privacy for discrete-valued mechanisms and for the subsampled gaussian mechanism using FFT. In Banerjee, A. and Fukumizu, K. (eds.), *24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research, United States, 2021. Microtome Publishing. 24th International Conference on Artificial Intelligence and Statistics (AISTATS) ; Conference date: 13-04-2021 Through 15-04-2021.

Kurakin, A., Ponomareva, N., Syed, U., MacDermed, L., and Terzis, A. Harnessing large-language models to generate private synthetic text. *arXiv preprint arXiv:2306.01684*, 2023.

Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, 2022.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient 384 prompt tuning. *arXiv preprint arXiv:2104.08691*, 282, 2021.

Levy, D., Sun, Z., Amin, K., Kale, S., Kulesza, A., Mohri, M., and Suresh, A. T. Learning with User-Level privacy. *Advances in Neural Information Processing Systems*, 34: 12466–12479, 2021.

Li, G., Rezaei, S., and Liu, X. User-Level Membership Inference Attack against Metric Embedding Learning. In *ICLR 2022 Workshop on PAIR2Struct: Privacy, Accountability, Interpretability, Robustness, Reasoning on Structured Data*, 2022.

Li, J., Khodak, M., Caldas, S., and Talwalkar, A. Differentially private meta-learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJgqMRVYvr.

Liu, Y., Suresh, A. T., Yu, F. X. X., Kumar, S., and Riley, M. Learning discrete distributions: user vs item-level privacy. *NeurIPS*, 33:20965–20976, 2020.

Liu, Y., Suresh, A. T., Zhu, W., Kairouz, P., and Gruteser, M. Algorithms for bounding contribution for histogram estimation under user-level privacy. In *ICML*, pp. 21969–21996, 2023.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJ0hF1Z0b.

Miao, Y., Xue, M., Chen, C., Pan, L., Zhang, J., Zhao, B. Z. H., Kaafar, D., and Xiang, Y. The Audio Auditor: User-Level Membership Inference in Internet of Things Voice Services. In *Privacy Enhancing Technologies Symposium (PETS)*, 2021.

Narayanan, S., Mirrokni, V., and Esfandiari, H. Tight and Robust Private Mean Estimation with Few Users. In *ICML*, pp. 16383–16412, 2022.

Pelikan, M., Azam, S. S., Feldman, V., Silovsky, J., Talwar, K., Likhomanenko, T., et al. Federated learning with differential privacy for end-to-end speech recognition. *arXiv preprint arXiv:2310.00098*, 2023.

Ponomareva, N., Hazimeh, H., Kurakin, A., Xu, Z., Denison, C., McMahan, H. B., Vassilvitskii, S., Chien, S., and Thakurta, A. G. How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy. *Journal of Artificial Intelligence Research*, 77:1113–1201, July 2023. ISSN 1076-9757. doi: 10.1613/jair.1.14649. URL http://dx.doi.org/10.1613/jair.1.14649.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

Rush, K., Charles, Z., and Garrett, Z. FAX: Scalable and differentiable federated primitives in jax. *arXiv preprint arXiv:2403.07128*, 2024.

Scao, T. L. and Rush, A. M. How many data points is a prompt worth? *arXiv preprint arXiv:2103.08493*, 2021.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Song, C. and Shmatikov, V. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Song, M., Wang, Z., Zhang, Z., Song, Y., Wang, Q., Ren, J., and Qi, H. Analyzing User-Level Privacy Attack Against Federated Learning. *IEEE Journal on Selected Areas in Communications*, 38(10):2430–2444, 2020.

Vadhan, S. The complexity of differential privacy. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pp. 347–450, 2017.

Wang, Z., Song, M., Zhang, Z., Song, Y., Wang, Q., and Qi, H. Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 2512–2520, 2019.

Wei, K., Li, J., Ding, M., Ma, C., Su, H., Zhang, B., and Poor, H. V. User-Level Privacy-Preserving Federated Learning: Analysis and Performance Optimization. *IEEE Transactions on Mobile Computing*, 21(9):3388–3401, 2021.

Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

Xu, Z., Collins, M., Wang, Y., Panait, L., Oh, S., Augenstein, S., Liu, T., Schroff, F., and McMahan, H. B. Learning to generate image embeddings with user-level differential privacy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7969–7980, 2023a.

Xu, Z., Zhang, Y., Andrew, G., Choquette, C., Kairouz, P., Mcmahan, B., Rosenstock, J., and Zhang, Y. Federated learning of gboard language models with differential privacy. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pp. 629–639, 2023b.

Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., and Bartlett, P. Gradient diversity: a key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1998–2007. PMLR, 2018.

Zhou, X. and Bassily, R. Task-level Differentially Private Meta Learning. In *NeurIPS*, 2022.

Zhu, Y., Dong, J., and Wang, Y.-X. Optimal accounting of differential privacy via characteristic function. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings*

of The 25th International Conference on Artificial Intelligence and Statistics, volume 151 of *Proceedings of Machine Learning Research*, pp. 4782–4817. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/zhu22c.html.

A. Detailed Related Work

A.1. Theoretical Advances in User-Level DP

Here, we survey the existing theoretical work on user-level DP (ULDP) in various settings.

Reductions to example-level DP. Ghazi et al. (2023b;a) give ULDP bounds by running example-level DP algorithms on subsets of data where the algorithm is stable to deletions ((Ghazi et al., 2023b) only handles output perturbation while (Ghazi et al., 2023a) can handle any arbitrary algorithm). This involves a brute-force-search for a stable subset of the data (together with a propose-test-release loop), which can be super-polynomial in the number of examples, and is thus computationally intractable. We note that the ULDP bounds of (Ghazi et al., 2023a) rely on generic group privacy reductions (Vadhan, 2017). In contrast, we give tight accounting for deriving ULDP guarantees from example-level DP guarantees in the specific setting of `DP-SGD-ELS`, enabling us to scale this method to practical LLM settings.

We also note that prior works (Amin et al., 2019; Epasto et al., 2020) have also reduced learning with ULDP to the example-level DP setting where each example is associated with a weight, and the weights are computed to maximize the final utility. On the other hand, we fix a number G_{ELS} of examples per user globally and randomly select G_{ELS} samples; this can be viewed as assigning a binary per-example weights. It is unclear how to adapt the analytical approaches of (Amin et al., 2019; Epasto et al., 2020) to LLMs as we do not have access to utility bounds.

Approaches based on clipping user-level gradient. Several prior approaches rely on bounding user contributions by clipping user-level gradients and combining them with (different choices of) robust mean estimation algorithms (Levy et al., 2021; Bassily & Sun, 2023; Asi & Liu, 2024; De et al., 2022). This line of work aims for better rates of convergence under the weaker assumptions. On the other hand, we empirically evaluate practical algorithms derived from these approaches by using user-level gradient clipping but replacing the inefficient robust aggregation approaches with a simple unweighted average (which is efficiently implementable on hardware accelerators).

Simpler theoretical settings. Related work also considers ULDP in stylized problems such as learning discrete distributions (Liu et al., 2020) and histograms (Liu et al., 2023). Mean estimation under ULDP was also considered in (Girgis et al., 2022; Narayanan et al., 2022; Cummings et al., 2022). This subroutine was also the key building block of the theoretical works (Levy et al., 2021; Bassily & Sun, 2023; Asi & Liu, 2024) in their learning bounds under ULDP. Continual mean estimation (George et al., 2024) and continual observation (Dong et al., 2023) have also been considered under ULDP.

Other related work. Fang & Yi (2024) give an approach to privacy amplification from sampling users in a graph structure.

A.2. Federated Learning and User-Level DP

As we noted in the main paper, empirical advances in ULDP have been driven primarily by research in federated learning, starting with (Geyer et al., 2017; McMahan et al., 2018; Agarwal et al., 2018; Kato et al., 2023). Specifically, McMahan et al. (2018) propose DP-FedSGD (resp. DP-FedAvg) which clips user-level gradients (resp. pseudo-gradients generated by multiple local gradient steps). Note that DP-FedSGD coincides with `DP-SGD-ULS` that we analyze. This user-level gradient clipping approach can be generalized beyond DP-SGD to algorithms that add noise that is correlated across iterations (Kairouz et al., 2021); these algorithms have been deployed in industrial systems to provide formal ULDP guarantees (Xu et al., 2023b).

Kato et al. (2023) give ULDP algorithms in a cross-silo federated learning setting. Here, a user's data might be split across multiple data silos and each silo might contain multiple datapoints from a single user. Their approach combines `DP-SGD-ELS` with FedAvg, where they use generic group privacy reductions to promote example-level DP guarantees to the user-level.

