Simple Baselines Are Strong Performers for Differentially Private Natural Language Processing

Anonymous Author(s)
Affiliation
Address
email

1 Introduction

Machine learning systems trained on sensitive user data can be vulnerable to privacy attacks [46, 18]. The issue is especially pressing for recent applications involving text [6] powered by ever larger deep learning models, as training data for these systems are often directly derived from private user data, and these models are known capable of memorizing and regurgitating sensitive training examples [8].

As a result, there has been a large interest in developing methods that provide data privacy guarantees for models of text. The gold-standard for providing such a guarantee in machine learning is *Differential Privacy* (DP) [14]. Unfortunately, DP training has typically struggled to produce useful models when applied to large language models, resulting in models with either vacuous privacy guarantees [13] or performance far below non-private baselines. This is widely attributed to the fact that the core primitive of *Differentially Private Stochastic Gradient Descent* (DP-SGD) [47, 1] injects noise that must scale with the number of parameters, resulting in large noise levels for large models [59].

We tackle the problem of building high performing DP language models for sentence classification and language generation tasks with tens to hundreds of thousands of examples. We pursue this goal by re-examining the performance of the baseline DP optimization algorithm for fine-tuning large language models, and study how choices of hyperparameters, task formulation, and pretrained models affect the performance of models given fixed privacy budgets. In contrast to the mainstream perception, our empirical results demonstrate that large pretrained models with hundreds of millions of parameters can be effectively and efficiently fine-tuned to yield models with high performance at stringent privacy levels. For language generation, the performance of our models surpasses strong non-private baselines. For sentence classification, the performance of our fine-tuned models surpasses those obtained under heuristic privacy notions [21] which do not possess formal guarantees. Figure 1 illustrates these results. We summarize our contributions below.

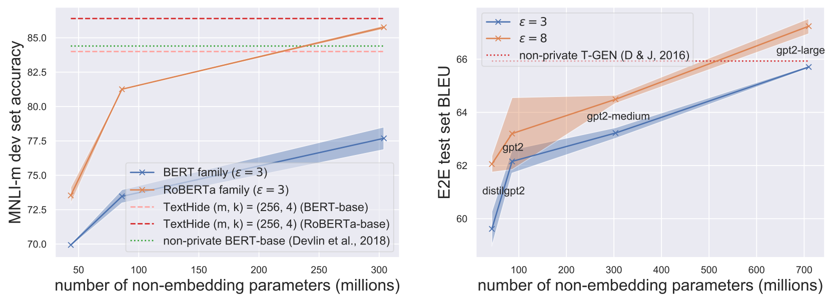
(1) We show that with appropriate hyperparameters and task setup, fine-tuning pretrained language models with DP-Adam yields strong performance for a suite of NLP tasks at stringent privacy levels ($\epsilon \in \{3, 8\}$). Notably, some of our fine-tuned models outperform strong non-private learning baselines and models obtained under heuristic privacy notions.

(2) On the computational side, DP-SGD and DP-Adam can have prohibitive memory cost due to clipping per example gradients. We present a memory saving trick that generalizes the trick by Goodfellow [17] to the case of sequential inputs. Combining this with a recent layer-by-layer clipping procedure [27] enables privately fitting large Transformers [52] with almost the same memory storage as non-private learning at the cost of one additional backward pass per clip.

(3) We show that the dimensionality of gradient updates fails to explain private fine-tuning performance. In contrast to private learning with convex objectives (where high dimensionality degrades performance), we find that larger pretrained models lead to improved private fine-tuning results, and parameter-efficient adaptation methods designed with a reduced dimensionality don't necessarily outperform fine-tuning all parameters.

Empirical results indicate that high performing DP language models at modest privacy budgets can be efficiently trained by directly fine-tuning pretrained models with DP optimization. This enables building practical private NLP models for a range of common tasks where privacy could be at stake.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Sentence classification (MNLI [56]) (b) Language generation (E2E [38])

Figure 1: Fine-tuning pretrained models with DP-Adam yields strong performance when under the right setup. Fine-tuning larger models produces better results. Fine-tuned RoBERTa-base under DP at $\epsilon = 3$ outperforms TextHide (the extension of InstaHide [22] for text classification) with BERT-base. Non-private generation baseline numbers based on that reported by Wiseman et al. [57].

2 Problem Formulation

We build DP models for sentence classification and language generation tasks on small private datasets. We leverage off-the-shelf (public) pretrained language models to simplify the learning problem. We fine-tune these models with DP-Adam [1, 26]. DP optimizers augment usual optimizers by clipping per example gradients with a norm constraint C, and adding Gaussian noise to the clipped gradients whose standard deviation is controlled by C and a noise multiplier σ determined from the privacy budget. Appendix A recaps DP-Adam. We account privacy spending with Rényi DP [34] and detail the procedure in Appendix B. We now describe task setups.

Sentence classification. We fine-tune models of various sizes in the BERT [11] and RoBERTa [30] families, as these masked language models are known to work well for sentence classification in the GLUE [54] benchmark. Each example/record here consists of some input sentences and a label.

Language Generation. We fine-tune the autoregressive GPT-2 of various sizes [45], as this model family is known to work well for generation. The tasks we consider have training sets that are grouped into records. For table-to-text generation tasks such as E2E [38] and DART [36], each record in the training set consists of a pair of table entry and corresponding text description to predict.

3 Effective Differentially Private Language Model Adaptation

By studying the impact of hyperparameters and task design, we demonstrate that the performance of the basic DP-Adam baseline can be substantially improved, even matching some strong non-private baselines. Our analyses also reveal common failure modes and explain poor results reported in past works that consider DP optimization as baselines.

3.1 Good DP Language Models Require Good Hyperparameters

DP optimization is sensitive to the choice of hyperparameters [41]. Our experiments suggest that performance can vary from being close to trivial with ill-chosen hyperparameters to near past state-of-the-arts with appropriately chosen ones. As a consequence, we present simple but widely applicable guidelines on setting the most important hyperparameters. Unless otherwise stated, the unmentioned hyperparameters are set to defaults documented in Appendix H.

Batch Size & Learning Rate. Batch size is one of the most important hyperparameters in our experience. We focus on a setting where the number of training epochs is fixed. This settings roughly corresponds to when the total compute budget is fixed in a non-data-parallel setting.[1] In this setup, the learning rate and batch size jointly affects performance, since using larger batches implies performing fewer gradient updates. To study this joint influence, we fine-tune GPT-2 on the E2E dataset for table-to-text generation with DP-Adam at $\epsilon = 3$ with various batch sizes and learning rates. Figure 2 shows that the best performing models are obtained with both a large batch size and large learning rate. Using a small learning rate together with a small batch size yields considerably worse results. Note a seq2seq baseline achieves a test BLEU of ~65 without privacy on this task [57].

[1]This is appropriate for large models as they tend to be fine-tuned with small micro-batches combined with gradient accumulation; the number of backpropagation passes is roughly constant with respect to the batch size employed for gradient updates.

