**Large Language Models Can Be Strong Differentially Private Learners**

Xuechen Li\(^1\), Florian Tramèr\(^2\), Percy Liang\(^1\), Tatsunori Hashimoto\(^1\)
\(^1\)Stanford University \(^2\)Google Research
{l|xuechen,tramer,pliang}@cs.stanford.edu, thashim@stanford.edu

**Abstract**

Differentially Private (DP) learning has seen limited success for building large deep learning models of text, and straightforward attempts at applying Differentially Private Stochastic Gradient Descent (DP-SGD) to NLP tasks have resulted in large performance drops and high computational overhead. We show that this performance drop can be mitigated with (1) the use of large pretrained language models; (2) non-standard hyperparameters that suit DP optimization; and (3) fine-tuning objectives which are aligned with the pretraining procedure. With the above, we obtain NLP models that outperform state-of-the-art DP-trained models under the same privacy budget and strong non-private baselines—by directly fine-tuning pretrained models with DP optimization on moderately-sized corpora. To address the computational challenge of running DP-SGD with large Transformers, we propose a memory saving technique that allows clipping in DP-SGD to run without instantiating per-example gradients for any linear layer in the model. The technique enables privately training Transformers with almost the same memory cost as non-private training at a modest run-time overhead. Contrary to conventional wisdom that DP optimization fails at learning high-dimensional models (due to noise that scales with dimension) empirical results reveal that private learning with pretrained language models tends to not suffer from dimension-dependent performance degradation. Code to reproduce results can be found at https://github.com/lxuechen/private-transformers.

1 Introduction

Machine learning systems trained on sensitive user data can be vulnerable to privacy attacks (Shokri et al., 2017; Hayes et al., 2019). This issue is especially pressing for recent applications of large language models, as these models are capable of memorizing and reconstructing sensitive examples contained in the training data (Zhang et al., 2016; Carlini et al., 2020).

As a result of these concerns, there has been a large interest in developing methods that provide data privacy guarantees for large language models. The standard paradigm for providing such a guarantee in machine learning is *Differential Privacy* (DP) (Dwork et al., 2006; 2014). Unfortunately, DP learning has typically struggled to produce useful models when applied to large language models, resulting in models with either vacuous privacy guarantees (Dupuy et al., 2021) 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) (Song et al., 2013; Bassily et al., 2014; Abadi et al., 2016) injects noise that must scale with the number of parameters, resulting in large noise levels for large language models (Yu et al., 2021b).

We tackle the problem of building performant DP language models for sentence classification and language generation tasks with merely 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, training objective, and pretrained models affect the performance 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 with modest privacy leakage. For text generation, the performance of our models surpasses even strong non-private baselines. For sentence classification, the performance of our fine-tuned models...
surpasses those obtained under heuristic privacy notions (Huang et al., 2020a) which do not possess formal privacy guarantees. Figure 1 illustrates these results. We summarize our contributions below.

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

(2) Running DP-SGD can be memory-intensive due to clipping per-example gradients. We present ghost clipping, a memory saving technique that makes fine-tuning large Transformers under DP memory efficient. Our technique generalizes the Goodfellow (2015) trick to handle sequential inputs, and can be combined with a layer-by-layer clipping procedure (Lee & Kifer, 2020) to enable privately fitting large Transformers with almost the same memory cost as non-private training—at the cost of one additional backward pass per processed batch.

(3) We show that the dimensionality of gradient updates does not explain private fine-tuning performance. While there exist dimension-dependent lower bounds for private (convex) optimization (Bassily et al., 2014), we find that larger pretrained models lead to better private fine-tuning results. Moreover, parameter-efficient adaptation methods that reduce the dimensionality of updates do not necessarily outperform a baseline method that fine-tunes all model parameters.

Our empirical studies indicate that directly fine-tuning pretrained models with DP optimization results in performant DP language models under modest privacy budgets. This enables building practical private NLP models for a range of common tasks where privacy could be at stake.

2 Problem Statement

We build models for sentence classification and language generation tasks with datasets of modest sizes under (central/global) approximate-DP (also known as \((\epsilon, \delta)\)-DP) (Dwork et al., 2014).

**Definition 1** \((\epsilon, \delta)\)-DP. A randomized algorithm \( M : X \rightarrow Y \) is \((\epsilon, \delta)\)-differentially private if for all adjacent datasets \( X, X' \in X \) and all \( Y \subseteq Y \), \( P(M(X) \in Y) \leq e^\epsilon P(M(X') \in Y) + \delta. \)

Two datasets are adjacent if and only if one can be obtained from the other by including an extra record (Mironov et al., 2019).\(^1\) How a record is defined is task dependent and will be made clear below. Intuitively, DP algorithms ensure that random outputs obtained from similar inputs are difficult to distinguish. \( \epsilon \) and \( \delta \) are privacy leakage parameters that measure the loss of privacy and small values imply more privacy. Unlike heuristic privacy notions (Huang et al., 2020b), DP allows for the tracking of privacy loss through the calculation of leakage parameters, and ensures privacy under composition (Dwork et al., 2014), meaning that the overall privacy loss of multiple DP algorithms releasing multiple statistics can be reasoned in a principled manner.

\(^1\) An alternative definition of adjacency assumes all datasets are of equal size and relies on the replacement of records. This is not the definition we adopt.
DP learning typically relies on DP optimizers which privatize gradients before performing updates. The privatization step ensures that the parameter updates leak limited information about the training examples through gradients. Specifically, this step clips per-example gradients with a norm constraint $C$, and adds Gaussian noise $z \sim \mathcal{N}(0, C^2 \sigma^2 I_p)$ to the sum of clipped gradients. Here, $\sigma$ is the noise multiplier determined from the privacy budget $(\epsilon, \delta)$, number of gradient updates $S$, and sampling rate $q = \frac{B}{N}$ for a batch size of $B$ and a dataset with $N$ examples. Intuitively, clipping individual gradients ensures that each example has bounded influence on the parameter update, whereas noising the gradient prevents exact tracing of particular examples. The noise being isotropic implies that larger models would experience heavier noise per update, as the norm of the $p$-dimensional Gaussian $\|z\|_2$ scales as $C\sigma \sqrt{p}$. This is widely believed to be the cause for DP optimization performing poorly at training high-dimensional deep learning models (Kamath, 2020).

Our starting point for building DP language models is (public) pretrained models. Pretrained language models tend to contain general knowledge of language (Manning et al., 2020) and thus should make the downstream private learning problem easier. We fine-tune these models with DP-Adam (Abadi et al., 2016; Kingma & Ba, 2014) (see Appendix A for details) and track privacy loss through Rényi DP (Mironov, 2017), but also report the converted $\epsilon$ from a Gaussian DP CLT (Dong et al., 2019) and from accurately composing tradeoff functions via fast Fourier transform (Gopi et al., 2021). We consider privacy levels $\epsilon \in \{3, 8\}$ and $\delta = \frac{1}{2|D_{\text{train}}|}$ throughout for a training set of size $|D_{\text{train}}|$ (see Appendix B for further details). We tune hyperparameters on a text generation task (E2E; introduced below) and transfer these to remaining tasks. We outline two broad classes of NLP problems considered in this paper and define what constitutes a record below.

**Sentence Classification.** The goal is to learn a model that classifies sentences into one of a few categories. For these tasks, each example/record consists of input sentences and a label to be predicted. We fine-tune models of various sizes in the BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) families, as these models are known to work well for sentence classification.

**Language Generation.** The goal is to learn a model that generates natural language sentences given some context. For table-to-text generation tasks such as E2E (Novikova et al., 2017) and DART (Nan et al., 2020), each example/record in the training data consists of a pair of table entry and corresponding text description to be predicted. For a dialogue generation task such as PersonaChat (Zhang et al., 2018), each example/record consists of metadata, a dialogue history, and a response to be predicted. We fine-tune GPT-2 (Radford et al., 2019) of various sizes for these problems, as this model family is known to work well for text generation.

### 3 Effective Differentially Private Fine-Tuning

By studying the impact of hyperparameters and choice of fine-tuning objective, we demonstrate that the performance of the DP-Adam baseline can be substantially improved, even matching some strong non-private learning results. Our analyses reveal common failure modes when straightforwardly applying DP optimization and explain poor results reported in past works that consider these baselines.

#### 3.1 Hyperparameter Tuning

DP optimization is sensitive to the choice of hyperparameters (Papernot et al., 2019). Our experiments suggest that performance can vary from being close to trivial with ill-chosen hyperparameters to near previous state-of-the-art with appropriately chosen ones. As a consequence, we present simple but effective guidelines on setting the most important hyperparameters. Unless otherwise stated, the unmentioned hyperparameters are set to defaults documented in Appendix H.

##### 3.1.1 Batch Size, Learning Rate & Training Epochs

Our experiments suggest that batch size is one of the most important hyperparameters to set correctly, and the dependence of the optimal batch size on learning rate and training epochs makes its selection complex. We first describe batch size selection in realistic, compute-bound settings and then describe how the complexity of identifying the optimal batch size in these situations arise due to constraints on the number of training epochs.
**Fixed Training Epochs** \( E \). We first describe a realistic situation in which there is a constraint on the total amount of compute budget available. For the case of DP-SGD, this compute budget constraint often corresponds to a constraint on the number of examples that are processed by SGD.\(^2\) In this fixed training epoch setting, the learning rate and batch size jointly affect performance, since using larger batches implies performing fewer gradient updates. To study this joint influence empirically, 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 \( \sim65 \) without privacy here (Wiseman et al., 2018).