Several follow-up works have leveraged algorithms similar to `DP-SGD-ULS` in other settings such as contextual bandits (Huang et al., 2023), meta-learning (Li et al., 2020; Zhou & Bassily, 2022), and embedding learning (Xu et al., 2023a) as well as applications such as medical image analysis (Adnan et al., 2022), speech recognition (Pelikan et al., 2023) and releasing mobility reports (i.e. aggregate location statistics) (Kapp et al., 2023).

A.3. User-level Privacy Attacks

The work on ULDP is motivated in part due to the various privacy attacks conducted at the user level. For instance, an adversary might be able to infer whether a user's data was used to train a model (Song & Shmatikov, 2019), even if the

adversary does not have access to the exact training samples of the user (Kandpal et al., 2023). Further, (Kandpal et al., 2023) demonstrate that example-level DP is not effective in mitigating such user inference attacks, especially at low false positive rates.

Such attacks have been designed not only for LLMs (Kandpal et al., 2023) but also for embedding learning for vision (Li et al., 2022), speech recognition for IoT devices (Miao et al., 2021), facial recognition systems (Chen et al., 2023). In federated learning, Wang et al. (2019) and Song et al. (2020) study the risk to user-level privacy from a malicious server. ULDP provides formal upper bounds on the success rates of all such user-level privacy attacks and the algorithms we study are broadly applicable in all of these settings.

B. Tight Accounting for `DP-SGD-ELS`

Notation. Given a distribution P over a space $\mathcal{A}$ and $n \in \mathbb{Z}_{>0}$, let $P^{\otimes n}$ denote the product distribution on $\mathcal{A}^n$.

Main Result. In this section, we show the following tight accounting statement for Algorithm 1:

Theorem 3. *Let $\mathcal{M}_f$ be T iterations of `DP-SGD-ELS` using noise multiplier σ, loss function f, group size $G_{ELS} = K$, and Poisson sampling with probability p. That is, $\mathcal{M}_f(D)$ is the distribution of models $(\theta^1, \theta^2, \dots \theta^T)$ produced by applying `DP-SGD-ELS` to a dataset D. Let D and D' be two datasets such that $D' = D \sqcup A$ where $|A| \leq K$. Then for all ε:*

$$H_{e^\varepsilon}^{\mathrm{sym}}(\mathcal{M}_f(D), \mathcal{M}_f(D')) \leq H_{e^\varepsilon}^{\mathrm{sym}}(\mathcal{N}(0, \sigma^2)^{\otimes T}, \mathcal{N}(\mathrm{Binom}(K, p), \sigma^2)^{\otimes T}).$$

Furthermore, this is tight, i.e. there exists a loss function f and datasets D, D' such that:

$$H_{e^\varepsilon}^{\mathrm{sym}}(\mathcal{M}_f(D), \mathcal{M}_f(D')) = H_{e^\varepsilon}^{\mathrm{sym}}(\mathcal{N}(0, \sigma^2)^{\otimes T}, \mathcal{N}(\mathrm{Binom}(K, p), \sigma^2)^{\otimes T}).$$

Since we cap the number of examples any user can contribute to the dataset in Alg. 1, this implies Theorem 1. To prove this, we use the following lemma from (Choquette-Choo et al., 2024), derived using an analysis based on Mixture-of-Gaussians mechanisms:

Lemma 2 (Lemma 4.5 of (Choquette-Choo et al., 2024))**.** *Let $\boldsymbol{x}$ be a random variable on $\mathbb{R}^d$, and x be a random variable on $\mathbb{R}$ such that $\|\boldsymbol{x}\|_2$ is stochastically dominated by x (that is, there is a coupling of $\boldsymbol{x}$ and x such that under this coupling, $\|\boldsymbol{x}\|_2 \leq x$ with probability 1). Then for all $\varepsilon > 0$:*

$$H_{e^\varepsilon}(\mathcal{N}(0, \sigma^2 I_d), \mathcal{N}(\boldsymbol{x}, \sigma^2 I_d)) \leq H_{e^\varepsilon}(\mathcal{N}(0, \sigma^2), \mathcal{N}(x, \sigma^2)),$$
$$H_{e^\varepsilon}(\mathcal{N}(\boldsymbol{x}, \sigma^2 I_d), \mathcal{N}(0, \sigma^2 I_d)) \leq H_{e^\varepsilon}(\mathcal{N}(x, \sigma^2), \mathcal{N}(0, \sigma^2)).$$

We will also use the following "quasi-convexity" property of DP:

Lemma 3. *Let $w_1, w_2, \dots, w_n \geq 0$ be probabilities summing to 1. Given distributions $\{P_i\}, \{Q_i\}$, let $P = \sum_i w_i P_i$ and $Q = \sum_i w_i Q_i$. Then for any $\alpha \geq 0$:*

$$H_\alpha(P, Q) \leq \max_i H_\alpha(P_i, Q_i).$$

Proof. We have:

$$\begin{aligned} H_\alpha(P, Q) &= \int \max\{P(x) - \alpha Q(x), 0\} dx = \int \max\{\sum_i w_i (P_i(x) - \alpha Q_i(x)), 0\} dx \\ &\overset{(*_1)}{\leq} \int \sum_i w_i \max\{P_i(x) - \alpha Q_i(x), 0\} dx = \sum_i w_i \int \max\{P_i(x) - \alpha Q_i(x), 0\} dx \\ &= \sum_i w_i H_\alpha(P_i, Q_i) \overset{(*_2)}{\leq} \max_i H_\alpha(P_i, Q_i). \end{aligned}$$

$(*_1)$ is the observation that $\max\{a+c, b+d\} \leq \max\{a,b\} + \max\{c,d\}$ i.e. "the max of sums is less than the sum of maxes" and $(*_2)$ holds because the w_i are non-negative and sum to 1. □

Finally, we will use the following observation about bijections and hockey-stick divergences:

Observation 1. *Let f be any bijection, and for distribution X let $f(X)$ denote the distribution given by $f(x), x \sim X$. Then for any P, Q, ε:*

$$H_{e^\varepsilon}^{\mathrm{sym}}(P,Q) = H_{e^\varepsilon}^{\mathrm{sym}}(f(P), f(Q)).$$

Proof. This follows by applying the post-processing property of DP, which says for any function g

$$H_{e^\varepsilon}^{\mathrm{sym}}(P,Q) \geq H_{e^\varepsilon}^{\mathrm{sym}}(g(P), g(Q)).$$

The observation follows by applying the post-processing property to f and f^{-1}. □

Proof of Theorem 3. Since $(\theta^1, \theta^2, \ldots, \theta^T) \leftrightarrow (\theta^1 - \theta^0, \theta^2 - \theta^1, \ldots, \theta^T - \theta^{T-1})$ is a bijection (we treat the initialization θ^0 as public), we can assume through the proof that DP-SGD-ELS instead outputs the tuple $(\theta^1 - \theta^0, \theta^2 - \theta^1, \ldots, \theta^T - \theta^{T-1})$. For simplicity of presentation, we will assume f is C-Lipschitz and thus that clip is a no-op.

The tightness of this results follows from considering a one-dimensional 1-Lipschitz loss function f such that that for all $z \in D$, $f(\theta, z) = 0$ and for all $z \in A$, $f(\theta, z) = -\theta$. Letting $\eta = 1$, the distribution of each $\theta^t - \theta^{t-1}$ is exactly $x \sim \mathcal{N}(0, \sigma^2)$ for D and $x \sim \mathcal{N}(\mathrm{Binom}(K,p), \sigma^2)$ for D'.

For the upper bound, we will show

$$H_{e^\varepsilon}(\mathcal{M}_f(D), \mathcal{M}_f(D')) \leq H_{e^\varepsilon}(\mathcal{N}(0,\sigma^2)^{\otimes T}, \mathcal{N}(\mathrm{Binom}(K,p), \sigma^2)^{\otimes T}).$$

The analogous bound on $H_{e^\varepsilon}(\mathcal{M}_f(D'), \mathcal{M}_f(D))$ (and thus the desired bound on $H_{e^\varepsilon}^{\mathrm{sym}}(\mathcal{M}_f(D), \mathcal{M}_f(D'))$) follows by Lemma 28 of (Zhu et al., 2022).