Recall in the non-private world, pretrained language models are typically fine-tuned with small batch sizes and small learning rates with Adam (bottom left panel in Figure 2). [2] This implies that naïely fine-tuning pretrained language models privately using the non-private setup would result in more performance degradation than necessary. On the other hand, Tramèr and Boneh [51] studied how the batch size and learning rate jointly affect the performance of image classification while holding other hyperparameters fixed. They heuristically suggested a *linear scaling rule*: Scaling the learning rate together with the batch size by the same constant should yield models with almost the same performance. However, Figure 2 indicates that this fails to hold consistently as it falsely predicts that large batch and high learning rate (top right most entry) would have equal performance to small batch and low learning rate (bottom left entry). We explain why linear scaling fails for small batches in Appendix K.

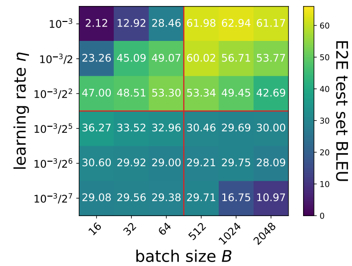


Figure 2: Large batch sizes and learning rates lead to the best performance.

Clipping Norm & Learning Rate. DP optimization is sensitive to the clipping norm C. Since the scale of noise depends on this clipping norm (recall its standard deviation is $C\sigma$), picking C much larger than the actual gradient norm implies more noise is being applied than necessary. In practice, we found that a small clipping norm which enforces almost all gradients to be clipped throughout training leads to the best performance when accompanied by a large learning rate. Figure 3 demonstrates this on the E2E dataset. This finding also explains the poor performance of full fine-tuning baselines in recent works [60].[3]

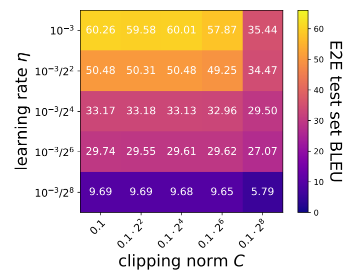


Figure 3: Small clipping norm with large learning rate is performant.

3.2 Improving the Task Alignment Helps Private Learning

Our fine-tuned models on language generation tasks worked well since the pretraining objective and downstream task are *aligned*: Both involve predicting sequences of tokens drawn from some corpus. This alignment simplified the task and benefitted private learning.

For sentence classification tasks, this alignment does not naturally occur. Recall the standard approach for adapting masked language models [11, 53] for classification involves stacking a freshly initialized net on top of the encoding of a special `[CLS]` token and jointly optimizing all parameters [11]. This workflow introduces a discrepancy between pretraining and fine-tuning: Pretraining predicts masked out words that belong to a large vocabulary whereas fine-tuning predicts integer labels.

To avoid this, we instead consider learning to predict the missing word during fine-tuning. For example, for sentiment classification, we reframe the problem as filling in the `[MASK]` token in the sequence "`<INPUT>`. It is `[MASK]`." and compare the probabilities of words "awesome" and "terrible". This text infilling task is almost exactly the procedure used for pretraining masked language models, and recent works have demonstrated its effectiveness for knowledge probing [43], few-shot learning [15] and multi-task fine-tuning [55]. We study how this affects private learning. Table 1 shows that this text-infilling objective brings strong performance gains.

4 Ghost Clipping: Clipping Without Instantiating per Example Gradients

DP-SGD and DP-Adam are memory costly due to per example gradient clipping. Naïvely implemented, this step instantiates giant gradient vectors for each example in a batch during optimization. This is prohibitively expensive for large language models. We present a memory trick that extends the trick by Goodfellow [17] to handle sequential data and can be combined with a recent clipping procedure by Lee and Kifer [27] that does not instantiate entire gradients. This trick enables fitting large Transformers [52] under DP with almost the same memory cost as non-private training, at the expense of an extra backprop pass per clipping step. Due to space constraint, we only give an abridged overview and guide the reader to Appendix C for the complete exposition. What distinguishes our trick from past work is how we compute per example gradient norms for linear/embedding layers.

[2] While the same learning rate might mean very different things for SGD with and without gradient clipping, this issue is less relevant for Adam which self-adjusts the scale of updates with its accumulated second moments.

[3] For instance, Yu et al. [60] included DP full fine-tuning RoBERTa as baseline with $C = 10$ and report much worse results than ours ($C = 0.1$); hyperparameters in their work obtained via private communication.

Consider a linear layer (bias omitted) with input $a \in \mathbb{R}^{B\times T\times d}$, weight matrix $W \in \mathbb{R}^{p\times d}$, and gradient with respect to outputs $g \in \mathbb{R}^{B\times T\times p}$, where B is the batch size, T is the sequence length, d and p are the input and output dimensions. The per example norms of gradients for this layer can be reformulated as $\|\nabla_W \mathcal{L}_i\|_\mathrm{F}^2 = \mathrm{vec}(a_i a_i^\top)^\top \mathrm{vec}(g_i g_i^\top)$. Note $a_i a_i^\top, g_i g_i^\top \in \mathbb{R}^{T\times T}$, and thus when implemented with usual primitives, the memory cost now is $\mathcal{O}(BT^2)$ as opposed to $\mathcal{O}(Bpd)$ before when $\{\nabla_W \mathcal{L}_i\}_i$ are naïvely instantiated. When $T = 1$, this is the Goodfellow [17] trick. Our trick is especially relevant for large Transformers, since these models tend to have large embedding layers ($d \gg T$) and is a major source of memory spending. Figure 4 confirms our trick yields substantial savings compared to existing approaches.

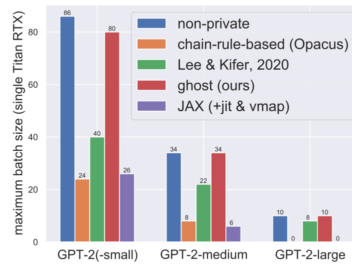


Figure 4: Training with our ghost clipping is 3 times more efficient than `Opacus` and is almost as efficient as non-private learning in terms of memory for typical sequences across model sizes.

5 Low Dimensional Updates Are Not Necessarily Better

The typical privatization procedure for gradients injects isotropic noise that leads to large noise levels for large models and dense fine-tuning updates. The aim of this section is to test if the dimensionality is indicative of final performance. Due to space constraint, we provide an outline of our empirical findings here and refer the reader to Appendix D for the full exposition.

We focus on answering two questions: (1) Do larger pretrained models lead to better or worse private performance? (2) Do adaptation methods designed with a reduced dimensionality of updates outperform full fine-tuning all parameters? Empirical results suggest that larger pretrained models consistently lead to better private learning results across sentence classification (see Table 1) and language generation tasks. Regarding the second question, experiments across various adaptation approaches show that there is no general relationship between the dimensionality of updates and final performance. Moreover, full fine-tuning generally has strong performance across different tasks.

Table 1: With larger pretrained models, full fine-tuning consistently leads to better differentially private models for sentence classification. RGP [60] is an approach that reduces dimensionality of updates by projection. Numbers are dev set accuracies.