Recall that 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). This implies that naively fine-tuning pretrained language models privately using the non-private setup would result in more performance degradation than necessary.

Recently, Tramèr & Boneh (2020) studied how the batch size and learning rate jointly affect learning private image classifiers 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 further explain why linear scaling fails for small batches in Appendix D.

**Fixed Update Steps** \( S \). In the fixed epoch setting, we saw that the optimal batch size setting was complex due to the trade-off between batch size and number of gradient updates. We now show that the complexity of setting batch sizes arises almost entirely from this tradeoff by considering a different setting, where the total number of gradient updates (rather than epochs) is fixed. In this case, using larger batches implies training for more epochs, and we find that using larger batch sizes almost always results in better performance at a given privacy budget (at the cost of processing more examples and using more compute), once other hyperparameters are fixed. We provide an explanation of this by introducing the idea of an *effective noise multiplier* \( \sigma_{\text{eff}} = \frac{\sigma}{q} = \frac{\sigma N}{B} \). Recall the noise multiplier \( \sigma \) is determined from the privacy budget \( (\epsilon, \delta) \), update steps \( S \), and sampling rate \( q \). In addition, recall the form of the privatized gradient \( \tilde{g} \) in DP-SGD/DP-Adam:

\[
\tilde{g} = g + \xi, \quad g = \frac{1}{B} \sum_{i=1}^{B} \text{Clip}(\nabla L_i, C), \quad \xi \sim \mathcal{N}(0, C^2 \frac{\sigma^2}{q^2} I_p) = \mathcal{N}(0, C^2 \frac{\sigma N^2}{B^2} I_p),
\]

where \( \nabla L_i \) is the gradient of the \( i \)th example in a batch of \( B \) examples and \( \text{Clip}(v, K) \) clips the vector \( v \) by the norm constraint \( K \). We observe that for moderately large batches, the signal-to-noise ratio \( r = \|\nabla L\|_2 / \|\xi\|_2 \) is mainly controlled by the batch size through the effective noise multiplier: The signal term \( \tilde{g} \) tends to concentrate quickly due to being an average of bounded vectors, whereas the effective noise multiplier \( \sigma_{\text{eff}} \) could vary significantly as the batch size \( B \) changes. Figure 3 (a) shows that the effective noise multiplier decreases as batch size increases. In addition, Figure 3 (b) plots the average signal-to-noise ratio \( r \) over the first 30 gradient updates against the final model’s performance on E2E and demonstrates that large batches (up to a threshold) lead to both increased signal-to-noise ratio at the beginning of training and better performance at the end of training. These findings additionally resonate with and explains recent empirical successes of private pretraining (Anil et al., 2021).

### 3.1.2 Clipping Norm

DP optimization is known to be sensitive to the choice of clipping norm. Since the scale of noise depends on this clipping norm (recall its standard deviation is \( C\sigma \)), picking the threshold \( C \) much larger than the actual gradient norm implies more noise is being applied than necessary. In practice, we have found that a small clipping norm which enforces almost all gradients to be clipped throughout training leads to the best performing models (see Figure 8 in Appendix H).

### 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. This alignment simplified

---

\(^2\)This is because DP-SGD often necessitates microbatching, in which case the number of backward passes is independent of the actual batch size for gradient updates but dependent on numbers of passes through the data.
the task and benefited private learning. While pretrained models are naturally aligned for language generation, it is much less so for classification tasks. The standard approach for adapting language models for classification involves stacking a freshly initialized network on top of the encoding of the special [CLS] token and jointly optimizing all parameters (Devlin et al., 2018). This workflow introduces a discrepancy between pretraining and fine-tuning: Pretraining predicts masked out words from a large vocabulary whereas fine-tuning predicts integer labels.

To eliminate the discrepancy, we instead consider learning to predict the missing word during fine-tuning for classification. 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 (Petroni et al., 2019), few-shot learning (Gao et al., 2020) and multi-task fine-tuning (Wei et al., 2021). On SST-2, we found that using the generic template as described above already improved private fine-tuning performance by 3 ∼ 5% across different settings. Table 1 (to be presented) contains additional results and Appendix E includes analyses on choices of label words.

4 GHOST CLIPPING: CLIIPPING WITHOUT PER-EXAMPLE GRADIENTS

DP-SGD has high memory overhead due to clipping per-example gradients. Naïvely implemented, this step instantiates a giant gradient vector for each example during optimization and can be prohibitively expensive. For example, Hoory et al. (2021) pretrained BERT with DP optimization and reported memory issues when using the large batches necessary to achieve high performance. A 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 (Tramer & Boneh, 2020). This solution, however, is unlikely to be sufficient as neural language models become larger and fitting a few copies of the gradient in memory can be difficult. Lee & Kifer (2020) observed that per-example gradients need not be instantiated at all, if the goal is to sum the clipped gradients. They presented a clipping procedure that only instantiates the per-example gradient for parameters of a single layer in the model one at a time, as opposed to the entire model at once, at the cost of an extra backpropagation pass per processed batch.

Unfortunately, we find this trick to be still insufficient for sequence models such as Transformers (Vaswani et al., 2017), as the memory requirement for per-example gradients of embedding layers and language modeling heads can be costly. We extend the Lee & Kifer (2020) approach such that training Transformers with DP optimization can have almost the same memory consumption as non-private training. Unlike their approach, our extension avoids instantiating the per-example gradient even for individual linear layers. We call this approach ghost clipping, as the per-example gradient is the ghost that never explicitly appears. We anticipate this extension to be useful for both privately fine-tuning and pretraining large Transformers.

4.1 THE MEMORY TRICK BY LEE & KIFER (2020)

Per-example gradient clipping is easy if we know per-example gradient norms. In this case, we first compute the scaling factor $c_i = \min(1, C/\|\nabla L_i\|_2)$, where $C$ is the clipping threshold and $L_i$ is the loss associated with the $i$th example. Then, we perform the usual backward pass with the reweighted
scalar loss $\sum_i c_i L_i$. This procedure gives us the sum of clipped gradients. Under this setup, the difficulty is computing the per-example gradient norm $\|\nabla L_i\|_2$. We emphasize two technicalities that enable computing this quantity without instantiating the full per-example gradient $\nabla L_i$.

First, for a typical 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 $d^{(l)}$ and the gradient of the loss w.r.t. the output $g^{(l)}$, both of which are available during backpropagation. 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. $\|u\|_2 = \|\|u_1\|_2, \ldots, \|u_k\|_2\|_2$. The second observation means that computing the per-example gradient norm $\|\nabla L_i\|_2$ can be done by computing the per-example gradient norms for individual layers of the neural net $\|\nabla W^{(l)} \nabla L_i\|_2, \ldots, \|\nabla W^{(T)} \nabla L_i\|_2$ one at a time ($L$ is layer count). Moreover, the first observation implies that the norms for each layer can be computed using quantities freely available to a typical backward pass. Overall, the per-example gradient norm of any network without parameter sharing can be computed in a layer-by-layer fashion with only one per-example gradient tensor for a single layer being instantiated at any time.

4.2 Ghost clipping for Transformers with sequential data

The trick by Lee & Kifer (2020) still requires instantiating the per-example gradient of individual layers (although not simultaneously). This can be problematic in terms of memory for Transformers with large embedding layers.\(^3\) Here, we present a specialized procedure for computing the per-example gradient norm for linear and embedding layers when they are applied to sequential data.\(^4\) This procedure reduces memory footprint and can be viewed as a generalization of the Goodfellow (2015) trick 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 $s$. Here, $T$ is the number of time steps in the input, and we omitted biases for simplicity. Simple calculation shows that the per-example gradient is the product of two matrices:

$$\nabla_W L_i = g_i^T a_i \in \mathbb{R}^{p \times d}. \quad (1)$$

Since the per-example gradient norms are the end goal, the per-example gradients $\{\nabla_W L_i\}_{i=1}^B$ themselves need not be instantiated explicitly. More precisely, we observe that the squared per-example gradient norm for this layer $\|\nabla_W L_i\|_2^2$ obeys the following identity:

$$\|\nabla_W L_i\|_2^2 = \text{vec}(a_i a_i^T) \text{vec}(g_i g_i^T). \quad (2)$$

See Appendix F for a derivation. Implemented with common primitives in machine learning libraries, (2) has a memory complexity of order $O(B T^2)$ when $a_i a_i^T, g_i g_i^T \in \mathbb{R}^{T \times T}$ are instantiated,\(^5\) as opposed to $O(Bpd)$ in the naïve approach which goes through instantiating (1).\(^6\)

The memory efficiency of this procedure is exemplified with off the shelf pretrained language models, most of which have large embeddings. For instance, for GPT-2, $d = 50,000$ and $p = 768$ for the embedding layer, and the context window $T \leq 1024$.\(^8\) Our method in theory reduces the memory cost of this large embedding layer by at least a factor of 22. In practice, we also observe significant savings, since embedding layers can be a major source of memory spending for training large language models.\(^9\) To stress-test ghost clipping, we compare it with 4 baselines: The PyTorch package Opacus that implements DP optimization by instantiating per-example gradients, the approach by Lee & Kifer (2020), non-private training in PyTorch, and naïve DP optimization implemented in PyTorch.

---

\(^3\)For GPT-2, per-example gradients w.r.t. the embedding for ten examples alone occupy ~1.5GB of memory.