By adaptive composition of privacy loss distributions (see e.g. Theorem 2.4 of (Doroshenko et al., 2022)), it suffices to show given any fixed θ^t, if P, Q are the distribution of $\theta^{t+1} - \theta^t$ conditioned on θ^t using D and D' respectively, then for all ε we have:

$$H_{e^\varepsilon}(P,Q) \leq H_{e^\varepsilon}(\mathcal{N}(0,\sigma^2), \mathcal{N}(\mathrm{Binom}(K,p), \sigma^2)).$$

Recall that S^{t+1} is the set of examples sampled in iteration t, and let P_S, Q_S denote the distributions of P, Q respectively conditioned on the event $S^{t+1} \cap D = S$. The distribution of $S^{t+1} \cap D$ is the same for D and D', so by Lemma 3 for all ε:

$$H_{e^\varepsilon}(P,Q) \leq \max_S H_{e^\varepsilon}(P_S, Q_S),$$

hence it suffices to show for any fixed S and all ε:

$$\max_S H_{e^\varepsilon}(P_S, Q_S) \leq H_{e^\varepsilon}(\mathcal{N}(0,\sigma^2), \mathcal{N}(\mathrm{Binom}(K,p), \sigma^2)).$$

Now let $P'_S = -\frac{p + \eta \sum_{z \in S} \nabla f(\theta^t, z)}{\eta C}$ where $p \sim P_S$. We define Q'_S analogously. The correspondence

$$p \leftrightarrow -\frac{p + \eta \sum_{z \in S} \nabla f(\theta^t, z)}{\eta C}$$

is a bijection on $\mathbb{R}^d$, so $H_{e^\varepsilon}(P_S, Q_S) = H_{e^\varepsilon}(P'_S, Q'_S)$. We can exactly write the distributions of P'_S, Q'_S:

$$P'_S = \mathcal{N}(0, \sigma^2 I_d), Q'_S = \mathcal{N}\left(\sum_{z \in S^{t+1} \cap A} \frac{\nabla f(\theta^t, z)}{C}, \sigma^2 I_d\right).$$

By triangle inequality and C-Lipschitzness of f, $\left\|\sum_{z \in S^{t+1} \cap A} \frac{\nabla f(\theta^t, z)}{C}\right\|_2 \leq |S^{t+1} \cap A|$. Furthermore, $|S^{t+1} \cap A|$ is distributed according to $\mathrm{Binom}(|A|, p)$, and so $\left\|\sum_{z \in S^{t+1} \cap A} \frac{\nabla f(\theta^t, z)}{C}\right\|_2$ is stochastically dominated by $\mathrm{Binom}(K, p)$. By Lemma 2 we now have for all ε:

$$H_{e^\varepsilon}(P'_S, Q'_S) \leq H_{e^\varepsilon}(\mathcal{N}(0, \sigma^2), \mathcal{N}(\mathrm{Binom}(K, p), \sigma^2)),$$

which completes the proof. □

B.1. Implementation in `dp_accounting`

B.1.1. Computing ε and δ

The following code snippet using the `dp_accounting` library (DP Team, 2022) and `scipy` methods can be used to compute ε as a function of δ (or vice-versa) for `DP-SGD-ELS` (Algorithm 1) according to Theorem 1, for a given number of steps T, example sampling probability p, noise multiplier $\sigma_{\text{ELS}} = \sigma$, and group size $G_{\text{ELS}} = K$:

```
def get_group_level_event(T, p, sigma, K):
    sensitivities = range(K+1)
    probs = [scipy.stats.binom.pmf(x, K, p) for x in sensitivities]
    single_round_event = dp_accounting.dp_event.MixtureOfGaussiansDpEvent(
      sigma, sensitivities, probs
    )
    dp_sgd_event = dp_accounting.dp_event.SelfComposedDpEvent(
      single_round_event, T
    )
    return dp_sgd_event

event = get_group_level_event(T, p, sigma, K)
accountant = dp_accounting.pld.PLDAccountant()
accountant.compose(dp_sgd_event)

# Compute epsilon given delta
print(accountant.get_epsilon(delta))

# Compute delta given epsilon
print(accountant.get_delta(epsilon))
```

B.1.2. Computing σ_{ELS}

To figure out the minimum σ_{ELS} needed to achieve a target (ε, δ)-DP guarantee for `DP-SGD-ELS` (Algorithm 1), we can use `dp_accounting`'s `calibrate_dp_mechanism`:

```
def get_group_level_sigma(T, p, epsilon, delta, K)
    sigma_to_event = lambda sigma: get_group_level_event(T, p, sigma, K)
    return dp_accounting.calibrate_dp_mechanism(
        dp_accounting.pld.PLDAccountant,
        sigma_to_event,
        epsilon,
        delta
    )
```

C. Comparing Variances of ELS and ULS

Recall that in the setting of Section 3, the stochastic gradients produced by ELS and ULS in an iteration t are respectively given by

$$g^t_{\text{ELS}} = \frac{1}{B}\sum_{z\in S^t} \nabla f(\theta^t, z) + \zeta^t_{\text{ELS}}, \quad \zeta^t_{\text{ELS}} \sim \mathcal{N}\left(0, \left(\frac{\sigma_{\text{ELS}} L_{\text{ELS}}}{B}\right)^2 I_d\right),$$

$$g^t_{\text{ULS}} = \frac{1}{B}\sum_{u\in U^t}\sum_{z\in D_u} \nabla f(\theta^t, z) + \zeta^t_{\text{ULS}}, \quad \zeta^t_{\text{ULS}} \sim \mathcal{N}\left(0, \left(\frac{\sigma_{\text{ULS}} L_{\text{ULS}}}{M}\right)^2 I_d\right),$$

where B denotes the per-iterate (expected) compute budget of both methods, and M is the (expected) cohort size of ULS. Further recall that in this setting, there are N users, each of which have K examples.

For any specific instatiation of this setting, we can use the DP accounting tools from Appendix B to explicitly compute $\text{var}(\zeta^t_{\text{ELS}})$ and $\text{var}(\zeta^t_{\text{ULS}})$.

We do so in the following setting. We fix $N = 1024$ users, each with $K = 32$ examples. We set $G_{\text{ELS}} = 32$ (though the choice here has almost no impact on the noise variance). We vary the cohort size M, and set $G_{\text{ULS}} = B/M$ to normalize compute between ELS and ULS. We fix $T = 1000, \delta = 10^{-6}$ and vary ε in the desired (ε, δ)-DP guarantee, and compute the corresponding noise multipliers $\sigma_{\text{ELS}}, \sigma_{\text{ULS}}$ via DP accountants.

To compute variance, the only remaining relevant quantities are L_{ELS} and L_{ULS}. We fix $L_{\text{ELS}} = 10$, and vary L_{ULS}. Intuitively, the ratio of these two tells us how diverse the gradients across a user are. We consider two settings. In the first, the G_{ULS} gradients across a user in ULS are minimally diverse, so that $L_{\text{ELS}} = L_{\text{ULS}}$ for all group sizes G_{ULS}. In the second, the gradients are maximally diverse, so that

$$L_{\text{ULS}} = \frac{L_{\text{ELS}}}{\sqrt{G_{\text{ULS}}}}.$$

This setting occurs, for example, if all G_{ULS} gradients computed at each user for ULS are orthogonal with length L_{ELS}. For varying B, M and ε, we then compare three variances: the variance of ζ^t_{ELS}, and the variance of ζ^t_{ULS} for each setting of L_{ULS}.

The results are given in Fig. 6. While the results vary across settings, we see a few robust findings. First, when $L_{\text{ULS}} = L_{\text{ELS}}$, the variance of ELS is lower in nearly all settings. When $L_{\text{ULS}} = L_{\text{ELS}}/\sqrt{G_{\text{ULS}}}$, the variance of ULS is often (but not always) lower than that of ELS. We see that ULS especially tends to incur lower variance when either (1) ε is small or (2) when the compute budget is sufficiently large.