Model	$\epsilon = 3$				$\epsilon = 8$			
	MNLI-(m/mm)	QQP	QNLI	SST-2	MNLI-(m/mm)	QQP	QNLI	SST-2
RGP (RoBERTa-base)	-	-	-	-	80.5/79.6	85.5	87.2	91.6
Full (RoBERTa-base)	79.33/79.82	83.30	83.06	86.35	80.05/80.51	83.97	83.68	88.82
Full + infilling (RoBERTa-base)	81.28/82.06	84.06	86.41	92.78	81.84/82.62	84.61	86.85	92.78
Full + infilling (RoBERTa-large)	85.69/86.16	85.60	90.29	94.04	86.14/86.48	86.04	90.65	94.38

6 Scope and Limitation

We have presented strategies for effectively and efficiently fine-tuning large pretrained language models under DP for building high performing private NLP models. Our empirical results suggest that DP isn't as impractical a notion of privacy for building NLP systems as many have believed. For researcher and practitioners working on building private NLP models with datasets of modest sizes, these results suggest that DP fine-tuning with a proper setup is perhaps worth a serious try before prematurely shifting to less formal notions of privacy which have not stood (or may not stand) against the test of time. Below we list some limitations and unsolved questions.

Dimensionality vs Performance. Empirical results on scaling the model size suggest that higher dimensional updates do not necessarily hurt performance. A better characterization of the learning dynamics is likely needed before we may have a complete understanding of these observations.

Scaling Laws for Private Learning. While scaling laws [24] for non-private learning have become prevalent, we are unaware of a case study in private learning. Studies on how the dimensionality of models affects private learning in precise quantitative terms will likely be both useful for practitioners and an interesting theoretical endeavor on its own.

References

[1] 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*, pages 308–318, 2016.

[2] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. Large-scale differentially private bert. *arXiv preprint arXiv:2108.01624*, 2021.

[3] Raef Bassily, Vitaly Feldman, Kunal Talwar, and Abhradeep Thakurta. Private stochastic convex optimization with optimal rates. *arXiv preprint arXiv:1908.09970*, 2019.

[4] Rishi Bommasani, Steven Wu, and Xanda Schofield. Towards private synthetic text generation. In *NeurIPS 2019 Machine Learning with Guarantees Workshop*, 2019.

[5] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[7] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 267–284, 2019.

[8] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*, 2020.

[9] Dingfan Chen, Tribhuvanesh Orekondy, and Mario Fritz. Gs-wgan: A gradient-sanitized approach for learning differentially private generators. *arXiv preprint arXiv:2006.08265*, 2020.

[10] Maximin Coavoux, Shashi Narayan, and Shay B Cohen. Privacy-preserving neural representations of text. *arXiv preprint arXiv:1808.09408*, 2018.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

[13] Christophe Dupuy, Radhika Arava, Rahul Gupta, and Anna Rumshisky. An efficient dp-sgd mechanism for large scale nlp models. *arXiv preprint arXiv:2107.14586*, 2021.

[14] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[15] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

[16] Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. The gem benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:2102.01672*, 2021.

[17] Ian Goodfellow. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*, 2015.

[18] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Membership inference attacks against generative models. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, volume 2019, pages 133–152. De Gruyter, 2019.

[19] Shlomo Hoory, Amir Feder, Avichai Tendler, Alon Cohen, Sofia Erell, Itay Laish, Hootan Nakhost, Uri Stemmer, Ayelet Benjamini, Avinatan Hassidim, et al. Learning and evaluating a differentially private pre-trained language model. In *Proceedings of the Third Workshop on Privacy in Natural Language Processing*, pages 21–29, 2021.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[21] Yangsibo Huang, Zhao Song, Danqi Chen, Kai Li, and Sanjeev Arora. Texthide: Tackling data privacy in language understanding tasks. *arXiv preprint arXiv:2010.06053*, 2020.

[22] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International Conference on Machine Learning*, pages 4507–4518. PMLR, 2020.

[23] P Kairouz, M Ribero, K Rush, and A Thakurta. Fast dimension independent private adagrad on publicly estimated subspaces. *arXiv preprint arXiv:2008.06570*, 2020.

[24] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[25] Gavin Kerrigan, Dylan Slack, and Jens Tuyls. Differentially private language models benefit from public pre-training. *arXiv preprint arXiv:2009.05886*, 2020.

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

[27] Jaewoo Lee and Daniel Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *arXiv preprint arXiv:2009.03106*, 2020.

[28] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

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

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[31] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[32] Oren Melamud and Chaitanya Shivade. Towards automatic generation of shareable synthetic clinical notes using neural language models. *arXiv preprint arXiv:1905.07002*, 2019.

[33] Fatemehsadat Mireshghallah, Huseyin A Inan, Marcello Hasegawa, Victor Rühle, Taylor Berg-Kirkpatrick, and Robert Sim. Privacy regularization: Joint privacy-utility optimization in language models. *arXiv preprint arXiv:2103.07567*, 2021.

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

[35] Ilya Mironov, Kunal Talwar, and Li Zhang. R\\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

[36] Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*, 2020.

[37] Marcel Neunhoeffer, Zhiwei Steven Wu, and Cynthia Dwork. Private post-gan boosting. *arXiv preprint arXiv:2007.11934*, 2020.

[38] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*, 2017.

[39] Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

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

[41] Nicolas Papernot, Steve Chien, Shuang Song, Abhradeep Thakurta, and Ulfar Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy. 2019.

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[43] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[44] Chen Qu, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork. Privacy-adaptive bert for natural language understanding. *arXiv preprint arXiv:2104.07504*, 2021.

[45] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[46] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[47] 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*, pages 245–248. IEEE, 2013.

[48] Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. Enabling fast differentially private sgd via just-in-time compilation and vectorization. *arXiv preprint arXiv:2010.09063*, 2020.

[49] Amirsina Torfi, Edward A Fox, and Chandan K Reddy. Differentially private synthetic medical data generation using convolutional gans. *arXiv preprint arXiv:2012.11774*, 2020.

[50] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[51] Florian Tramèr and Dan Boneh. Differentially private learning needs better features (or much more data). *arXiv preprint arXiv:2011.11660*, 2020.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[53] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[54] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[55] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[56] Adina Williams, Nikita Nangia, and Samuel R Bowman. The multi-genre nli corpus. 2018.

[57] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. Learning neural templates for text generation. *arXiv preprint arXiv:1808.10122*, 2018.

[58] Zekun Xu, Abhinav Aggarwal, Oluwaseyi Feyisetan, and Nathanael Teissier. On a utilitarian approach to privacy preserving text generation. *arXiv preprint arXiv:2104.11838*, 2021.

[59] Da Yu, Huishuai Zhang, Wei Chen, and Tie-Yan Liu. Do not let privacy overbill utility: Gradient embedding perturbation for private learning. *arXiv preprint arXiv:2102.12677*, 2021.

[60] Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. Large scale private learning via low-rank reparametrization. *arXiv preprint arXiv:2106.09352*, 2021.

[61] Huanyu Zhang, Ilya Mironov, and Meisam Hejazinia. Wide network learning with differential privacy. *arXiv preprint arXiv:2103.01294*, 2021.

[62] Yingxue Zhou, Zhiwei Steven Wu, and Arindam Banerjee. Bypassing the ambient dimension: Private sgd with gradient subspace identification. *arXiv preprint arXiv:2007.03813*, 2020.