\(^4\)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.

\(^5\)This is assuming the space complexity for multiplying two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ is roughly $O(mp)$, which is the case for most workloads running on a framework like PyTorch.

\(^6\)More sophisticated solutions may even avoid instantiating $a_i a_i^T$ and $g_i g_i^T$ entirely by trading in more run-time. Custom CUDA kernels are likely needed to make these solutions fast in practice.

\(^7\)We omitted the cost of storing $a_i$ and $g_i$, since our goal is to compare the additional cost induced by computing gradient norms.

\(^8\)In practice, for fine-tuning tasks, the maximum sequence length is usually a few hundred.

\(^9\)While there are alternative approaches for reducing the memory footprint of embedding layers during training, these methods tend to introduce extra hyperparameters that require tuning and privacy spending.
JAX with jit and vmap enabled. We include the JAX baseline since recent studies show that DP optimization can be made cheap through compiler optimization (Subramani et al., 2020). Figure 4 (a) shows that for typical inputs, our technique is the most memory friendly and allows fitting batches almost as large as those in non-private training. Since ghost clipping allows us to fit larger batches but with a run-time penalty, a natural question is whether it improves throughput with the use of larger batches. Figure 4 (b) shows that while ghost clipping only provides minor gains compared to Opacus for smaller models, it allows processing ~10% more examples compared to the approach by Lee & Kifer (2020) for fitting GPT-2-large, a model that neither Opacus or JAX could handle. See Appendix G for the setup of these experiments.

Figure 4: Left: Ghost clipping is 3 times more memory efficient than Opacus and is almost as efficient as non-private training for typical sequences across model sizes. For GPT-2-large, we were unable to fit single-example micro batches together with gradient accumulation with Opacus or JAX on a TITAN RTX GPU (24 GBs of VRAM). Right: DP optimization with ghost clipping processes ~10% more examples than the approach by Lee & Kifer (2020) under unit time for GPT-2-large.

5 LOW DIMENSIONAL UPDATES ARE NOT NECESSARILY BETTER

Since the norm of the noise added to gradients in DP scales with dimensionality, it is natural to ask whether privatizing and updating fewer parameters would result in improved performance. We decompose this question into two aspects: (1) Do smaller pretrained models lead to better private fine-tuned performance, and (2) do parameter-efficient adaptation methods designed with a reduced dimensionality of updates outperform full fine-tuning? Our experiments below show that the answer to both questions is negative. Reported numbers in this section are averaged over three seeds.

5.1 LARGER PRETRAINED MODELS RESULT IN BETTER PERFORMANCE

We observe that larger pretrained models lead to better private fine-tuned performance. Specifically, we fully fine-tune four sizes of GPT-2 models (for language generation) and three sizes of BERT/RoBERTa models (for sentence classification) at the same privacy budget with DP-Adam and compare their performances. Since the performance of DP optimization heavily depends on hyperparameter choices, we need to ensure that our hyperparameters are not particularly favoring larger models. We thus tune hyperparameters on the smallest model and then reuse the same hyperparameters for all fine-tuning workloads. Figure 1 from earlier demonstrates gains on E2E and MNLI from model scaling, and we find similar improvements on 5 additional tasks (deferred to Figure 5 in Appendix C).

5.2 FULL FINE-TUNING WITH DP-ADAM MATCHES STATE-OF-THE-ART

There is a range of lightweight fine-tuning methods that reduce the dimensionality of updates, including some that are designed for DP (Yu et al., 2021c). Do methods that optimize fewer parameters lead to better results under DP even if they perform similarly non-privately? Empirical results suggest otherwise and that full fine-tuning is a strong baseline that even matches specialized low-dimensional DP learning methods for both classification and generation. Below, we study the two sets of tasks separately. For completeness, all experimental details are in Appendix K.

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. (2021c). The associated datasets have modest sizes: SST-2 and QNLI have 60k+ and 100k+ training examples, respectively. MNLI and QQP each contains less than 400k examples.
Table 1 shows that using larger pretrained models and the text-infilling objective generally improve classification accuracy. We compare full fine-tuning with reparameterized gradient perturbation (RGP) (Yu et al., 2021c), 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 to reduce DP noise in high-dimensional models. We note that full fine-tuning with the text infilling objective outperforms well-tuned RGP on all tasks despite being the simplest baseline. Computationally, while RGP is faster per-update, it requires more than 3 times as many epochs as full fine-tuning – overall, both methods comparable in terms of wall time.

Table 1: Full fine-tuning larger pretrained models with text infilling has best performance. Results are dev set accuracies. Best numbers based on two-sample test for each privacy level are in bold.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\epsilon = 3$</th>
<th>$\epsilon = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNLI-(m/mm)</td>
<td>QQP</td>
</tr>
<tr>
<td>RGP (RoBERTa-base)</td>
<td>82.47/82.10</td>
<td>85.41</td>
</tr>
<tr>
<td>RGP (RoBERTa-large)</td>
<td>85.53/85.81</td>
<td>86.65</td>
</tr>
<tr>
<td>full (RoBERTa-base)</td>
<td>82.45/82.99</td>
<td>85.56</td>
</tr>
<tr>
<td>full + infilling (RoBERTa-base)</td>
<td>86.43/86.46</td>
<td>86.43</td>
</tr>
</tbody>
</table>

| $\epsilon \approx$ (Gaussian DP + CLT) | 2.52 | 2.52 | 2.00 | 1.73 | 5.83 | 5.85 | 4.75 | 4.33 |
| $\epsilon \approx$ (Compose tradeoff func.) | 2.75 | 2.75 | 2.57 | 2.41 | 7.15 | 7.16 | 6.87 | 6.69 |

Table-To-Text Generation. We study different fine-tuning methods under DP for table-to-text generation where the goal is to generate natural language descriptions of table entries. We consider the datasets E2E (Novikova et al., 2017) and DART (Nan et al., 2020). E2E consists of simple restaurant reviews, whereas DART consists of open-domain table entries from Wikipedia and is more complex. Both datasets are small: E2E has more than 40k training examples, whereas DART has more than 60k. Since we are the first to experiment with this task under DP, we compare full fine-tuning (full) against a suite of parameter-efficient approaches which includes LoRA (Hu et al., 2021), prefix-tuning (Li & Liang, 2021) (prefix), RGP, and fine-tuning the top 2 Transformer blocks (top2), all of which optimize few parameters. On GPT-2 (125 million parameters), prefix-tuning with default hyperparameters optimizes $\approx 10$ million parameters; LoRA with rank 4 optimizes $\approx 0.15$ million parameters. We also report results for training from randomly initialized weights (retrain). Hyperparameters of each method were tuned only the E2E dataset; the complete search ranges are in Appendix I. Table 2 shows that LoRA and full fine-tuning are generally the most performant on E2E. Tables 7 and 8 in Appendix J contain the full result on E2E and DART and confirm the trend.

Table 2: Full fine-tuning performs on par with or outperforms others methods that execute gradient update in low dimensional spaces. Results are on E2E from fine-tuning GPT-2.

<table>
<thead>
<tr>
<th>Metric</th>
<th>$\epsilon = 3$</th>
<th>$\epsilon = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gaussian DP + CLT</td>
<td>Compose tradeoff func.</td>
</tr>
<tr>
<td>BLEU</td>
<td>$\epsilon \approx 2.68$</td>
<td>$\epsilon \approx 2.75$</td>
</tr>
<tr>
<td></td>
<td>$\epsilon \approx 6.77$</td>
<td>$\epsilon \approx 7.27$</td>
</tr>
<tr>
<td>ROUGE-L</td>
<td>$\epsilon \approx 2.68$</td>
<td>$\epsilon \approx 2.75$</td>
</tr>
<tr>
<td></td>
<td>$\epsilon \approx 6.77$</td>
<td>$\epsilon \approx 7.27$</td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
</tr>
</tbody>
</table>

Chit-Chat Dialog Generation. We stress-test full fine-tuning under DP with the task of chit-chat dialog generation. This task has the distinct challenge that the response space is intrinsically diverse (Li et al., 2015; Gao et al., 2018) since human conversations can be informal and noisy (Zhang et al., 2019). Moreover, dialog datasets are usually formed with user data which may contain sensitive information. We use the Persona-Chat dataset (Zhang et al., 2018) as a testbed and build off a processed version that has $\approx 130k$ training entries. Each entry contains a dialog history, persona
descriptions of the respondent, and the response. We fine-tune GPT-2, GPT2-medium, and DialoGPT-medium on this dataset both privately and non-privately by training to predict the response with the dialog history and persona description. We report the F1 score and perplexity on the validation split, and human evaluated quality scores of generations. Table 3 shows that private models have strong performance. In particular, fine-tuned DialoGPT-medium at $\epsilon = 8$ beats the (non-private) winning entry of the ConvAI2 challenge (Dinan et al., 2019) on perplexity and has a human evaluation rating that is close to non-private models. Samples from our private models can be found in Appendix O.

Table 3: Fine-tuning with DP-Adam yields high quality chit-chat dialog generation models.