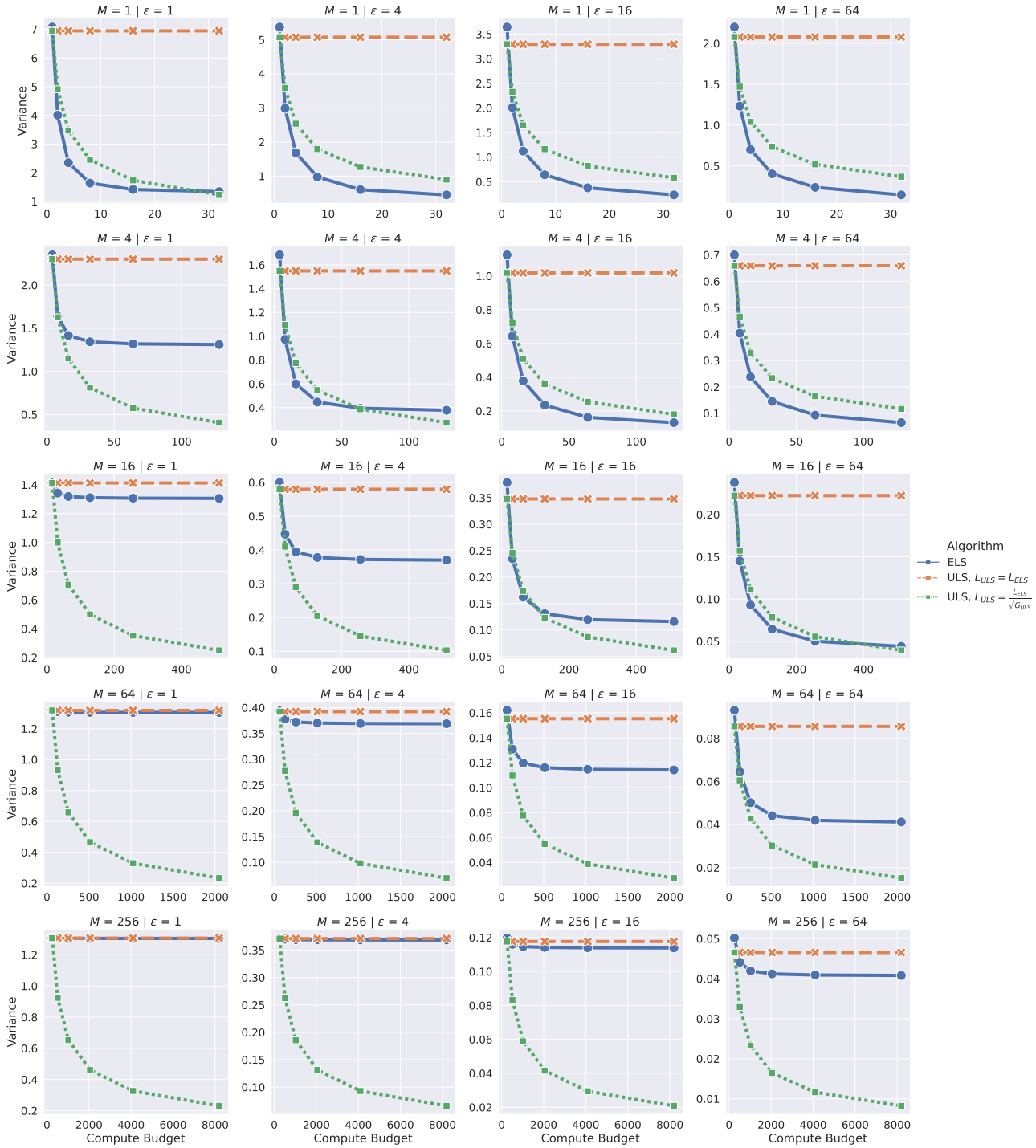


Figure 6: Noise variance of ELS and ULS, for varying compute budget B, cohort size M and privacy level ε. For ULS, we fix the cohort size M and vary G_{ULS}. We compare two settings, one in which $L_{\text{ULS}} = L_{\text{ELS}}$, and one in which $L_{\text{ULS}} = L_{\text{ELS}}/\sqrt{G_{\text{ULS}}}$. Throughout, we fix $N = 1024$ users, each with $K = 32$ examples, $G_{\text{ELS}} = 32$, $T = 1000$, $\delta = 10^{-6}$, and $L_{\text{ELS}} = 10$.

D. Proof of Lemma 1

Proof. Let $B_{K,p}(c) = \Pr[\mathrm{Binom}(K,p) = c]$. By linearity of expectation, we have

$$\begin{aligned}
&\exp((\alpha-1)R_\alpha(P_K(K\sigma), Q(K\sigma))) \\
&= \mathbb{E}_{x\sim\mathcal{N}(0,K^2\sigma^2)}\left[\left(\sum_{c\in\{0,1,\ldots,K\}} B_{K,p}(c)\cdot\exp\left(\frac{2cx-c^2}{2K^2\sigma^2}\right)\right)^\alpha\right] \\
&= \mathbb{E}_{x\sim\mathcal{N}(0,K^2\sigma^2)}\left[\sum_{c_1,c_2,\ldots,c_\alpha\in\{0,1,\ldots,K\}}\left(\prod_i B_{K,p}(c_i)\right)\cdot\exp\left(\frac{2(\sum_i c_i)x-\sum_i c_i^2}{2K^2\sigma^2}\right)\right] \\
&= \sum_{c_1,c_2,\ldots,c_\alpha\in\{0,1,\ldots,K\}}\left(\prod_i B_{K,p}(c_i)\right)\cdot\mathbb{E}_{x\sim\mathcal{N}(0,K^2\sigma^2)}\left[\exp\left(\frac{2(\sum_i c_i)x-\sum_i c_i^2}{2K^2\sigma^2}\right)\right] \\
&= \sum_{c_1,c_2,\ldots,c_\alpha\in\{0,1,\ldots,K\}}\left(\prod_i B_{K,p}(c_i)\right)\cdot\exp\left(\frac{(\sum_i c_i)^2-\sum_i c_i^2}{2K^2\sigma^2}\right)
\end{aligned}$$

This last step follows from the fact that for $a,\nu\in\mathbb{R}$ and $y\sim\mathcal{N}(0,\nu^2)$, $\mathbb{E}_y[e^{ay/\nu^2}] = e^{a^2/2\nu^2}$. For $c_i\sim\mathrm{Binom}(K,p)$, we define random variables $\{c_{i,j}|j\in\{0,1,\ldots,K\}, c_{i,j}\sim\mathrm{Ber}(p)\}$, and can write $c_i = \sum_{j\in\{0,1,\ldots,K\}} c_{i,j}$. Let $C = (c_{1,1},\ldots,c_{\alpha,K})\sim\mathrm{Ber}(p)^{\alpha K}$. Then we have:

$$\begin{aligned}
&\sum_{c_1,c_2,\ldots,c_\alpha\in\{0,1,\ldots,K\}}\left(\prod_i B_{K,p}(c_i)\right)\cdot\exp\left(\frac{(\sum_i c_i)^2-\sum_i c_i^2}{2K^2\sigma^2}\right) \\
&= \mathbb{E}_C\left[\exp\left(\frac{(\sum_{i,j} c_{i,j})^2-\sum_i(\sum_j c_{i,j})^2}{2K^2\sigma^2}\right)\right] \\
&= \mathbb{E}_C\left[\exp\left(\frac{\sum_{i\neq i',j,j'} c_{i,j}c_{i',j'}}{2K^2\sigma^2}\right)\right] \\
&= \mathbb{E}_C\left[\exp\left(\frac{\mathbb{E}_{j_1,j_2,\ldots j_\alpha \overset{u.a.r.}{\sim}\{0,1,\ldots,K\}}[\sum_{i\neq i'} c_{i,j_i}c_{i',j_{i'}}]}{2\sigma^2}\right)\right] \\
\text{(by Jensen's inequality)} &\leq \mathbb{E}_C\left[\mathbb{E}_{j_1,j_2,\ldots j_\alpha \overset{u.a.r.}{\sim}\{0,1,\ldots,K\}}\left[\exp\left(\frac{\sum_{i\neq i'} c_{i,j_i}c_{i',j_{i'}}}{2\sigma^2}\right)\right]\right] \\
\text{(by the law of total expectation)} &= \mathbb{E}_{c_1,\ldots,c_\alpha\sim\mathrm{Ber}(p)}\left[\exp\left(\frac{\sum_{i\neq i'} c_i c_{i'}}{2\sigma^2}\right)\right] \\
&= \exp((\alpha-1)R_\alpha(P_1(\sigma), Q(\sigma))).
\end{aligned}$$

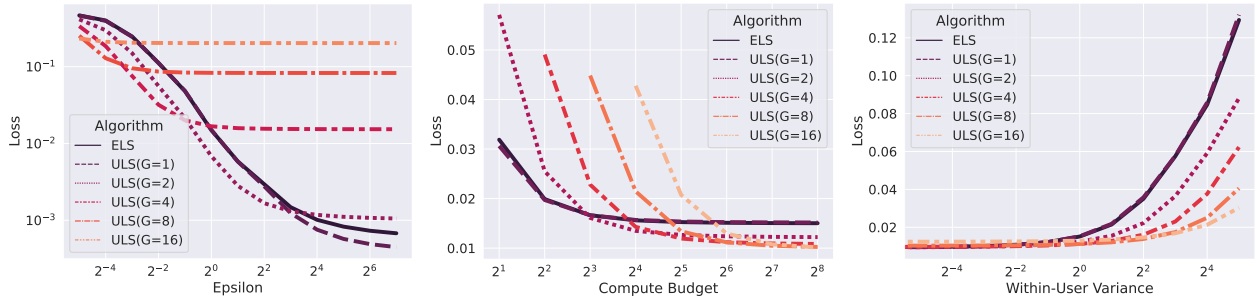
□

E. Synthetic Example: Mean Estimation

To better understand the behavior of ELS and ULS, we evaluate them on a mean estimation task with a square distance loss. By focusing on this simple setting, we can thoroughly explore the factors that influence their relative performance, including dataset characteristics, compute budget, privacy budget, and algorithm hyperparameters. In contrast to Section 3, the loss is not Lipschitz.