A DP-Adam

We use DP-Adam throughout. DP-Adam works just like regular Adam [26] but performs updates and moment accumulation with privatized gradients. The gradient privatization part is the same as that performed in DP-SGD [47, 1]. To determine the noise multiplier, we account privacy through Rényi differential privacy (RDP) [34, 35]. For completeness, we include the pseudocode below.

Algorithm 1 DP-Adam

Input: Data $\mathcal{D} = \{x_i\}_{i=1}^N$, learning rate η, noise multiplier σ, batch size B, Euclidean norm threshold for gradients C, epochs T, initial parameter vector $\theta_0 \in \mathbb{R}^p$, initial moment estimates $m_0, v_0 \in \mathbb{R}^p$, exponential decay rates $\beta_1, \beta_2 \in \mathbb{R}$, avoid division-by-zero constant $\gamma \in \mathbb{R}$.
for $t \in [T \cdot {}^N\!/_B]$ **do**
  Draw a batch B_t via Poisson sampling; each element has probability ${}^B\!/_N$ of being selected
  for $x_i \in B_t$ **do**
    $g_t(x_i) \leftarrow \nabla_{\theta_t} \mathcal{L}(x_i), \quad \tilde{g}_t(x_i) \leftarrow g_t(x_i) \cdot \min(1, {}^C\!/_{\|g_t(x_i)\|_2})$
  end for
  $z_t \sim \mathcal{N}(0, \sigma^2 C^2 I_p)$
  $\bar{g}_t = \frac{1}{B}\left(\sum_{i=1}^N \tilde{g}_t(x_i) + z_t\right)$
  $\theta_{t+1}, m_{t+1}, v_{t+1} \leftarrow \text{AdamUpdate}(\theta_t, m_t, v_t, \bar{g}_t, \beta_1, \beta_2, \gamma)$
end for
return $\theta_{TN/B}$

Algorithm 2 AdamUpdate

Input: $\theta_t, m_t, v_t, \bar{g}_t, \beta_1, \beta_2, \gamma$
$m_{t+1} \leftarrow \beta_1 \cdot m_t + (1-\beta_1) \cdot \bar{g}_t, \quad v_{t+1} \leftarrow \beta_2 \cdot v_t + (1-\beta_2) \cdot \bar{g}_t{}^2$
$\widehat{m}_{t+1} \leftarrow m_{t+1} / \left(1-\beta_1^t\right), \quad \widehat{v}_{t+1} \leftarrow v_{t+1} / \left(1-\beta_2^t\right)$
$\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \widehat{m}_{t+1} / \left(\sqrt{\widehat{v}_{t+1}} + \gamma\right)$
return $\theta_{t+1}, m_{t+1}, v_{t+1}$

B Privacy Accounting

We train all models under approximate-DP [14], and we view two datasets as being adjacent if and only if one can be obtained from the other by including an extra record [35]. Instead of accounting the privacy loss with Moments Accountant [1], we perform computation through (i) Rényi Differential Privacy (RDP) [34, 35], and (ii) Gaussian Differential Privacy (GDP) [12] with an associated central limit theorem. Both approaches are improvements over the Moments Accountant. Accounting loss with RDP provides strict upper bounds on the actual privacy leakage, whereas accounting with GDP and its central limit theorem, although asymptotically exact, only provides approximations to the actual loss under a finite number of compositions [12, Theorem 3.4].

Given the noise multiplier σ, sampling rate q, number of steps T, and δ, ϵ can be computed via first computing the Rényi DP leakage and then converting it to approximate DP. When a privacy budget ϵ is prescribed, we can numerically invert the above procedure to obtain a suitable σ for noisy optimization. This is what we do throughout all experiments. For completeness, given σ's chosen as above, we also report the ϵ_{GDP} estimated by going through the central limit theorem in Gaussian DP.

In addition, model selection from hyperparameter tuning on private training data could incur extra privacy leakage. We skip the step of private selection [29] and instead perform tuning only on the E2E task and reuse almost the exact hyperparameters for remaining tasks.

C Ghost Clipping: Clipping Without Instantiating per Example Gradients

DP-SGD and DP-Adam are memory costly due to their *per example* gradients clipping. Naïvely implemented, this step instantiates a giant gradient vector for each example in a batch during optimization, which is often prohibitively expensive for large language models. For example, Hoory et al. [19] pretrained BERT with DP optimization and reported memory issues when using large batch sizes that are necessary for handling noisy gradients.

A simple yet time costly solution to the memory problem is micro-batching: Split large batches into multiple smaller ones and aggregate the results after processing each small batch individually [51, 19]. This solution, however, is unlikely to be sufficient as neural language models become even larger and it becomes difficult to even fit a few copies of the gradient.

We argue that per example gradients need not be instantiated at all, if the goal is only to clip the gradients. Leveraging this insight, Lee and Kifer [27] presented a clipping procedure that only instantiates the per example gradient for a *single* layer of the model one at a time, as opposed to the entire model at once. We call this approach *ghost clipping*, as the per example gradient is much like the ghost that would never be explicitly instantiated.

Unfortunately, we find that this trick can still be insufficient for sequence models such as Transformers, as the memory requirement for per example gradients in embedding layers (and language modeling heads) can be costly. Here, we extend this approach and present a specialized version for Transformers such that DP-SGD and DP-Adam can be ran with almost the same peak memory consumption as non-private training, at the additional cost of an extra backpropagation pass. We anticipate this extension to be useful for both privately fine-tuning and pretraining ever larger Transformers.

C.1 Ghost Clipping

We briefly recap the approach by Lee and Kifer [27]. Note that per example gradient clipping is easy if we had access to the per example norms. In this case, we first compute the scaling factor $c_i = \min(1, C/\|\nabla \mathcal{L}_i\|_2)$, where C is the clipping threshold and $\mathcal{L}_i$ is the loss associated with the ith example. Then, we would perform the usual backward pass with the reweighted loss $\sum_i c_i \mathcal{L}_i$ which is a scalar.

With this in mind, the remaining difficulty becomes computing the gradient norm $\|\nabla \mathcal{L}_i\|_2$. We emphasize two core technicalities that enable computing this quantity without instantiating the full per example gradient $\nabla \mathcal{L}_i$.

First, for a common neural net layer l with parameters $W^{(l)}$ (without parameter sharing), the per example gradient w.r.t. parameters can be easily computed using the input to the layer $a^{(l)}$ and the gradient of the loss w.r.t. the output $g^{(l)}$, both of which are well available during a typical backward pass in autodiff libraries. For instance, for a linear layer with non-sequential input $a^{(l)} \in \mathbb{R}^{B \times d_l}$ and gradient w.r.t. output $g^{(l)} \in \mathbb{R}^{B \times d_{l+1}}$ (B, d_l, and d_{l+1} are the batch size, input and output dimensions, respectively), the per example gradient w.r.t. weights of the layer $\nabla_{W^{(l)}} \mathcal{L} \in \mathbb{R}^{B \times d_l \times d_{l+1}}$ is simply the following batched outer product:

$$(\nabla_{W^{(l)}} \mathcal{L})_{i,j,k} = \left(a^{(l)}\right)_{i,j} \left(g^{(l)}\right)_{i,k}. \tag{1}$$

Second, for a large vector formed by concatenating several small vectors $u = [u_1, \ldots, u_k]$, its Euclidean norm is simply the norm of the vector of norms, i.e.

$$\|u\|_2 = \|[u_1, \ldots, u_k]\|_2 = \|(\|u_1\|_2, \ldots, \|u_k\|_2)\|_2. \tag{2}$$

The second point means that computing the per example gradient norm $\|\nabla \mathcal{L}_i\|_2$ can be done by computing the per example gradient norms for individual layers of the neural net $\|\nabla_{W^{(1)}} \mathcal{L}_i\|_2, \ldots, \|\nabla_{W^{(L)}} \mathcal{L}_i\|_2$ one at a time. Moreover, the first point implies that the norms for each layer can be computed using quantities freely available to a typical backward pass. Overall, this means computing the per example gradient norm can be done in a layer-by-layer fashion if the network does not adopt parameter sharing, with only one per example gradient tensor for a single layer of the network being instantiated at once.