<table>
<thead>
<tr>
<th>Model</th>
<th>DP Guarantee</th>
<th>Gaussian DP +CLT</th>
<th>Compose tradeoff func.</th>
<th>Metrics</th>
<th>F1 ↑</th>
<th>Perplexity ↓</th>
<th>Quality (human) ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-2</td>
<td>$\epsilon = 3$</td>
<td>$\epsilon \approx 2.54$</td>
<td>$\epsilon \approx 2.73$</td>
<td>15.90</td>
<td>24.59</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$</td>
<td>$\epsilon \approx 6.00$</td>
<td>$\epsilon \approx 7.13$</td>
<td>16.08</td>
<td>23.57</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>-</td>
<td>17.96</td>
<td>18.52</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GPT-2-medium</td>
<td>$\epsilon = 3$</td>
<td>$\epsilon \approx 2.54$</td>
<td>$\epsilon \approx 2.73$</td>
<td>15.99</td>
<td>20.68</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$</td>
<td>$\epsilon \approx 6.00$</td>
<td>$\epsilon \approx 7.13$</td>
<td>16.53</td>
<td>19.25</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>-</td>
<td>18.64</td>
<td>15.40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DialoGPT-medium</td>
<td>$\epsilon = 3$</td>
<td>$\epsilon \approx 2.54$</td>
<td>$\epsilon \approx 2.73$</td>
<td>17.37</td>
<td>17.64</td>
<td>2.82 (2.56, 3.09)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$</td>
<td>$\epsilon \approx 6.00$</td>
<td>$\epsilon \approx 7.13$</td>
<td>17.56</td>
<td>16.79</td>
<td>3.09 (2.83, 3.35)</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>-</td>
<td>19.28</td>
<td>14.28</td>
<td>3.26 (3.00, 3.51)</td>
<td>-</td>
</tr>
<tr>
<td>HuggingFace (ConvAI2 winner)</td>
<td>non-private</td>
<td>-</td>
<td>-</td>
<td>19.09</td>
<td>17.51</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HuggingFace (our implementation)</td>
<td>non-private</td>
<td>-</td>
<td>-</td>
<td>16.36</td>
<td>20.55</td>
<td>3.23 (2.98, 3.49)</td>
<td>-</td>
</tr>
<tr>
<td>Reference</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3.74 (3.49, 4.00)</td>
<td>-</td>
</tr>
</tbody>
</table>

6 Related Work

Private NLP. The privacy-preserving NLP space is largely divided by whether or not a formal notion of privacy is considered. McMahan et al. (2017) successfully trained small word-level RNNs with 1.35 million parameters in a federated setting with more than 700k users under a global DP guarantee of $(\epsilon, \delta) = (4.6, 10^{-9})$. Ramaswamy et al. (2020) train production grade next-word prediction models using DP-FedAvg with millions of users. Qu et al. (2021) studied fine-tuning BERT for language understanding tasks under local DP. Kerrigan et al. (2020) presented initial results that public pretraining is helpful for downstream DP fine-tuning. However, they did not attempt fine-tuning large pretrained models with DP-SGD. Bommasani et al. (2021) briefly commented on the possibility of achieving cheaper private learning by fine-tuning large pretrained language models. Anil et al. (2021) pretrained BERT under global DP on datasets with hundreds of millions of examples. Dupuy et al. (2021) studied private BERT fine-tuning on datasets of utterances, but reported results with $\epsilon$ on the order of at least 100. Orthogonally, many works considered training language models that satisfy empirical notions of privacy (Xu et al., 2021; Coavoux et al., 2018; Mireshghallah et al., 2021; Melamud & Shivade, 2019). Our work is distinct from all works mentioned above in that we study fine-tuning large language models (with hundreds of millions of parameters) under global DP with stringent guarantees ($\epsilon \in \{3, 8\}$) on smaller datasets (much less than a million examples).

7 Scope and Limitations

We presented strategies for fine-tuning large pretrained language models under DP for a wide range of NLP tasks. For researchers and practitioners working on private NLP, our empirical results suggest that DP fine-tuning with a proper setup is a competitive baseline that is worth trying before prematurely shifting to less formal notions of privacy which have not stood against the test of time. Below we list some limitations and future directions.

Pretraining vs Private Learning. Our model scaling results suggest that using larger pretrained models improves performance. This argument, however, is dependent on the particular choice of pretrained models. How pretraining helps private learning and whether better pretrained models for private learning could be built are interesting future avenues.

Scaling Laws for Private Learning. While scaling laws (Kaplan et al., 2020) for non-private deep learning have become prevalent, we are unaware of a case study in the private realm. Studies on how the dimensionality of models (and pretraining) generally affect private deep learning in precise quantitative terms will likely be a useful tool in trading off compute budget and model quality.
ETHICS STATEMENT

While the present paper studies private NLP, all experiments performed herein are based on publicly available datasets for reproducibility purposes. The paper empirically studies various behaviors of (mostly existing) algorithms, thus likely does not introduce any new ethical or societal concerns. The present paper does not introduce any new dataset.

REPRODUCIBILITY

All experiments in the paper are based on publicly available datasets. Links to these datasets are included in the main text and appendices. Hyperparameters necessary for reproducing our experiments are documented in Appendix I and H. Code for the ghost clipping technique can be found at https://github.com/lxuechen/private-transformers.

ACKNOWLEDGMENTS

We thank Xiang Lisa Li for help on reproducing non-private prefix-tuning results. We thank Da Yu for discussions and help on reproducing results for the RGP method. We thank Abhradeep Guha Thakurta and Rishi Bommasani for helpful discussions. We thank members of the Stanford statistical machine learning group for comments on early versions of the abstract. We thank Guodong Zhang and Mayee Chen for comments on an early draft. We thank Stanford HAI for a Google Cloud Credits Grant. XL is supported by a Stanford Graduate Fellowship.

REFERENCES


A DP-ADAM

We use DP-Adam throughout. DP-Adam works just like regular Adam (Kingma & Ba, 2014) but performs updates and moment accumulation with privatized gradients. The gradient privatization part is the same as that performed in DP-SGD (Song et al., 2013; Abadi et al., 2016). Seemingly uncommon, DP-Adam is used in many private ML libraries. To determine the noise multiplier, we account privacy through Rényi differential privacy (RDP) (Mironov, 2017; Mironov et al., 2019). For completeness, we include the pseudocode for DP-Adam below.

Algorithm 1 DP-Adam

1: Input: Data D = \{x_i\}_{i=1}^{N}, learning rate \eta, noise multiplier \sigma, batch size B, Euclidean norm threshold for gradients \(C\), epochs \(E\), 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 \(\gamma \in \mathbb{R}\).
2: for \(t \in [E \cdot N/B] \) do
3: \hspace{1em} Draw a batch \(B_t\) via Poisson sampling; each element has probability \(B/N\) of being selected
4: \hspace{1em} for \(x_i \in B_t\) do
5: \hspace{2em} \(g_i(x_i) \leftarrow \nabla \theta_i \mathcal{L}(x_i), \hat{g}_t(x_i) \leftarrow g_t(x_i) \cdot \min(1, C/\|g_t(x_i)\|_2)\)
6: \hspace{1em} end for
7: \(z_t \sim N(0, \sigma^2 C^2 I_p)\)
8: \(\hat{g}_t = 1/B \left(\sum_{i=1}^{N} \hat{g}_i(x_i) + z_t\right)\)
9: \(\theta_{t+1}, m_{t+1}, v_{t+1} \leftarrow \text{AdamUpdate}(\theta_t, m_t, v_t, \hat{g}_t, \beta_1, \beta_2, \gamma)\)
10: end for
11: return \(\theta_{T \cdot N/B}\)

Algorithm 2 AdamUpdate

1: Input: \(\theta_t, m_t, v_t, \hat{g}_t, \beta_1, \beta_2, \gamma\)
2: \(m_{t+1} \leftarrow \beta_1 \cdot m_t + (1 - \beta_1) \cdot g_t, v_{t+1} \leftarrow \beta_2 \cdot v_t + (1 - \beta_2) \cdot g_t^2\)
3: \(\hat{m}_{t+1} \leftarrow m_{t+1}/(1 - \beta_1^t), \hat{v}_{t+1} \leftarrow v_{t+1}/(1 - \beta_2^t)\)
4: \(\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \hat{m}_{t+1}/\sqrt{\hat{v}_{t+1} + \gamma}\)
5: return \(\theta_{t+1}, m_{t+1}, v_{t+1}\)

B PRIVACY ACCOUNTING

We train all models under approximate-DP (Dwork et al., 2014), and we view two datasets as being adjacent if and only if one can be obtained from the other by including an extra record (Mironov et al., 2019). Instead of accounting the privacy loss with Moments Accountant (Abadi et al., 2016), we perform computation through (i) Rényi DP (Mironov, 2017; Mironov et al., 2019), (ii) Gaussian DP with an associated central limit theorem (Dong et al., 2019), and (iii) numerically composing tradeoff functions via fast Fourier transform (Gopi et al., 2021; Koskela et al., 2020). All approaches are improvements over the Moments Accountant. Accounting loss with Rényi DP provides strict upper bounds on the actual privacy leakage but may result in loose bounds. Accounting loss with Gaussian DP and its central limit theorem, although asymptotically exact, only provides approximations to the actual loss under a finite number of compositions (Dong et al., 2019, Theorem 3.4). Gopi et al. (2021) observed that accounting loss with GDP and its CLT results in underestimation and proposed to numerically compose tradeoff functions resulting in both upper and lower bounds on the actual leakage \(\epsilon\). We therefore also report the converted \(\epsilon\) with the approach by Gopi et al. (2021) using their code.\(^{11}\)

Given the noise multiplier \(\sigma\), sampling rate \(q\), number of steps \(S\), and failure constant \(\delta, \epsilon\) can be computed via first computing the Rényi DP leakage and then converting it to approximate DP. When a privacy spending (specified by a set of given \(\epsilon\) and \(\delta\)) is prescribed, we can numerically invert the above procedure to obtain a suitable \(\sigma\) for noisy optimization. This is what we do throughout all

\(^{11}\)https://github.com/tensorflow/privacy/blob/7c4f5bab0964bd32b7ceafa009d9488920856440/tensorflow_privacy/privacy/optimizers/dp_optimizer.py#L385

\(^{11}\)https://github.com/microsoft/prv_accountant
experiments. For completeness, given the $\sigma$ chosen as above, we also report the leakage estimated by going through the central limit theorem in Gaussian DP (Bu et al., 2020).