We first sample a population mean $\mu = \mathcal{N}(0, I_d)$, for $d = 32$. For each of $N = 256$ users, we sample a user mean $\mu_u = \mathcal{N}(\mu, \sigma_1^2 I_d)$, and user data $\{x_{u,j} \sim \mathcal{N}(\mu_u, \sigma_2^2 I_d)\}_{j=1}^K$ where $K = 16$. We refer to σ_2 as the "within-user variance". Our goal is to estimate μ under (ε, δ)-DP using ELS or ULS. To normalize compute, we set the cohort size M in ULS as $M = B/G_{\text{ULS}}$. We fix $T = 256, \delta = 10^{-6}$ and $\sigma_1 = 1$. We use default values of $\varepsilon = 1$, a per-iterate compute budget of 64, and $\sigma_2 = 1$, but vary each separately to study how they affect the performance of ELS and ULS. While we vary G_{ULS} in all experiments, we find use $G_{\text{ELS}} = K$ throughout (as it uniformly gives the best performance for ELS). For each experiment setting, we sweep over learning rate and clip norm, and report the results for the best setting across 128 random trials.

We visualize our results in Fig. 7. We find that ULS with $G_{\text{ULS}} = 1$ is comparable to if not better than ELS across all settings. We also see that ULS benefits from larger values of G_{ULS} when one of the following occurs: σ_2 (the within-user variance) is large, the compute budget (dictated by the batch size B) is large, or when ε is small. In these regimes, ULS improves significantly on ELS. We note that the notion that larger values of G_{ULS} can improve performance of ULS is corroborated theoretically for Lipschitz losses in Section 3.



(a) Varying ε. (b) Varying compute budget. (c) Varying within-user variance.

Figure 7: Performance of ELS and ULS on a synthetic mean estimation task with (a) varying ε, (b) varying compute budget, and (c) varying within-user variance σ_2.

F. Language Model Experimental Setup

While Section 3 exhibits conditions under which ULS outperforms ELS for ULDP training, it is unclear whether this translates into benefits in realistic language model tasks. To investigate this, we apply both to language model fine-tuning, across a variety of privacy and compute budgets.

Model. We use a 350 million parameter decoder-only transformer model, with a WordPiece tokenizer, implemented in Praxis (pra). We use a sequence length of 128, and train via a causal language modeling loss (i.e., next token prediction with cross-entropy loss).

Datasets. For fine-tuning, we use the Stack Overflow and CC-News datasets. Stack Overflow consists of questions and answers from the eponymous social media site, and is directly partitioned into users (Authors, 2019). The train split has 135,818,730 examples partitioned across 342,477 users. CC-News consists of English language articles on the web. We partition it according to the base domain of each article's URL. This results in 708,241 examples partitioned across 8,759 users.

For pre-training, we use a modified version of the C4 dataset (Raffel et al., 2020). Because LLMs can memorize training data (Carlini et al., 2021), we attempt to minimize privacy leakage between C4 and the fine-tuning datasets. We use a filtering scheme based on (Kurakin et al., 2023). First, we use the NearDup method (Lee et al., 2022) to identify and remove approximate duplicates between C4 and the fine-tuning datasets. Second, we filter the URLs in what remains, removing all examples associated to `stackoverflow.com` or URLs contained in CC-News. We refer to the result as C4--. For

details, see Appendix G.

Training. We perform non-private pre-training of our transformer model on the C4-- dataset. We train for 400,000 steps, using a batch size of 512. We use the Adafactor optimizer (Shazeer & Stern, 2018) with a cosine learning rate decay schedule. We tune the learning rate tuned based on C4 validation set performance. For fine-tuning, we use a dataset-dependent number of steps: 10,000 for Stack Overflow, and 2000 for CC-News. For both `ELS` and `ULS`, we use Adafactor with a constant learning rate to perform model updates. We tune the learning rate and clip norm throughout. To decouple the two, we use the normalized clipping scheme proposed by De et al. (2022).

DP accounting and sampling. We vary ε and set $\delta = n^{-1.1}$, where n is the number of examples in the dataset[2]. For Algorithm 1, $n = |D_{\text{sub}}|$, while for Algorithm 2, $n = |D|$. We compute the noise multiplier assuming amplification via sub-sampling (at the example level for `ELS`, and at the user level for `ULS`). For efficiency reasons, in practice we sample by shuffling. For Algorithm 1, we shuffle the dataset D_{sub} and sample batches of a fixed size B in shuffled order. For Algorithm 2, we shuffle the set of users and sample user cohorts of a fixed size M in shuffled order.

Software and compute resources. We define our model in Praxis (pra). For `ELS`, we use `tf.data` pipelines to create and iterate over datasets, and implement Algorithm 1 in JAX. For `ULS`, we use Dataset Grouper (Charles et al., 2023) to create and efficiently iterate over users in our datasets. `ULS` is implemented as a parallelized compute process in FAX (Rush et al., 2024), enabling linear scaling with respect to compute resources. All experiments were run in PAX (pax). We used TPU v3 pod slices, in 4×4, 8×8, and 16×16 topologies for our small, medium, and large compute budget experiments.

G. Datasets

We use two user-partitioned datasets, Stack Overflow and CC-News, for fine-tuning and evaluation. The Stack Overflow dataset (Authors, 2019), consists of questions and answers from `stackoverflow.com`, and is naturally partitioned by user on the platform. The dataset contains three splits: train (examples before 2018-01-01 UTC), test (examples after 2018-01-01 UTC) and validation (examples from held-out users across time). For our experiments, we use the train split for fine-tuning and the test split for evaluation. Fig. 8 depicts the distribution of user dataset size in the train split. Stack Overflow evaluation metrics reported throughout the main paper are measured on the full test split. In Appendix J.2, we also include an ablation on model personalization using Stack Overflow test. Personalization is done by using half of each test user's examples for further fine-tuning and evaluating each personalized model on the reserved half of each test user's examples.

CC-News consists of English-language articles on the web, a subset of the Colossal Clean Crawled Corpus (C4) dataset. For CC-News, we leverage Dataset Grouper to obtain user-level partitioning by base domain of each article's URL (Charles et al., 2023). We reserve a portion of each user's data in CC-News for evaluation and train on the remainder. In order to do so, we remove all users with only a single example. Eval user datasets are then formed by taking the first 10% of the data of each user, but only up to a maximum of 32 examples. The remaining 90% of user data is allocated for training. Fig. 8 shows the distribution of examples per user in the CC-News training dataset, after the held-out portion of examples are removed. Dataset statistics for Stack Overflow and CC-News are reported in Table 1.

Table 1: User-level statistics for the train and test splits of Stack Overflow and CC-News.

Dataset	**Split**	**Dataset-level Statistics**		**User-level Statistics (# examples / user)**		
		# Examples	# Users	Min	Median	Max
Stack Overflow	Train	135.8 M	342.5 K	1	183	194.2 K
	Test	16.6 M	204.1 K	1	43.2 K	29
CC-News	Train	661.6 K	7.4 K	1	16	24.4 K
	Test	45.3 K	7.4 K	1	2	32

To create a pre-training dataset with minimal privacy leakage, we start with the C4 dataset (Raffel et al., 2020) and apply

[2]For CC-News accounting, we make the assumption that there are $10\times$ more users than are actually present in the dataset, due to the smaller number of users. We revisit this choice later on.

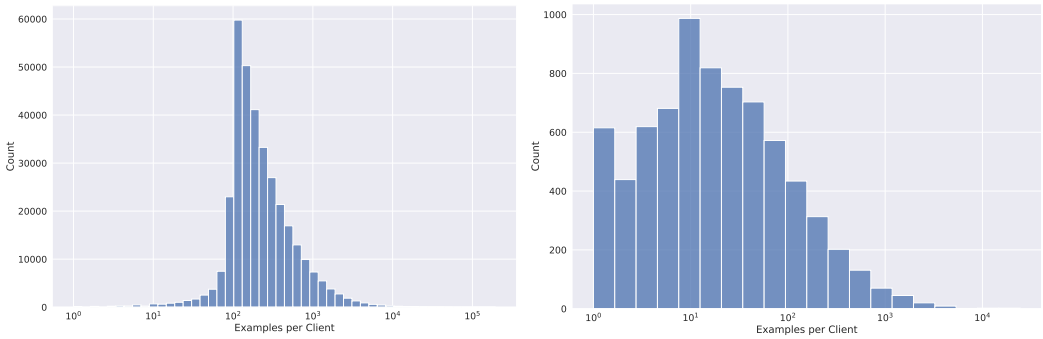


Figure 8: Histograms of user dataset sizes for the training splits of Stack Overflow (left) and CC-News (right).