C.2 Ghost Clipping for Transformers With Sequential Data

Vanilla ghost clipping still requires instantiating the per example gradient of individual layers (although not simultaneously). This may become problematic in terms of memory for Transformers with embedding layers that have large vocabularies. Here, we present a specialized procedure for computing the per example gradient norm for linear and embedding layers[4] when they are applied to sequential data. This procedure reduces time and memory complexity and can be viewed as a generalization of the trick by Goodfellow [17] that additionally handles sequential inputs.

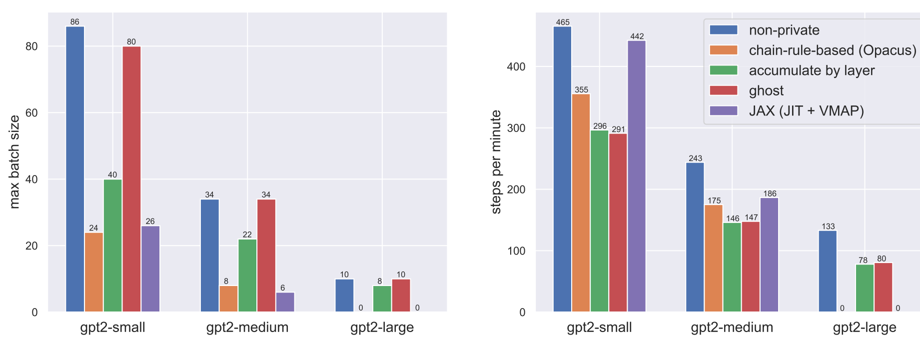
Let $a \in \mathbb{R}^{B \times T \times d}$ be the input to a linear layer with weight matrix $W \in \mathbb{R}^{p \times d}$, and $s \in \mathbb{R}^{B \times T \times p}$ be the output with $s_{i,j} = W a_{i,j}$. Let $g \in \mathbb{R}^{B \times T \times p}$ be the gradient of the loss w.r.t. the output s. Simple calculation shows that the per example gradient is the product of two matrices:

$$\nabla_W \mathcal{L}_i = g_i^\top a_i \in \mathbb{R}^{p \times d}. \tag{3}$$

Since the per example gradient norms are the end goal, the per example gradients $\{\nabla_W \mathcal{L}_i\}_{i=1}^B$ themselves need not be instantiated explicitly. More precisely, we observe that the squared Frobenius norm $\|\nabla_W \mathcal{L}_i\|_{\mathrm{F}}^2$ obeys the following identity:

$$\|\nabla_W \mathcal{L}_i\|_{\mathrm{F}}^2 = \mathrm{vec}(a_i a_i^\top)^\top \mathrm{vec}(g_i g_i^\top), \tag{4}$$

where $a_i a_i^\top, g_i g_i^\top \in \mathbb{R}^{T \times T}$; see Appendix G for a derivation. Implemented with the usual primitives in machine learning libraries, (4) has an asymptotic memory complexity of order $\mathcal{O}(BT^2)$, as opposed to the naïve approach which goes through instantiating (3) and is of order $\mathcal{O}(Bpd)$ in terms of memory. The memory saving of this procedure is most exemplified for off-the-shelf pretrained language models which have large vocabularies. For GPT-2, $d \approx 50,000$ and $p = 768$ for the embedding layer, and the context window $T \leq 1024$.[5] Our method in theory reduces the memory cost due to large embeddings by a factor of 22. In practice, we observe significant savings for most pretrained models which generally are bottlenecked by large embedding layers.[6] We compare ghost clipping implemented using (4) (in PyTorch) with a JAX implementation that clips by instantiates per-sample gradients powered by `jit` and `vmap`, `Opacus`, ghost clipping without using (4), and non-private training in PyTorch. Figure 5 (a) shows that for typical inputs, our trick is the most memory friendly and allows fitting batches almost as large as non-private training. Setup for this experiment is detailed in Appendix I.



(a) Memory (b) Time

Figure 5: **Left:** Training with ghost clipping is 3 times more efficient than `Opacus` and is almost as efficient as non-private learning in terms of memory for typical sequences across model sizes. For GPT2-large, we were unable to fit a single example with the typical length using `Opacus` or DP-Adam implemented in JAX on a TITAN RTX GPU (24 GBs of VRAM). **Right:** Per-update, training with ghost clipping is at most 20% slower than `Opacus` and 50% slower than non-private for the typical batch across model sizes.

D Low Dimensional Updates Are Not Necessarily Better

Theory on DP ERM with convex objectives suggests that the error of private learning degrades with the dimensionality of a parametric model ($\mathcal{O}(1/\sqrt{n} + \sqrt{p}/n\epsilon)$ in precise terms for a p dimensional

[4] An embedding layer is essentially a linear layer: The embedding lookup operation applied to indices is equivalent to a matrix multiplication of the embedding matrix with one-hot encoded indices.

[5] In practice, for fine-tuning tasks, the average sequence length is much shorter.

[6] While there are alternative approaches for reducing the memory footprint of the embedding layer during training, these methods typically introduce extra hyperparameters that would require tuning and therefore privacy spending.

model trained on n examples [3]). The aim of this section is to test if this statement remains valid in the realm of DP fine-tuning, and if so, to what extent. We focus on answering two questions: (1) Do larger pretrained models lead to better or worse private performance, and (2) do parameter-efficient adaptation methods designed with a reduced dimensionality of updates outperform full fine-tuning. We study these questions separately below. All reported numbers in this section are the average over three random seeds.

D.1 Larger Pretrained Models Result in Better Performance

We empirically observe that larger pretrained models tend to lead to better private fine-tuning performance. Specifically, we perform the following experiment: We privately fine-tune pretrained models of various sizes at the same privacy budget. To ensure our hyperparameters aren't favoring larger models, we lightly tune on the smallest model and then reuse the same hyperparameters for all fine-tuning workloads. Figure 1 demonstrates our findings.

D.2 The Full Fine-Tuning Baseline Matches State-of-the-Art

There has been a range of lightweight fine-tuning methods that reduce the dimensionality of updates, including some that are specifically designed for differentially private learning [60]. We study whether these low-dimensional methods lead to improvements in performance under DP.