Model selection from hyperparameter tuning on private training and validation data incurs extra leakage (Liu & Talwar, 2019; Chaudhuri & Vinterbo, 2013; Papernot & Steinke, 2021). We perform tuning only on the E2E task and reuse almost the exact hyperparameters for remaining tasks. Note the general strategy of tuning hyperparameters on a separate (public) dataset and thereafter transfer has been applied to train private production language models (Ramaswamy et al., 2020).

C ADDITIONAL RESULTS ON MODEL SCALING

We repeat the model scaling experiments from Figure 1 on the other tasks considered in the paper. Figure 5 shows that the trend that larger and better pretrained models lead to improved private fine-tuned performance holds consistently across all tasks. TextHide numbers based on the TextHide\textsubscript{intra} formulation (Huang et al., 2020a).

![Graphs showing model scaling results across various tasks.](image)

Figure 5: Larger and better pretrained models consistently lead to better private fine-tuned performance on sentence classification and language generation tasks.
D \hspace{1em} WHEN AND WHY DOES LINEAR SCALING FAIL?

Recall Tramèr & Boneh (2020) 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 \( \{512, 1024, 2048, 4096\} \). These values are fairly 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 (Tramèr & Boneh, 2020, Claim D.1)). For instance, Tramèr & Boneh (2020) 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 underestimates the true noise multiplier \( \sigma \) that would be obtained with numerical computation. Figure 6 provides an illustration for \((\epsilon, \delta) = (3, 10^{-5})\) when the sample size \( N = 50k \) and number of training epochs \( E = 50 \).

![Figure 6: “Square-root” relationship underestimates the noise multiplier for small batch sizes.](image)

E \hspace{1em} HOW DOES THE CHOICE OF LABEL WORDS AFFECT SENTENCE CLASSIFICATION

Recall in Section 3.2 we cast sentence classification as filling in the missing word among \( K \) candidates for a \( K \)-way classification problem. Since a label word could be mapped to multiple possible words, we study how the choice of label words affect performance. We again use the sentiment classification task SST-2 as a testbed. Figure 7 shows the effect of varying the label word, where we measure the alignment between the label word and the downstream task by the zero-shot performance of the infilling task (x-axis). We find that increasing the task alignment of the label words alone improves performance by \( 1 - 2\% \) (orange curve), which is in contrast to the non-private setting, where choice of label words do not affect performance in statistically significant ways (blue curve).

![Figure 7: Better text labels help private learning more than non-private learning.](image)
F 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:

$$\|\nabla_W \mathcal{L}_i\|_F^2 = \|g_i^T a_i\|_F^2 = \left\| \sum_{k=1}^{T} g_{i,k} a_{i,k}^T \right\|_F^2$$

$$= \sum_{r=1}^{d} \sum_{s=1}^{p} \left( \sum_{k=1}^{T} a_{i,k,r} g_{i,k,s} \right)^2$$

$$= \sum_{r=1}^{d} \sum_{s=1}^{p} \sum_{k_{1}=1}^{T} \sum_{k_{2}=1}^{T} a_{i,k_{1},r} g_{i,k_{1},s} a_{i,k_{2},r} g_{i,k_{2},s}$$

$$= \left( \sum_{r=1}^{d} \sum_{s=1}^{p} g_{i,k_{1},s} g_{i,k_{2},s} \right) \left( \sum_{r=1}^{d} \sum_{s=1}^{p} a_{i,k_{1},r} a_{i,k_{2},r} \right)$$

$$= \text{vec}(a_i^T a_i)^T \text{vec}(g_i^T g_i).$$

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

$$\|\nabla_W \mathcal{L}_i\|_F^2 = \text{vec}(a_i^T a_i)^T \text{vec}(g_i^T g_i) = \|a_i\|_2^2 \|g_i\|_2^2.$$

This is exactly what is used in the Goodfellow (2015) trick.

G Protocol for Experiments in Section 4.2

For these experiments, we used mock examples with the same format as examples in the E2E dataset. We created mock input sequences of length 100, as this length is almost the maximum length of examples in the actual E2E training set.

Our JAX implementation is adapted from a codebase used in the work by Subramani et al. (2020) and utilizes the package flaxmodels for loading pretrained models. The Opacus (Yousefpour et al., 2021) baseline is based on version 0.14.0 of the library, where for a fair comparison, we also optimized the implementation of the privacy engine by replacing all einsum operations with basic primitives that manipulate tensors directly. We found certain einsum steps to cause large and unnecessary speed and memory overheads.

To identify the maximum batch size that each approach could use, we ran binary search over the range of possible batch sizes until the upper bound matched the lower bound and that OOM did not occur. To estimate the throughput of each private method, we compared them to non-private training first. Specifically, for a pairwise comparison, we took the maximum batch size for non-private training and some private method, and computed the least common multiple as a batch size to perform updates with. Using the least common multiple batch size ensures that both methods will process exactly the same number of examples and perform the same number of updates. By estimating the time elapse of these methods for performing a fixed number of updates, we obtained throughputs for non-private training and the private method. This gives us the relative throughput of the private method with non-private training as the reference.

For methods implemented in PyTorch, the time elapse was recorded with torch.profiler. When estimating the time elapse for a training procedure, we first performed 3 gradients updates as a warm up process before taking the actual steps which will be timed. In particular, this eliminates the time that JAX uses for compiling computation graphs with vmap and jit.

All experimental runs in the section were under full precision.

12https://github.com/matthias-wright/flaxmodels
Details and additional results for Studies in Section 3.1

Table 4: Default hyperparameters for ablation studies.

<table>
<thead>
<tr>
<th>Method</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP guarantee ( (\epsilon, \delta) )</td>
<td>((3, 1/2</td>
</tr>
<tr>
<td>Clipping norm ( C )</td>
<td>0.1</td>
</tr>
<tr>
<td>Batch size ( B )</td>
<td>1024</td>
</tr>
<tr>
<td>Learning rate ( \eta )</td>
<td>(10^{-3})</td>
</tr>
<tr>
<td>Learning rate decay</td>
<td>no</td>
</tr>
<tr>
<td>Epochs ( E )</td>
<td>10 for E2E; 3 for SST-2</td>
</tr>
<tr>
<td>Weight decay ( \lambda )</td>
<td>0</td>
</tr>
<tr>
<td>Noise scale ( \sigma )</td>
<td>calculated numerically so that a DP budget of ( (\epsilon, \delta) ) is spent after ( E ) epochs</td>
</tr>
</tbody>
</table>

Figure 8: Additional results on hyperparameter sensitivity.

Hyperparameter Search Ranges for experiments in Section 5

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 training epochs and batch size. For SST-2, we reused the same batch size as for private E2E fine-tuning, and the number of epochs exactly as in typical non-private fine-tuning for SST-2 (number of epochs equals to 3 in this case). For remaining classification tasks, we use a batch size such that the sampling rate is the same as for SST-2, and a number of training epochs that is roughly proportional to the dataset size. Appendix L outlines why we transfer the sampling rate as opposed to the batch size. We list the range of hyperparameters that we searched over for each individual adaptation method on E2E 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 & Liang (2021) (10 for the former and 512 for the latter). For Adam, we use the default hyperparameters set by PyTorch (Paszke et al., 2017).
Table 5: Hyperparameter search range for different methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Full</th>
<th>Prefix</th>
<th>Linear</th>
<th>FT2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guarantee $(\epsilon, \delta)$</td>
<td>$(3, \frac{1}{2}</td>
<td>D_{\text{train}}</td>
<td>)$</td>
<td>$(3, \frac{1}{2}</td>
</tr>
<tr>
<td>Clipping norm $C$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Batch size $B$</td>
<td>${512, 1024}$</td>
<td>${512, 1024}$</td>
<td>${512, 1024}$</td>
<td>${512, 1024}$</td>
</tr>
<tr>
<td>Learning rate $\eta$</td>
<td>$(200, 100, 30, 10, 3) \cdot 10^{-5}$</td>
<td>$(200, 100, 30, 10, 3) \cdot 10^{-5}$</td>
<td>$(200, 100, 30, 10, 3) \cdot 10^{-5}$</td>
<td>$(200, 100, 30, 10, 3) \cdot 10^{-5}$</td>
</tr>
<tr>
<td>LR decay</td>
<td>{yes, no}</td>
<td>{yes, no}</td>
<td>{yes, no}</td>
<td>{yes, no}</td>
</tr>
<tr>
<td>Epochs $E$</td>
<td>${10, 30, 50}$</td>
<td>${10, 30, 50}$</td>
<td>${10, 30, 50}$</td>
<td>${10, 30, 50}$</td>
</tr>
<tr>
<td>Weight decay $\lambda$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Noise scale $\sigma$</td>
<td>calculated numerically so that a DP budget of $(\epsilon, \delta)$ is spent after $E$ epochs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Hyperparameter search range for different methods (continued).