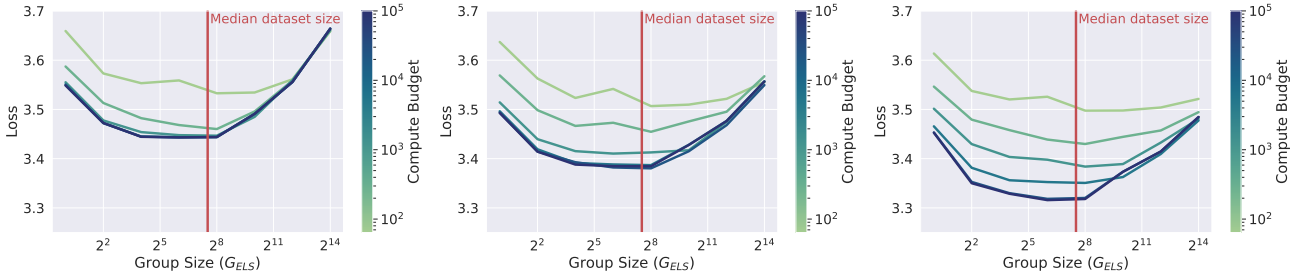
de-duplication techniques. First, we apply the *approximate* duplicate detection method NearDup proposed by Lee et al. (2022) to filter out near-duplicates between C4 and the union of Stack Overflow and CC-News. To further remove potential overlap, we filter out all examples associated with `stackoverflow.com` or any URL in CC-News. This yields the C4--dataset, which we use for pre-training. Statistics of example counts at each stage of the filtering pipeline are reported in Table 2.

Table 2: Example counts from each stage of the pipeline to filter C4 and produce C4--.

Dataset	Split	# Examples	# Ex, de-duplicated	# Ex, de-duplicated and URL-filtered
C4	Train	364.6 M	345.6 M	325.9 M

H. Configuring DP-SGD-ELS

In Algorithm 1, $G_{\texttt{ELS}}$ governs an important trade-off. Smaller values mean that ELS trains on a small fraction of the dataset, while larger values means that users with more examples are potentially over-represented in sampling. To understand this, we fine-tune on both datasets using ELS, for varying compute budget and group size. The results for Stack Overflow and CC-News are given in Figures 9 and 10. While behavior for boundary values can be quite complicated and dependent on ε, setting G_{ELS} to the median dataset size works well throughout.

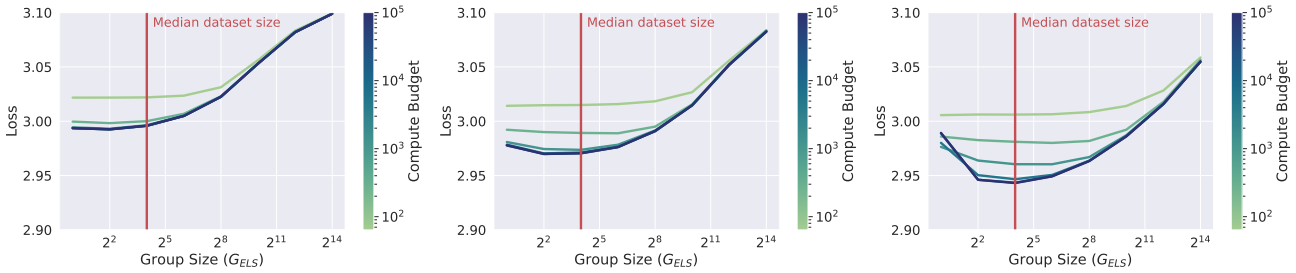


(a) $\varepsilon = 1$. (b) $\varepsilon = 4$. (c) $\varepsilon = 16$.

Figure 9: Loss of ELS on Stack Overflow for varying ε and $G_{\texttt{ELS}}$. The median user dataset size is plotted vertically.

I. Configuring DP-SGD-ULS

In Section 4 we discussed the problem of how to set the group size parameter G_{ULS} for ULS. We expand on our discussion, and give a heuristic for selecting G_{ULS} for a given compute budget. Recall that in Section 3, we showed that the variance of the additive noise in Algorithm 2, which we denote v_{ULS}, satisfies $v_{\text{ULS}} \propto L_{\text{ULS}}\sigma_{\text{ULS}}$, where $\propto$ denotes proportionality, L_{ULS} is the maximum per-user gradient norm, and σ_{ULS} is the noise multiplier. We use v_{ULS} as a proxy for downstream performance.



(a) $\varepsilon = 1$. (b) $\varepsilon = 4$. (c) $\varepsilon = 16$.

Figure 10: Loss of ELS on CC-News for varying ε and $G_{\texttt{ELS}}$. The median user dataset size is plotted vertically.

As discussed in Section 3, the quantity L_{ULS} is a function of G_{ULS}, which we denote $L_{\text{ULS}}(G_{\text{ULS}})$. To see this, note that each user-level gradient is an average over (at most) G_{ULS} example gradients. If, for example, every user has completely orthonormal example gradients, then $L_{\text{ULS}}(G_{\text{ULS}}) = 1/\sqrt{G_{\text{ULS}}}$. The exact dependence of L_{ULS} on G_{ULS} is data-dependent, but can be estimated as a function of G_{ULS} by computing the norm of user-level gradients across the dataset. We do this as follows. We first sample some set U of users. For each user $u \in U$, we randomly select a subset of its dataset of size (at most) G_{ULS}. For each such user, we then compute

$$\rho_u = \left\| \frac{1}{|D_u|} \sum_{z \in D_u} \nabla f(\theta, z) \right\|$$

where θ is the pre-trained model. Let ψ be some statistical estimator on sets of nonnegative real numbers. We can then approximate

$$L_{\text{ULS}}(G_{\text{ULS}}) \approx \psi(\{q_u | u \in U\}). \tag{8}$$

As we discuss in Section 4, we let ψ denote the median, rather than a maximum suggested by (4).

The noise multiplier σ_{ULS} is independent of G_{ULS}, but depends on the sampling probability q in Algorithm 2. In settings with a fixed cohort size M, this means that σ is a function of M, which we denote $\sigma_{\text{ULS}}(M)$. We can compute this function via (3) using DP accounting libraries.

Fixing all other parameters of interest (including the desired privacy level (ε, δ)), the variance v_{ULS} of the noise added in ULS effectively satisfies the following:

$$v_{\text{ULS}}(G_{\text{ULS}}, M) \propto L_{\text{ULS}}(G_{\text{ULS}}) \sigma_{\text{ULS}}(M). \tag{9}$$

Now, say we have a desire compute budget B. To configure ULS, we must select a group size G_{ULS} and cohort size M such that $G_{\text{ULS}} M = B$. By (9), we would like to solve:

$$\min_{G_{\text{ULS}} M = B} L_{\text{ULS}}(G_{\text{ULS}}) \sigma_{\text{ULS}}(M) \tag{10}$$

While we can compute $L(G_{\text{ULS}})$ and $\sigma_{\text{ULS}}(M)$ at individual points, directly optimizing (10) is challenging. Therefore, we consider a conceptually simpler problem: Suppose we are given some G_{ULS} and M, such that $G_{\text{ULS}} M = B$. If we instead wanted to utilize a compute budget of $B' = 2B$, should we use the operating point $G'_{\text{ULS}} = 2G_{\text{ULS}}, M' = M$ or the operating point $G'_{\text{ULS}} = G_{\text{ULS}}, M' = 2M$? This can be answered by computing the following quantities:

$$\tau_G = \frac{L_{\text{ULS}}(2G_{\text{ULS}})}{L_{\text{ULS}}(G_{\text{ULS}})}, \quad \tau_M = \frac{\sigma_{\text{ULS}}(2M)}{\sigma_{\text{ULS}}(M)}. \tag{11}$$

Intuitively, these represent how much we shrink the objective in (10) by doubling G_{ULS} or M, respectively. If $\tau_G < \tau_M$, then we should double G_{ULS}, and otherwise we should double M. We formalize this iterative strategy in Algorithm 3, and refer to it as the "Estimate-and-Double" algorithm.