Do methods that optimize fewer parameters lead to better results under DP even if they perform similarly non-privately? The theory of DP ERM on convex objectives suggests that this should be the case. However, empirical results suggest that this is generally not the case for DP fine-tuning, and that full fine-tuning is a strong baseline that matches even specialized low-dimensional differentially private learning methods for both classification and generation. Below, we study the two sets of tasks separately.

D.2.1 Sentence Classification

We study DP fine-tuning on tasks from the GLUE benchmark that have more than 10k training examples (MNLI, QQP, QNLI, and SST-2), following the experimental setup of Yu et al. [60]. The associated datasets have modest sizes: SST-2 and QNLI have 60k+ and 100k+ training examples respectively. MNLI and QQP have larger training sets each containing less than 400k examples. Table 2 shows that both using a larger pretrained model and the text-infilling objective improve classification accuracy. We also compare full fine-tuning with *reparameterized gradient perturbation* (RGP) [60], as it is the state-of-the-art for DP fine-tuning on sentence classification at the time of writing. The method is designed to privatize gradients projected onto low dimensional subspaces and was motivated by the need to reduce DP noise in high-dimensional models. We note that direct fine-tuning with the text infilling objective outperforms well-tuned RGP on all tasks except QQP, despite being one of the simplest baselines. Computationally, while RGP is faster per-update, it requires more than 5 times as many epochs as full fine-tuning – overall, the latter is actually faster in terms of wall time.

Table 2: With better and larger pretrained models, full fine-tuning consistently leads to better differentially private models on a subset of tasks from the GLUE benchmark [54]. Reported numbers are dev set accuracies; MNLI results take the format of matched/mismatched.

Model	$\epsilon = 3$				$\epsilon = 8$			
	MNLI-(m/mm)	QQP	QNLI	SST-2	MNLI-(m/mm)	QQP	QNLI	SST-2
RGP (RoBERTa-base) [60]	-	-	-	-	80.5/79.6	85.5	87.2	91.6
Full (RoBERTa-base)	79.33/79.82	83.30	83.06	86.35	80.05/80.51	83.97	83.68	88.82
Full + infilling (RoBERTa-base)	81.28/82.06	84.06	86.41	92.78	81.84/82.62	84.61	86.85	92.78
Full + infilling (RoBERTa-large)	85.69/86.16	85.60	90.29	94.04	86.14/86.48	86.04	90.65	94.38
ϵ (Gaussian DP + CLT)	1.01	1.04	1.41	1.53	3.03	3.07	3.77	4.01

D.2.2 Table-to-Text Generation

We study different adaptation methods for table-to-text generation tasks where the goal is to generate natural language descriptions of table entries. We consider the datasets E2E [38] and DART [36]. E2E is a simple dataset of restaurant reviews, whereas DART consists of open-domain table entries from Wikipedia and is more complex. For evaluation, we run its official pipeline[7] for E2E and the pipeline reused in the GEM benchmark [16][8] for DART. Both datasets are small from a DP perspective: E2E has 40k+ training examples, whereas DART has roughly 60k.

The methods of comparison are LoRA [20], prefix-tuning [28], RGP, and fine-tuning the top 2 layers (top2), all of which optimize substantially fewer parameters. In particular, on GPT-2 (125 million parameters), prefix-tuning instantiated with its default hyperparameters optimizes roughly 10 million parameters; LoRA with rank 4 optimizes roughly 0.15 million parameters. For completeness, we also report results obtained by training with randomly initialized weights (retrain). Hyperparameters of each method were tuned only the E2E dataset; the complete search ranges are in Appendix F. Tables 3 and 4 show that both LoRA and full fine-tuning are strong performers.

Table 3: Results on E2E by adapting GPT-2. Full fine-tuning is a strong baseline that often outperforms alternative methods which optimize fewer parameters or designed with DP in mind.

Method	DP Guarantee	Metrics				
		BLEU	NIST	METEOR	ROUGE-L	CIDEr
full	$\epsilon = 3$	**61.519**	**6.697**	**0.384**	0.657	**1.761**
	$\epsilon = 8$	63.189	7.444	0.400	0.664	1.919
	non-private	69.463	8.780	0.461	0.714	2.422
LoRA	$\epsilon = 3$	58.153	5.463	0.370	**0.658**	1.581
	$\epsilon = 8$	**63.389**	**7.449**	**0.407**	**0.675**	**1.948**
	non-private	69.682	8.822	0.463	0.717	2.491
prefix-tuning	$\epsilon = 3$	47.772	5.775	0.331	0.590	1.300
	$\epsilon = 8$	49.263	6.276	0.349	0.607	1.496
	non-private	68.845	8.722	0.456	0.708	2.418
RGP	$\epsilon = 3$	58.482	5.249	0.363	0.656	1.507
	$\epsilon = 8$	58.455	5.525	0.364	0.650	1.569
	non-private	68.328	8.722	0.445	0.688	2.345
top2	$\epsilon = 3$	25.920	1.510	0.197	0.445	0.452
	$\epsilon = 8$	26.885	1.547	0.207	0.464	0.499
	non-private	65.752	8.418	0.443	0.687	2.180
retrain	$\epsilon = 3$	15.457	0.376	0.113	0.352	0.116
	$\epsilon = 8$	24.247	1.010	0.145	0.400	0.281
	non-private	65.731	8.286	0.429	0.688	2.004

E Related Work

DP Deep Learning. DP-SGD has been viewed as ineffective for large models due to the addition of large Gaussian noise to gradient updates. Improvements to the learning procedure mostly fall under two distinct camps: (i) Simplifying the private learning problem, and (ii) reducing the scale of noise. For instance, Papernot et al. [41], Tramèr and Boneh [51], Abadi et al. [1] consider transferring features learned on public datasets to simplify the subsequent private learning task. On the other hand, Zhou et al. [62], Kairouz et al. [23] remove the ambient dimension dependence of DP noise by identifying subspaces in which private gradients lie and would be privatized. Yu et al. [59, 60] execute such ideas with tricks and demonstrate improved results on standard private learning benchmarks. Zhang et al. [61] applied the sparse vector technique to learning wide neural layers to reduce the amount of injected noise.

Our work largely falls under the first camp – we study how DP fine-tuning can be made practically effective. Our work is also distinct from prior works in that we focus on privately fine-tuning large pretrained models. Lastly, there are alternative solutions in the literature that enforces DP which are

[7] https://github.com/tuetschek/e2e-metrics

[8] https://github.com/GEM-benchmark/GEM-metrics

Table 4: Results on DART by adapting GPT-2. Trend is consistent with results on E2E.