<table>
<thead>
<tr>
<th>Method</th>
<th>LoRA</th>
<th>RGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DP guarantee $(\epsilon, \delta)$</td>
<td>$(3, \frac{1}{2}</td>
<td>D_{\text{train}}</td>
</tr>
<tr>
<td>Clipping norm $C$</td>
<td>0.1</td>
<td>${0.1, 1, 10}$</td>
</tr>
<tr>
<td>Batch size $B$</td>
<td>${512, 1024}$</td>
<td>${512, 1024}$</td>
</tr>
<tr>
<td>Learning rate $\eta$</td>
<td>${300, 100, 30, 10, 3} \cdot 10^{-5}$</td>
<td>${300, 100, 30, 10, 3} \cdot 10^{-5}$</td>
</tr>
<tr>
<td>LR decay</td>
<td>{yes, no}</td>
<td>{yes, no}</td>
</tr>
<tr>
<td>Epochs $E$</td>
<td>${5, 10, 30, 50}$</td>
<td>${5, 10, 30, 50}$</td>
</tr>
<tr>
<td>Weight decay $\lambda$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rank $k$</td>
<td>${1, 2, 4, 8}$</td>
<td>${1, 2, 4, 8}$</td>
</tr>
<tr>
<td>Noise scale $\sigma$</td>
<td>calculated numerically so that a DP budget of $(\epsilon, \delta)$ is spent after $E$ epochs</td>
<td></td>
</tr>
</tbody>
</table>
J FULL RESULTS FOR EXPERIMENTS IN SECTION 5.2

Table 7: Full results on E2E from fine-tuning GPT-2.

<table>
<thead>
<tr>
<th>Method</th>
<th>DP Guarantee</th>
<th>Gaussian DP +CLT</th>
<th>Compose tradeoff func.</th>
<th>BLEU</th>
<th>NIST</th>
<th>METEOR</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
<th>CIDEr</th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>$\epsilon = 3$ $\approx 2.33$ $\epsilon \approx 2.67$</td>
<td>61.519 6.697</td>
<td>0.384 65.670 1.761</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 5.51$ $\epsilon \approx 6.98$</td>
<td>63.189 7.444</td>
<td>0.400 66.429 1.919</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>69.463 8.780</td>
<td>0.461 71.359 2.422</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoRA</td>
<td>$\epsilon = 3$ $\approx 2.68$ $\epsilon \approx 2.75$</td>
<td>58.153 5.463</td>
<td>0.370 65.773 1.581</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 6.77$ $\epsilon \approx 7.28$</td>
<td>63.389 7.449</td>
<td>0.407 67.525 1.948</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>69.682 8.822</td>
<td>0.463 71.709 2.491</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prefix</td>
<td>$\epsilon = 3$ $\approx 2.33$ $\epsilon \approx 2.67$</td>
<td>47.772 5.775</td>
<td>0.331 58.964 1.300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 5.51$ $\epsilon \approx 6.98$</td>
<td>49.263 6.276</td>
<td>0.349 60.730 1.496</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>68.845 8.722</td>
<td>0.456 70.805 2.418</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RGP</td>
<td>$\epsilon = 3$ $\approx 2.18$ $\epsilon \approx 2.59$</td>
<td>58.482 5.249</td>
<td>0.363 65.560 1.507</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 5.19$ $\epsilon \approx 6.89$</td>
<td>58.455 5.525</td>
<td>0.364 65.030 1.569</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>68.328 8.722</td>
<td>0.445 68.844 2.345</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top2</td>
<td>$\epsilon = 3$ $\approx 2.68$ $\epsilon \approx 2.75$</td>
<td>25.920 1.510</td>
<td>0.197 44.536 0.452</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 6.77$ $\epsilon \approx 7.28$</td>
<td>26.885 1.547</td>
<td>0.207 46.421 0.499</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>65.752 8.418</td>
<td>0.443 68.704 2.180</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>retrain</td>
<td>$\epsilon = 3$ $\approx 2.33$ $\epsilon \approx 2.67$</td>
<td>15.457 0.376</td>
<td>0.113 35.240 0.116</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 5.51$ $\epsilon \approx 6.98$</td>
<td>24.247 1.010</td>
<td>0.145 39.951 0.281</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>65.731 8.286</td>
<td>0.429 68.751 2.004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Full results on DART from fine-tuning GPT-2. Trend is consistent with results on E2E.

<table>
<thead>
<tr>
<th>Method</th>
<th>DP Guarantee</th>
<th>Gaussian DP +CLT</th>
<th>Compose tradeoff func.</th>
<th>BLEU</th>
<th>NIST</th>
<th>METEOR</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
<th>CIDEr</th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>$\epsilon = 3$ $\approx 2.28$ $\epsilon \approx 2.65$</td>
<td>0.294 62.815 40.773</td>
<td>52.063 31.025 0.887 -0.058</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 5.35$ $\epsilon \approx 6.95$</td>
<td>0.319 66.423 43.609</td>
<td>54.576 35.057 0.901 0.043</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>0.369 71.563 47.168</td>
<td>56.717 42.783 0.915 0.178</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LoRA</td>
<td>$\epsilon = 3$ $\approx 2.68$ $\epsilon \approx 2.76$</td>
<td>0.304 63.641 40.753</td>
<td>52.012 32.329 0.885 -0.029</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 6.68$ $\epsilon \approx 7.26$</td>
<td>0.318 66.336 43.056</td>
<td>54.082 34.163 0.899 0.036</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>0.366 71.192 47.336</td>
<td>57.430 42.254 0.915 0.182</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prefix</td>
<td>$\epsilon = 3$ $\approx 2.28$ $\epsilon \approx 2.65$</td>
<td>0.269 59.503 38.229</td>
<td>49.444 25.726 0.860 -0.144</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 5.35$ $\epsilon \approx 6.95$</td>
<td>0.297 64.009 41.581</td>
<td>52.602 30.463 0.892 -0.021</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>0.353 70.341 46.643</td>
<td>56.858 40.163 0.912 0.148</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RGP</td>
<td>$\epsilon = 3$ $\approx 2.10$ $\epsilon \approx 2.54$</td>
<td>0.265 58.688 37.202</td>
<td>49.011 25.748 0.873 -0.175</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 5.02$ $\epsilon \approx 6.86$</td>
<td>0.279 60.005 38.258</td>
<td>49.835 28.304 0.874 -0.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>0.324 65.667 42.617</td>
<td>53.477 35.551 0.895 0.022</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>top2</td>
<td>$\epsilon = 3$ $\approx 2.68$ $\epsilon \approx 2.76$</td>
<td>0.022 3.570 2.183</td>
<td>3.166 0.388 0.098 -1.952</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\epsilon = 8$ $\approx 6.68$ $\epsilon \approx 7.26$</td>
<td>0.054 11.475 7.054</td>
<td>10.042 2.453 0.240 -1.660</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>non-private</td>
<td>-</td>
<td>0.232 47.782 26.361</td>
<td>37.864 26.794 0.806 -0.593</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
K DETAILS FOR EXPERIMENTS IN SECTION 5.2

Sentence Classification. Results for RGP in Table 1 are taken from documented numbers in their released codebase. These results are under the DP guarantees of $(\epsilon, \delta) = (3, 10^{-5})$ or $(\epsilon, \delta) = (8, 10^{-5})$. These guarantees are strictly looser than our guarantees which are based on $\delta = \frac{1}{2}/D_{\text{train}}$ (recall the smallest dataset in this cohort of tasks has 60k+ records). The RGP numbers in Table 1 are higher than those reported in their paper (Yu et al., 2021c), since the latter numbers are not based on fine-tuning the official RoBERTa models.

Table-To-Text Generation. To evaluate models trained on E2E and DART, we evaluate generations from models obtained with beam search with a beam size of 5. For evaluation, we run the official pipeline for E2E, and the pipeline used in the GEM benchmark (Gehrmann et al., 2021) for DART.

Chit-Chat Dialog Generation. We built off Huggingface’s codebase of the winning entry of the ConvAI2 competition, and used their preprocessed training set with the minor modification of truncating the number of training examples to be a multiple of the batch size. The original ConvAI2 competition is aimed at advancing research on building engaging chatbots and also requested models to predict the mostly likely response given a list of candidates. The challenge included hits@1 as part of its suite of automatic metrics. For simplicity, we skip this step of predicting the most likely response for both training and evaluation. Results for the entry HuggingFace (ConvAI2 winner) in Table 3 are taken from the official validation set leader board. Our reimplementation of HuggingFace’s submission uses the released code for their winning entry, fine-tunes GPT with the default hyperparameters, and removes the classification loss for learning to predict the most likely response given candidates.

Our work involved tuning hyperparameter with models trained via DP-Adam on one private dataset and transferring such hyperparameters to other private datasets. Since different datasets may be of different sizes, transferring the batch size may cause a discrepancy in the effective noise multiplier across workloads with different datasets. Transferring the batch size based on hyperparameter tuning on small datasets to larger datasets can be particularly problematic, as the effective noise multiplier

---

References:

14 https://github.com/tuetschek/e2e-metrics
15 https://github.com/GEM-benchmark/GEM-metrics
16 https://github.com/huggingface/transfer-learning-conv-ai
17 http://convai.io/2018/
can be larger than ideal. In this work, we instead transferred the sampling rate $q$ across different datasets.