Algorithm 3 Configuring DP-SGD-ULS via "Estimate-and-Double"

Inputs: Initial group size G^0_{ULS} and cohort size M^0, desired compute budget B.
$G_{\text{ULS}} \leftarrow G^0_{\text{ULS}}, M \leftarrow M^0$.
Estimate $L_{\text{ULS}}(G_{\text{ULS}})$ via (8), compute $\sigma(M)$ via (3).
while $G_{\text{ULS}}M < B$ **do**
  Estimate $L_{\text{ULS}}(G_{\text{ULS}})$ via (8), compute $\sigma(M)$ via (3).
  $\tau_G \leftarrow \frac{L_{\text{ULS}}(2G_{\text{ULS}})}{L_{\text{ULS}}(G_{\text{ULS}})}$, $\tau_M \leftarrow \frac{\sigma(2M)}{\sigma(M)}$.
  if $\tau_G < \tau_M$ **then**
    $G_{\text{ULS}} \leftarrow 2G_{\text{ULS}}$
  else
    $M \leftarrow 2M$
  end if
end while

I.1. Validating Algorithm 3

To determine the efficacy of Algorithm 3, we compute the loss of ULS when fine-tuning on Stack Overflow and CC-News, for a variety of group sizes G_{ULS} and cohort sizes M. We vary these over:

$$G_{\text{ULS}} \in \{2^0, 2^1, \ldots, 2^8\}, \quad M \in \{2^5, 2^6, \ldots, 2^{12}\}, \quad \varepsilon \in \{1, 4, 16, 64\}. \tag{12}$$

The results for Stack Overflow are given in Fig. 11. We note that the anti-diagonals of the heatmaps represent a fixed compute budget. Using this information, we can now see how close the strategy in Algorithm 3 compares to the optimal setting of G_{ULS} and M for a given compute budget (over the aforementioned powers of 2 we sweep over). Note that specifically, we initialize with $G^0_{\text{ULS}} = 1, M^0 = 32$.

To get a better sense of this, we compute, for compute budgets $B \in \{2^5, 2^6, \ldots, 2^{20}\}$, the difference in loss between the optimal setting of G_{ULS} and M, and the following strategies:

- **Greedy Local Oracle**: Given G_{ULS}, M, this strategy has access to an oracle that can compute the fine-tuning loss when setting $G'_{\text{ULS}} = 2G_{\text{ULS}}, M' = M$, and when setting $G'_{\text{ULS}} = G_{\text{ULS}}, M' = 2M$. It then doubles whichever parameter results in a lower loss. Note that while this is not computationally tractable, it serves as a useful lower bound on the effectiveness of any local strategy.
- **Estimate-and-Double**: This is the strategy described by Algorithm 3, starting with $G^0_{\text{ULS}} = 2^0, M^0 = 2^5$. We estimate $L_{\text{ULS}}(G_{\text{ULS}})$ by sampling 128 users at random.
- **Random**: Given a compute budget B, this selects a random G_{ULS}, M from (12) such that $G_{\text{ULS}}M = B$.
- **Max Cohort:** This picks G_{ULS}, M from (12) with a maximum value of M such that $G_{\text{ULS}}M = B$.
- **Max Group Size:** This picks G_{ULS}, M from (12) with a maximum value of G_{ULS} such that $G_{\text{ULS}}M = B$.

The results are given in Figures 12 and 13. Note that suboptimality here refers to the difference in loss between DP-SGD-ULS, when configured using one of the strategies above, versus when it is configured optimally. We see that Algorithm 3 does well across compute budgets and ε, for both datasets, and performs as well as the oracle strategy for most compute budgets.

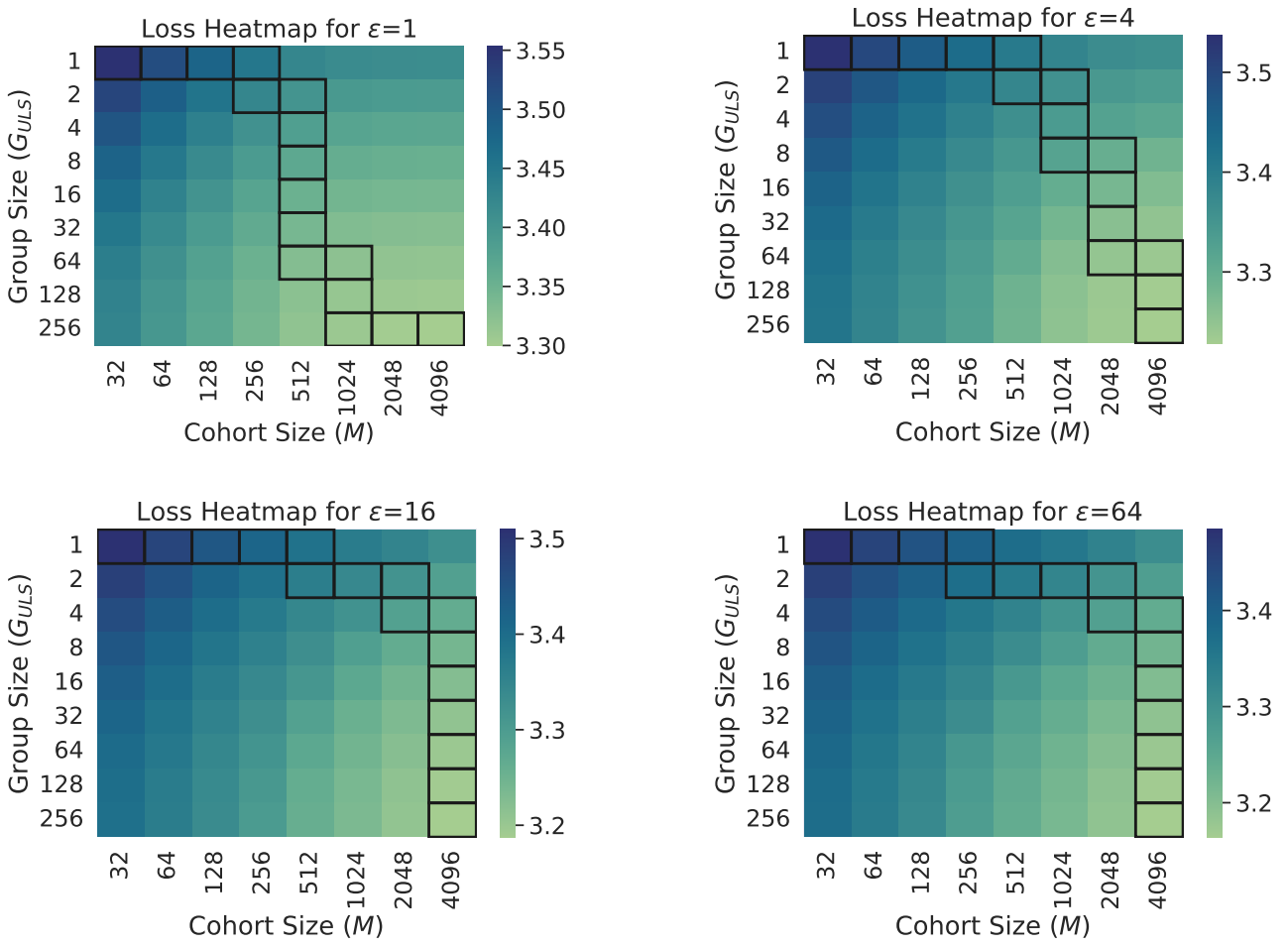


Figure 11: Loss heatmaps of ULS on Stack Overflow for varying group size G_{ULS}, cohort size M, and ε. Optimal settings of M and G_{ULS} for each compute budget are highlighted in black.

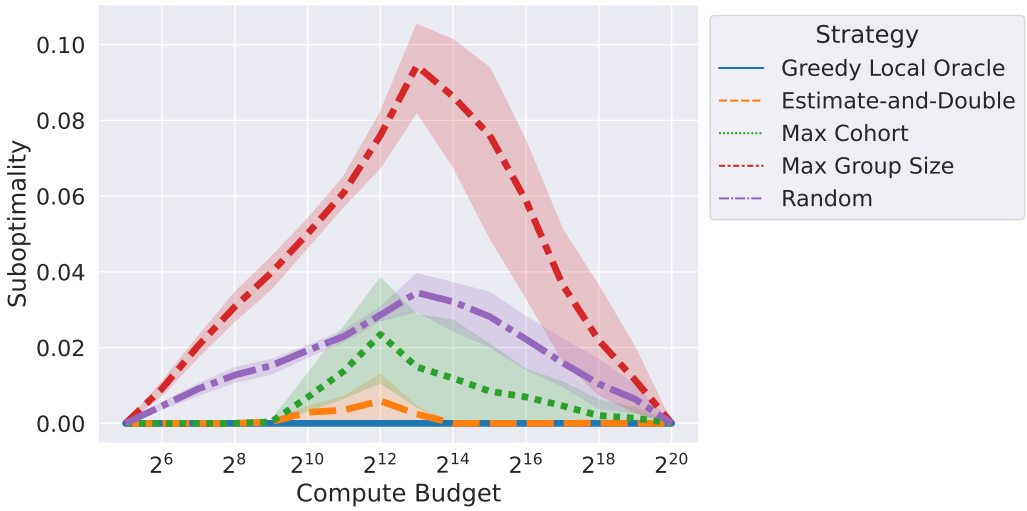


Figure 12: Sub-optimality (in terms of loss) for various strategies used to configure G_{ULS}, M in DP-SGD-ULS on Stack Overflow, for varying compute budgets. Results are averaged across $\varepsilon \in \{1, 4, 16, 64\}$, and opaque areas represent the standard deviation.