Method	DP Guarantee	Metrics METEOR	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	BERTScore	BLEURT
full	$\epsilon = 3$	0.294	0.628	**0.408**	**0.521**	31.025	**0.887**	-0.058
	$\epsilon = 8$	**0.319**	**0.664**	**0.436**	**0.546**	**35.057**	**0.901**	**0.043**
	non-private	0.369	0.716	0.472	0.567	42.783	0.915	0.178
LoRA	$\epsilon = 3$	**0.304**	**0.636**	0.408	0.520	**32.329**	0.885	**-0.029**
	$\epsilon = 8$	0.318	0.663	0.431	0.541	34.163	0.899	0.036
	non-private	0.366	0.712	0.473	0.574	42.254	0.915	0.182
prefix-tuning	$\epsilon = 3$	0.269	0.595	0.382	0.494	25.726	0.860	-0.144
	$\epsilon = 8$	0.297	0.640	0.416	0.526	30.463	0.892	-0.021
	non-private	0.353	0.703	0.466	0.569	40.163	0.912	0.148
RGP	$\epsilon = 3$	0.265	0.587	0.372	0.490	25.748	0.873	-0.175
	$\epsilon = 8$	0.279	0.600	0.383	0.498	28.304	0.874	-0.141
	non-private	0.324	0.657	0.426	0.535	35.551	0.895	0.022
top2	$\epsilon = 3$	0.022	0.036	0.022	0.032	0.388	0.098	-1.952
	$\epsilon = 8$	0.054	0.115	0.071	0.100	2.453	0.240	-1.660
	non-private	0.318	0.628	0.384	0.494	36.099	0.883	-0.082
retrain	$\epsilon = 3$	0.064	0.191	0.089	0.171	2.997	0.493	-1.513
	$\epsilon = 8$	0.093	0.250	0.119	0.217	7.765	0.573	-1.302
	non-private	0.232	0.478	0.264	0.379	26.794	0.806	-0.593

not based on gradient perturbation [40, 39]. These methods typically require extra public data and are not the present focus.

Private NLP. Works studying privacy-preserving NLP are largely divided by whether or not they consider a formal notion of privacy based on DP. Under global DP, McMahan et al. [31] successfully train small private word-level RNNs with 1.35 million parameters in a federated learning setting with more than 700k users with a DP guarantee of $(4.6, 10^{-9})$. On the other hand, Qu et al. [44] study fine-tuning BERT for language understand under local DP. Kerrigan et al. [25] demonstrate that public pretraining is helpful for subsequent downstream autoregressive training with DP-SGD, though they did not report results for fine-tuning large pretrained models with DP-SGD. Anil et al. [2] pretrain BERT under global DP on datasets with hundreds of millions of examples. Dupuy et al. [13] study private BERT fine-tuning on datasets of utterances, but report results with ϵ on the order of 100 to 10,000. Orthogonally, many works consider training language models that satisfy empirical notions of privacy [58, 10, 33, 32], either based on relaxed notions of DP or explicitly defending against specific privacy attacks. Our work is distinct from these mentioned works in that we study fine-tuning large language models (hundreds of millions of parameters) under global DP with stringent guarantees ($\epsilon < 10$) on small datasets (much less than a million examples).

DP Synthetic Data Generation. Fine-tuning generative language models on private data under DP can also be viewed as a means of accomplishing DP synthetic data generation – learning generative models from private data so that synthetic examples with similar characteristics similar could be sampled and used downstream. Previous work employed generative adversarial networks and focused mostly on image or tabular datasets [50, 37, 9, 49]. Bommasani et al. [5] briefly commented on the possibility of achieving cheaper private learning by fine-tuning large pretrained language models but did not execute the idea. Perhaps most directly related to our work is that by Bommasani et al. [4] who attempted fine-tuning GPT-2 on medical datasets but did not report quantitative results.

F Hyperparameter Search Ranges for Experiments in Section D

We compare different adaptation methods by reporting task specific metrics on the test split using hyperparameters that maximize validation BLEU on E2E. For sentence classification tasks, we reused the same hyperparameters, except for the number of epochs which we set to be the same as that used in non-private fine-tuning. We list the range of hyperparameters that we searched over for each individual adaptation method considered in the paper. Prefix-tuning has two additional hyperparameters: the length of the prefix and the dimensionality of the hidden layer. We set these to the default used by Li and Liang [28] (5 for the former and 512 for the latter). For Adam, we use the default hyperparamaters set by PyTorch [42].

Table 5: Hyperparameters search range for different methods. $\mathcal{D}_{\text{train}}$ is the training set.

Method	Full	Prefix	Linear	FT2
DP guarantee (ϵ, δ)	$(3, 1/2\lvert\mathcal{D}_{\text{train}}\rvert)$	$(3, 1/2\lvert\mathcal{D}_{\text{train}}\rvert)$	$(3, 1/2\lvert\mathcal{D}_{\text{train}}\rvert)$	$(3, 1/2\lvert\mathcal{D}_{\text{train}}\rvert)$
Clipping norm C	0.1	0.1	0.1	0.1
Batch size B	$\{512, 1024\}$	$\{512, 1024\}$	$\{512, 1024\}$	$\{512, 1024\}$
Learning rate η	$\{10^2, 30, 10, 3\} \cdot 10^{-5}$	$\{10^2, 30, 10, 3\} \cdot 10^{-5}$	$\{10^2, 30, 10, 3\} \cdot 10^{-5}$	$\{10^2, 30, 10, 3\} \cdot 10^{-5}$
Learning rate decay	{yes, no}	{yes, no}	{yes, no}	{yes, no}
Epochs T	$\{10, 30, 50\}$	$\{10, 30, 50\}$	$\{10, 30, 50\}$	$\{10, 30, 50\}$
Weight decay λ	0	0	0	0
Noise scale σ	calculated numerically so that a DP budget of (ϵ, δ) is spent after T epochs[9]			

Table 6: Hyperparameters search range for methods with low-rank updates.

Method	LoRA	RGP
DP guarantee (ϵ, δ)	$(3, 1/2\lvert\mathcal{D}_{\text{train}}\rvert)$	$(3, 1/2\lvert\mathcal{D}_{\text{train}}\rvert)$
Clipping norm C	0.1	$\{0.1, 1, 10\}$
Batch size B	$\{512, 1024\}$	$\{512, 1024\}$
Learning rate η	$\{300, 100, 30, 10, 3\} \cdot 10^{-5}$	$\{300, 100, 30, 10, 3\} \cdot 10^{-5}$
Learning rate decay	{yes, no}	{yes, no}
Epochs T	$\{5, 10, 30, 50\}$	$\{5, 10, 30, 50\}$
Weight decay λ	0	0
Rank k	$\{1, 2, 4, 8\}$	$\{1, 2, 4, 8\}$
Noise scale σ	calculated numerically so that a DP budget of (ϵ, δ) is spent after T epochs	

G Derivation of the Frobenius Norm Identity

Recall $a \in \mathbb{R}^{B \times T \times d}$ is the input to a linear layer with weight matrix $W \in \mathbb{R}^{p \times d}$, and $g \in \mathbb{R}^{B \times T \times p}$ is the gradient of the loss w.r.t. the output. The identity follows from trivial algebra:

$$
\begin{aligned}
\|\nabla_W \mathcal{L}_i\|_{\mathrm{F}}^2 = \left\|g_i^\top a_i\right\|_{\mathrm{F}}^2 &= \left\|\sum_{k=1}^{T} g_{i,k} a_{i,k}^\top\right\|_{\mathrm{F}}^2 \\
&= \sum_{r=1}^{d} \sum_{s=1}^{p} \left(\sum_{k=1}^{T} a_{i,k,r} g_{i,k,s}\right)^2 \\
&= \sum_{r=1}^{d} \sum_{s=1}^{p} \sum_{k_1=1}^{T} \sum_{k_2=1}^{T} a_{i,k_1,r} g_{i,k_1,s} a_{i,k_2,r} g_{i,k_2,s} \\
&= \sum_{k_1=1}^{T} \sum_{k_2=1}^{T} \left(\sum_{r=1}^{d} a_{i,k_1,r} a_{i,k_2,r}\right) \left(\sum_{s=1}^{p} g_{i,k_1,s} g_{i,k_2,s}\right) \\
&= \mathrm{vec}(a_i a_i^\top)^\top \mathrm{vec}(g_i g_i^\top).
\end{aligned}
$$

[9] Given a noise multiplier σ, ϵ can be computed via first computing the Rényi DP leakage and then converting it to approximate DP. When a privacy budget ϵ is prescribed, we can numerically invert the above procedure to obtain a suitable σ for noisy optimization. Given σ, we also report the ϵ_{GDP} estimated by going through the central limit theorem in Gaussian DP.