**M 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 related attacks, we follow the secret sharer framework (Carlini et al., 2019) 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 GPT-2. 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.

Table 9: Fine-tuning under DP prevents unintended memorization of downstream data. Numbers reported are exposure values estimated with the approximation by distribution model approach.

<table>
<thead>
<tr>
<th>Guarantee</th>
<th>Repetitions</th>
<th>$r = 1$</th>
<th>$r = 10$</th>
<th>$r = 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\epsilon = 3$</td>
<td></td>
<td>1.09 ± 0.86</td>
<td>1.32 ± 1.32</td>
<td>5.26 ± 4.20</td>
</tr>
<tr>
<td>non-private</td>
<td></td>
<td>13.82 ± 3.86</td>
<td>17.22 ± 0.00</td>
<td>17.78 ± 5.49</td>
</tr>
</tbody>
</table>

**N Templates and Label Words for Text-Infilling-Based Classification 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 (Gao et al., 2020) 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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Template</th>
<th>Label words</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2 $&lt;S_1&gt;$ It was [MASK] .</td>
<td>positive: great, negative: terrible</td>
<td></td>
</tr>
<tr>
<td>MNLI $&lt;S_1&gt;$ ? [MASK] , $&lt;S_2&gt;$</td>
<td>entailment: Yes, netural: Maybe, contradiction: No</td>
<td></td>
</tr>
<tr>
<td>QNLI $&lt;S_1&gt;$ ? [MASK] , $&lt;S_2&gt;$</td>
<td>entailment: Yes, not_entailment: No</td>
<td></td>
</tr>
<tr>
<td>QQP $&lt;S_1&gt;$ [MASK] , $&lt;S_2&gt;$</td>
<td>equivalent: Yes, not_equivalent: No</td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Templates and label words borrowed from the work by Gao et al. (2020).
O Uncurated Samples from Fine-Tuned Models

<table>
<thead>
<tr>
<th>Table</th>
<th>name : The Punter — Type : restaurant — food : Indian — price : cheap — customer rating : average — area : riverside — family friendly : no — near : Express by Holiday Inn</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-2 ($\epsilon = 3$)</td>
<td>The Punter is a cheap Indian restaurant near Express by Holiday Inn in the riverside area. It is not family-friendly.</td>
</tr>
<tr>
<td>GPT-2 ($\epsilon = 8$)</td>
<td>The Punter is a cheap Indian restaurant near Express by Holiday Inn in the riverside area. It is not family-friendly.</td>
</tr>
<tr>
<td>GPT-2-m ($\epsilon = 3$)</td>
<td>The Punter is a cheap Indian restaurant located in the riverside area near Express by Holiday Inn. It has an average customer rating and is not family-friendly.</td>
</tr>
<tr>
<td>GPT-2-m ($\epsilon = 8$)</td>
<td>The Punter is a restaurant providing Indian food in the cheap price range. It is located in the riverside area near Express by Holiday Inn. Its customer rating is average.</td>
</tr>
<tr>
<td>GPT-2-l ($\epsilon = 3$)</td>
<td>The Punter is a cheap Indian restaurant in the riverside area near Express by Holiday Inn. It is not family-friendly and has an average customer rating.</td>
</tr>
<tr>
<td>GPT-2-l ($\epsilon = 8$)</td>
<td>The Punter is a restaurant providing Indian food in the cheap price range. It is located in the riverside area near Express by Holiday Inn. Its customer rating is average.</td>
</tr>
<tr>
<td>Reference</td>
<td>The restaurant named The Punter has cheap Indian food and an average customer rating. It is near the Express by Holiday Inn on the riverside and is not family friendly.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>name : The Mill — Type : restaurant — food : English — price : moderate — customer rating : 3 out of 5 — area : city centre — family friendly : yes — near : Café Rouge</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-2 ($\epsilon = 3$)</td>
<td>The Mill is a moderately priced restaurant located in the city centre near Café Rouge.</td>
</tr>
<tr>
<td>GPT-2 ($\epsilon = 8$)</td>
<td>The Mill is a moderately priced restaurant located in the city centre near Café Rouge.</td>
</tr>
<tr>
<td>GPT-2-m ($\epsilon = 3$)</td>
<td>The Mill is an English restaurant located in the city centre near Café Rouge. It is moderately priced and has a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>GPT-2-m ($\epsilon = 8$)</td>
<td>The Mill is a moderately priced restaurant located in the city centre near Café Rouge. It is child friendly and has a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>GPT-2-l ($\epsilon = 3$)</td>
<td>The Mill is a moderately priced English restaurant in the city centre near Café Rouge. It is child friendly and has a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>GPT-2-l ($\epsilon = 8$)</td>
<td>The Mill is a kid friendly English restaurant in the city centre near Café Rouge. It has a moderate price range and a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>Reference</td>
<td>Serving moderately priced English food with a 3 out of 5 customer approval, The Mill restaurant is kid friendly and conveniently located at the city centre near the Café Rouge.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>name : The Vaults — Type : pub — food : Japanese — price : high — customer rating : 3 out of 5 — area : city centre — family friendly : yes — near : Raja Indian Cuisine</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-2 ($\epsilon = 3$)</td>
<td>Located near Raja Indian Cuisine in the city centre, The Vaults is a family-friendly Japanese pub with a high customer rating.</td>
</tr>
<tr>
<td>GPT-2 ($\epsilon = 8$)</td>
<td>The Vaults is a Japanese pub in the city centre near Raja Indian Cuisine. It has a high price range and a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>GPT-2-m ($\epsilon = 3$)</td>
<td>The Vaults is a Japanese pub located in the city centre near Raja Indian Cuisine. It has a high price range and a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>GPT-2-m ($\epsilon = 8$)</td>
<td>The Vaults is a Japanese pub located in the city centre near Raja Indian Cuisine. It has a high price range and a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>GPT-2-l ($\epsilon = 3$)</td>
<td>The Vaults is a Japanese pub in the city centre near Raja Indian Cuisine. It has a high price range and a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>GPT-2-l ($\epsilon = 8$)</td>
<td>The Vaults is a child friendly Japanese pub in the city centre near Raja Indian Cuisine. It has a high price range and a customer rating of 3 out of 5.</td>
</tr>
<tr>
<td>Reference</td>
<td>Located near the Raja Indian Cuisine at the city centre, is family-friendly pub The Vaults. Serving Japanese food, The Vaults, is high priced with a 3 out of 5 customer satisfaction rating.</td>
</tr>
</tbody>
</table>

Table 11: Fully fine-tuned GPT-2, GPT-2-medium, and GPT-2-large generations with E2E test table entries.
Table Real Madrid Castilla : manager : Luis Miguel Ramis — Abner (footballer) : club : Real Madrid Castilla — Abner (footballer) : club : C.D. FAS

GPT-2 ($\epsilon = 3$) Luis Miguel Ramis played for Real Madrid Castilla and played for C.D. FAS.
GPT-2 ($\epsilon = 8$) Luis Miguel Ramis is the manager of Abner (footballer) who plays for Real Madrid Castilla.
GPT-2-m ($\epsilon = 3$) Luis Miguel Ramis is the manager of Real Madrid Castilla. He plays for C.D. FAS.
GPT-2-m ($\epsilon = 8$) Luis Miguel Ramis is the manager of Real Madrid Castilla. He plays for C.D. FAS.
GPT-2-l ($\epsilon = 3$) Luis Miguel Ramis is the manager of Real Madrid Castilla and Abner (footballer) plays for C.D. FAS.
GPT-2-l ($\epsilon = 8$) Luis Miguel Ramis is the manager of Real Madrid Castilla and Abner (footballer) plays for C.D. FAS.

Reference Footballer, Abner, plays C.D. FAS. and Real Madrid Castilla, the manager of which, is Luis Miguel Ramis.


GPT-2 ($\epsilon = 3$) The capital of the United States is Washington, D.C. and is part of Benton County, Oregon.
GPT-2 ($\epsilon = 8$) The capital of the United States is Washington, D.C. and is part of Benton County, Oregon.
GPT-2-m ($\epsilon = 3$) Albany, Oregon is part of Benton County, Oregon in the United States where Asian Americans are an ethnic group.
GPT-2-m ($\epsilon = 8$) Albany, Oregon is part of Benton County, Oregon in the United States where Asian Americans are an ethnic group.
GPT-2-l ($\epsilon = 3$) Albany, Oregon is part of the United States where Asian Americans are an ethnic group and the capital is Washington D.C.
GPT-2-l ($\epsilon = 8$) Albany, Oregon is part of the United States where Asian Americans are an ethnic group and the capital is Washington D.C.

Reference The Asian Americans are an ethnic group in the United States, which has the capital city of Washington DC. It is also the location of Albany, part of Benton County in Oregon.


GPT-2 ($\epsilon = 3$) A Loyal Character Dancer is an English language spoken in the United States where Native Americans are the ethnic group.
GPT-2 ($\epsilon = 8$) A Loyal Character Dancer is written in English and is spoken in Great Britain. Native Americans are an ethnic group in the United States.
GPT-2-m ($\epsilon = 3$) A Loyal Character Dancer is written in English and is written in the United States where Native Americans are an ethnic group.
GPT-2-m ($\epsilon = 8$) A Loyal Character Dancer is written in English and is written in the United States where Native Americans are an ethnic group.
GPT-2-l ($\epsilon = 3$) A Loyal Character Dancer is written in English and is written in the United States where Native Americans are an ethnic group.
GPT-2-l ($\epsilon = 8$) A Loyal Character Dancer is written in English, which is spoken in Great Britain and the United States. Native Americans are an ethnic group in the United States.