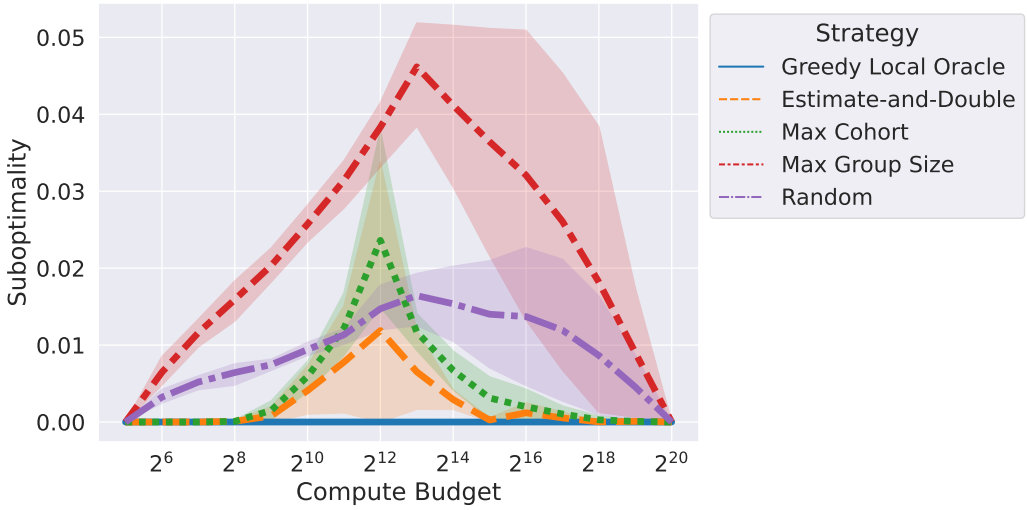


Figure 13: Sub-optimality (in terms of loss) for various strategies used to configure G_{ULS}, M in DP-SGD-ULS on CC-News, for varying compute budgets. Results are averaged across $\varepsilon \in \{1, 4, 16, 64\}$, and opaque areas represent the standard deviation.

J. Additional Experimental Results

J.1. Compute-Loss Tradeoffs

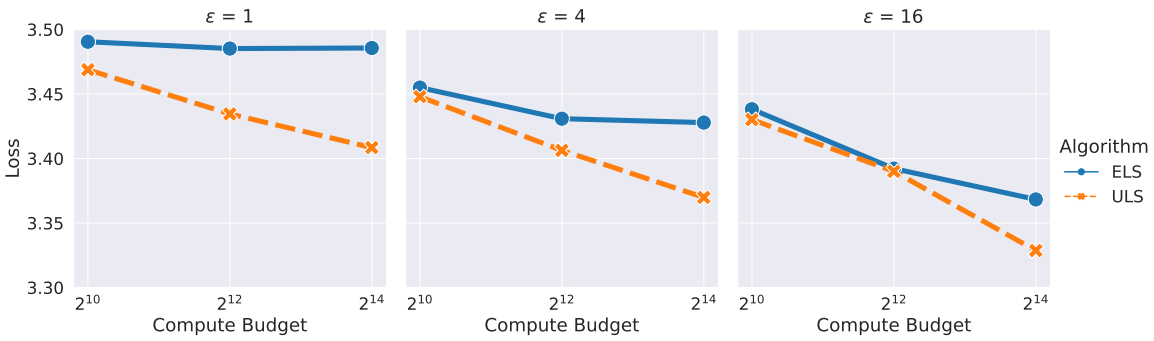


Figure 14: Compute-loss trade-offs on Stack Overflow, for varying privacy levels ε.

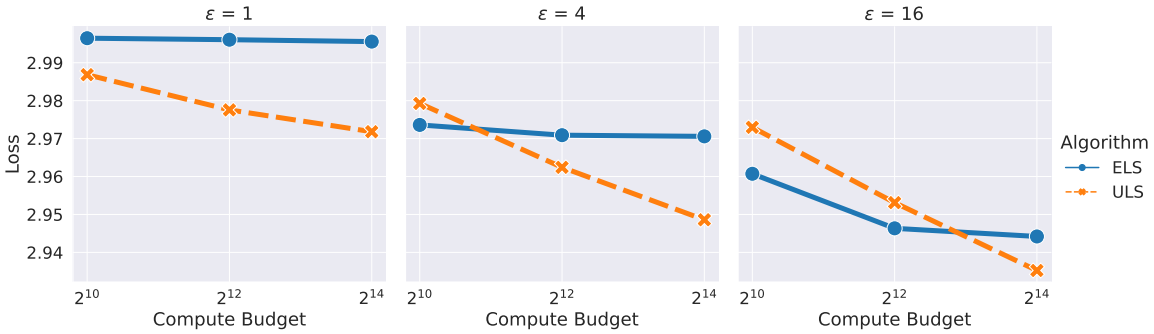


Figure 15: Compute-loss trade-offs on CC-News, for varying privacy levels ε.

In Figures 14 and 15, we present the same information as in Figures 3 and 4 in Section 4, but we instead consider the privacy-compute trade-offs for varying privacy levels ε. We find that ULS is more capable of improving its performance with increased compute budgets. In light of our analysis above, this makes intuitive sense: ELS can only use increased compute budgets to reduce its noise multiplier σ, which has diminishing returns. However, ULS can allocate increased compute budgets to reduce its clip norm C and its noise multiplier σ, trading them off as benefits saturate. We see the same effect in Fig. 2.

J.2. Personalizing Fine-Tuned Models

We take models trained via ELS and ULS, and further personalize them to user data. In general, we are interested in whether the two algorithms exhibit different personalization behavior. A priori, this is plausible, as algorithms that operate at a user-level (including FedAvg (McMahan et al., 2017)) can often exhibit improved personalization performance, even on LLM training tasks (Charles et al., 2023).

To test this, we take our fine-tuned models, and further personalize them on each individual test user's dataset. We compare models fine-tuned via ELS and ULS on the Stack Overflow dataset, and evaluate their personalization ability on the test users, comparing their performance with and without personalization. We do so by taking the Stack Overflow ELS and ULS checkpoints, and further fine-tuning on individual test user datasets. We perform four local epochs of SGD, with a tuned learning rate, on half of each test users' examples. We then evaluate the personalized models on the reserved half of each test users' examples. We record the performance of each model, with and without personalization, on the reserved half of the test users' examples. The results are in Figure 16.

We see that for both algorithms, personalization seems to incur a uniform reduction in loss. However, the gap between using and not using personalization seems to be roughly the same for both model checkpoints.Personalization does not seem to

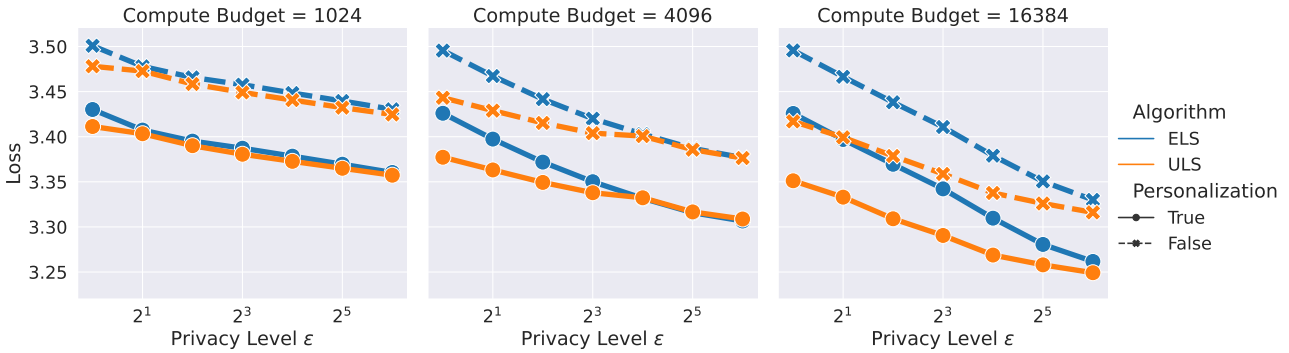


Figure 16: Privacy-loss trade-offs on Stack Overflow, for varying compute budgets, with and without personalization. We present the average loss across all test users on their held-out data.

change the fundamental shape of the trade-off curves. In particular, `ULS` with personalization seems to outperform or match `ELS` for the same privacy levels and compute budgets as without personalization.