Note that when $T = 1$, the identity takes the form of

$$\|\nabla_W \mathcal{L}_i\|_{\mathrm{F}}^2 = \mathrm{vec}(a_i a_i^\top)^\top \mathrm{vec}(g_i g_i^\top) = \|a_i\|_2^2 \, \|g_i\|_2^2 \, .$$

This is exactly the backbone of the trick proposed by Goodfellow [17].

H Default Hyperparameters for Studies in Section 3.1

Table 7: Default hyperparameters for ablation studies.

Method	Full
DP guarantee (ϵ, δ)	$(3, 1/2\lvert\mathcal{D}_{\text{train}}\rvert)$
Clipping norm C	0.1
Batch size B	1024
Learning rate η	10^{-1}
Learning rate decay	no
Epochs T	10 for E2E; 3 for any of MNLI, QQP, QNLI, SST-2
Weight decay λ	0
Noise scale σ	calculated numerically so that a DP budget of (ϵ, δ) is spent after T epochs

I Setup for Memory Profile Experiments in Section C.2

For this experiment, our JAX implementation is adapted from a codebase used for the work by Subramani et al. [48], the chain-rule-based baseline is based on `Opacus==0.14.0`. For a fair comparison, we also optimized the implementation of privacy engine in `Opacus`, since we found certain `einsum` operations to be more memory intensive as needed. All runs were based on full precision (fp32).

We used mock data with the format of the E2E dataset as a testbed for this experiment. We created mock inputs of length 100, as this length is almost the maximum length of examples in the actual E2E dataset.

J Does DP Fine-Tuning Prevent Unintended Memorization?

One of the ultimate goals of fitting models under DP is to ensure that training data extraction is unlikely given the trained model. To empirically evaluate whether DP fine-tuning helps prevent against unintended memorization and such attacks, we follow the secret sharer framework [7] and estimate the *exposure* of artificial canaries inserted into the training set used for fine-tuning. We use the E2E dataset as a testbed.

To create canaries, we first form a subvocabulary by randomly sampling $V = 10$ words in the original vocabulary of GPT2. Our canaries have prefixes of the form

" name : <word> | Type : <word> | area : <word> ",

where <word> is randomly sampled from the subvocabulary. The suffix which our model should learn to predict consists of randomly sampled words with an average length of $l = 5$. By definition, canaries with an estimated exposure close to $\log_2(V^l) \approx 17$ can likely be extracted. We experiment with canary-corrupted datasets for repetition values $r \in \{1, 10, 100\}$. A canary has a higher chance in being extracted when it's repeated for more than once in the training data.

K When and Why Does Linear Scaling Fail?

Recall Tramèr and Boneh [51] suggested that the following simple rule approximately holds in private learning: Scaling the learning rate together with the batch size by the same constant yields models with almost the same performance. Note that their experiments on MNIST, Fashion-MNIST, and CIFAR-10 used only batch sizes in the range of $\{512, 1024, 2048, 4096\}$. These values are fairly

Table 8: Fine-tuning under DP prevents unintended memorization of downstream data. Numbers reported are exposure values estimated with the *approximation by distribution model* approach.

Repetitions / Guarantee	$r = 1$	$r = 10$	$r = 100$
$\epsilon = 3$	1.09 ± 0.86	1.32 ± 1.32	5.26 ± 4.20
non-private	13.82 ± 3.86	$17.22 \pm 0.00.$	17.78 ± 5.49

large from a non-private learning perspective. Indeed, our experiments on E2E suggest that this rule does not generalize to batch sizes that are too small (sampling rates $q = {}^{B}\!/\!_{N} < 2^{-8}$).

We provide an explanation by noting that a core assumption which the linear scaling rule depends on fails to hold for small batch sizes. This assumption is that given a privacy budget, a "square-root" relationship holds between the noise multiplier and the sampling rate (see also [51, Claim D.1]). For instance, Tramèr and Boneh [51] showed that $\sigma \approx c\sqrt{q}$ when $q \in [2^{-7}, 1]$ for some constant c. Our numerical estimates show that this relationship fails to hold for small q – it under estimates the true noise multiplier σ that would be obtained with numerical computation. Figure 6 provides an illustration for $(\epsilon, \delta) = (3, 10^{-5})$ when the sample size $N = 50$k and number of training epochs $E = 50$.

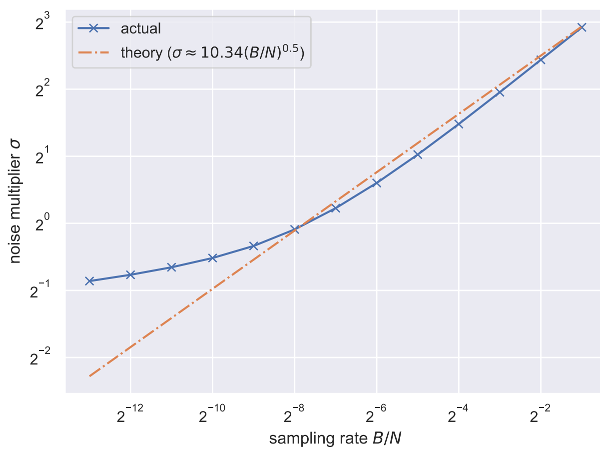


Figure 6: "Square-root" relationship underestimates the noise multiplier for small batch sizes.

L Templates and Label Words for the Text Infilling Formulation in Section 3.2

Recall that fine-tuning for classification can be reformulated as filling in the `[MASK]` token in a template sequence. Here, we list the templates used for each classification task considered in the paper. These templates are almost generic and are not obtained from expensive manual or automated search. We anticipate better templates obtained from automated search based on data [15] to improve the performance even further. However, we also expect that such a procedure would lead to some amount of increased privacy spending if it were based on private data.

Task	Template	Label words
SST-2	$<S_1>$ It was `[MASK]` .	positive: great, negative: terrible
MNLI	$<S_1>$? `[MASK]` , $<S_2>$	entailment: Yes, netural: Maybe, contradiction: No
QNLI	$<S_1>$? `[MASK]` , $<S_2>$	entailment: Yes, not_entailment: No
QQP	$<S_1>$ `[MASK]` , $<S_2>$	equivalent: Yes, not_equivalent: No

Table 9: Templates and label words borrowed from the work by Gao et al. [15].