Reference A Loyal Character Dancer is written in English which is spoken in Great Britain. The book is published in the United States where the Native Americans are an ethnic group.

Table 12: Fully fine-tuned GPT-2, GPT-2-medium, and GPT-2-large generations with DART test table entries.
Table 13: Fully fine-tuned GPT-2-medium and DialoGPT-medium generations for Persona-Chat validation examples.

<table>
<thead>
<tr>
<th>Persona</th>
<th>History</th>
<th>DialoGPT-medium ($\epsilon = 3$)</th>
<th>DialoGPT-medium ($\epsilon = 8$)</th>
<th>GPT-2-m ($\epsilon = 3$)</th>
<th>GPT-2-m ($\epsilon = 8$)</th>
</tr>
</thead>
</table>
| "i am 40 years old.", "i work as a car salesman.", "my wife spends all my money.", "i hate my job.", "i'm planning on getting a divorce."
| "i know the struggle, i volunteer at a homeless shelter.", "great, i hope you don't get to see me there.", "i hope not either, would not wish it on my worse enemy.", "between wife, job with no money, i am thinking now of divorce.", "i've given it a few thoughts before"
| i wish i had a better idea.
| i don't have any money, i do not know how i could manage.
| do you give demonstrations of your skills?
| "my mother is a dentist.", "my favourite artist is travis scott.", "i have one older sister.", "i am currently looking for a job preferably at a restaurant.", "i like playing basketball."
| "hello, how are you tonight?"
| i'm good, good luck.
| i don't have any money, i do not know how i could manage.
| do you give demonstrations of your skills?
| "my mom passed away when i was young.", "i work in the insurance business.", "my father is in the army.", "i recently was fired and had to get a new job."
| "god isn't really real!", "to each their own, but i believe he is.", "my children are gone now!", "i lost my mom at a young age, 'my wife took them all'"
| she was a great mother.
| they were so young, they were so crazy.
| what happened to her?
| sorry to hear that, i barely see my dad he's in the army.
| "i am also a graduate student.", "i walk dogs for a living.", "my parents taught me survival skills.", "i am an avid kayaker.", "i enjoy reading journals and guides related to psychology."
| "hi, how are you today?!", "good evening, how are you tonight? i'm well!", "i am doing good, i just finished playing ping pong.", "i am trying to write invoices for my dog walking business.", "that seems like hard work, i like to play the violin in my free time."
| i have a lot of dogs, i walk them on my walks.
| that is a very rewarding hobby, i have a dog named jasper.
| it is, i love learning to play the violin.
| i love the violin, i do not play anything, i kayak for fun however.

Reference
P Additional Related Work

Differentially Private 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. (2019); Tramèr & Boneh (2020); Abadi et al. (2016) consider transferring features learned on public datasets to simplify the subsequent private learning task. On the other hand, Zhou et al. (2020); Kairouz et al. (2020) remove the ambient dimension dependence of DP noise by identifying subspaces in which private gradients lie and would be privatized. Yu et al. (2021b;c) make such ideas practical and demonstrate improved results on private learning benchmarks. Zhang et al. (2021) applied the sparse vector technique to learning wide neural layers to reduce the amount of injected noise. Our work mostly falls under the first camp – improving private learning through simplifying the learning task. 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 enforce DP which are not based on gradient perturbation (Papernot et al., 2018; 2016). These methods typically require extra public data and are not the present focus.

Parameter-Efficient Fine-Tuning. Recent developments on pretrained model adaptation have produced a wide range of parameter-efficient fine-tuning methods for both vision and language tasks. We briefly summarize these, grouping by category. Approaches based on optimizing prompt-like constructions for NLP tasks include prefix-tuning (Li & Liang, 2021), P-tuning (Liu et al., 2021), and prompt-tuning (Lester et al., 2021). Adapter-based methods insert small subnetworks inside pretrained Transformers (Houlsby et al., 2019; Rücklè et al., 2020; Pfeiffer et al., 2020). Methods that optimize low-rank matrices include the work by Hu et al. (2021); Mahabadi et al. (2021). In addition, there are adaptation methods that only optimize biases for vision (Cai et al., 2020) and language tasks (Ben Zaken et al., 2021). Our evaluation in Section 5.2 covered the most representative methods that generally have state-of-the-art non-private learning performance (at the time of writing) for the range of NLP tasks studied in this paper.

Speeding Up DP-SGD. Apart from the work by Lee & Kifer (2020) and Subramani et al. (2020), there is an approach that approximates per-example gradient norms through the combination of random projection and forward-mode autodiff (Bu et al., 2021). While faster than vanilla private learning, this approach has the drawback of increased privacy spending and having an extra hyperparameter. Our ghost clipping technique, while only suited for Transformers applied to sequential data, does not introduce new hyperparameters.

Alternative Clipping Strategies. While there are alternative clipping strategies in the literature that show improvements on simple tasks (Pichapati et al., 2019; Asi et al., 2021), we have opted to study the simplest strategy that clips gradients by their Euclidean norm. We leave the study of these algorithms for NLP tasks to future work.

Concurrent Work. We are made aware of a concurrent work that also studies fine-tuning large language models under DP (Yu et al., 2021a). This work presents initial successes on fine-tuning under DP with low-rank methods such as LoRA. Our experiments on language generation (see Section 5.2 and Table 2) demonstrate similar findings. Yet, we moreover show that full fine-tuning with good hyperparameters attains similar performance and possesses similar model scaling properties, which was raise by Yu et al. (2021a) as interesting open questions to pursue. Lastly, our private fine-tuning results for sentence classification may be far from optimal, since we used hyperparameters mostly transferred from tuning on the E2E language generation task.

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 could be sampled and used for analysis. Previous work employed generative adversarial networks and focused primarily on image or tabular datasets (Torkzadehmahani et al., 2019; Neunhoeffer et al., 2020; Chen et al., 2020; Torfi et al., 2020). Perhaps more related is the work by Bommasani et al. (2019) which attempted fine-tuning GPT-2 on medical datasets to generate synthetic records but did not report any quantitative results.
Q  ESTIMATES OF RUN-TIME IN PRACTICE

The actual run-time of algorithms depends on implementation details. Here, we outline estimates of the run-time for full fine-tuning with DP-Adam on tasks considered in the paper. These numbers are based on running with a single RTX 3090 with PyTorch==1.9.0. Fine-tuning GPT-2 on E2E and DART takes less than 10 minutes per epoch, and fine-tuning for 10 epochs results in reasonably performing models. The time to fine-tune RoBERTa-base on classification tasks depends on the size of the dataset. It takes less than 10 minutes per epoch on the smallest SST-2, whereas for the largest MNLI, it takes less than an hour per epoch.

R  EFFECTS OF VARYING \( \epsilon \) AND \( \delta \) ON DP FULL FINE-TUNING PERFORMANCE

Figure 9 shows how different values of \( \epsilon \) and \( \delta \) affect the performance of full fine-tuning under DP.

Figure 9: **Left:** \( \delta \) affects performance marginally. **Right:** \( \epsilon \) affects performance more significantly when \( \epsilon < 2 \). Errorbars are one standard deviation away from the mean over five independent runs.

S  DP-ADAM VS DP-SGD

Our work focused on DP-Adam as opposed to DP-SGD, since Adam is more commonly used for non-private language model fine-tuning. While the two algorithms differ in their parameter update rule, the basic gradient privatization procedure is the same. We performed additional experiments fine-tuning GPT-2 on the E2E dataset with DP-SGD (with freshly tuned learning rate and clipping norm values) and observed that its performance (test set BLEU 63.175 at \((\epsilon, \delta) = (8, 10^{-5})\)) is on par with DP-Adam (test set BLEU 63.189 at \((\epsilon, \delta) = (8, 10^{-5})\)).

T  SUBTLETIES OF IMPLEMENTING DP MIXED PRECISION TRAINING

Mixed precision training (Micikevicius et al., 2017) accelerates updates by storing certain tensors in half-precision. To mitigate negative effects caused by potential arithmetic underflow, usual implementations upscale the loss with an adaptive factor pre-backpropagation and downscale the gradients with the same factor post-backpropagation. The scaling factor is adapted based on whether underflow is observed during training.

Special care needs to be taken when combining gradient privatization with mixed precision training. One implementation that ensures similar results across full and mixed precision training (1) upcales the loss with the adaptive factor \( K \), (2) clips per-example gradients by \( CK \), (3) adds to the sum of clipped gradients the usual Gaussian noise multiplied by \( K \), and (4) downcales noisy gradients by \( K \). This is the implementation that we adopt, and we were able to obtain similar results on the E2E dataset with and without mixed precision.

One alternative implementation (a) upcales the loss with the adaptive factor \( K \), (b) clips per-example gradients by \( C \), (c) adds the usual Gaussian noise to the sum of clipped gradients, and (d) downcales noisy gradients by \( K \). The main difference between this procedure and the prior is whether the factor...
is considered during clipping and noising. This procedure, while having the same DP guarantee, typically does not result in similar results as full precision when the same hyperparameters are used across the two settings (even with an optimizer like Adam which self-adjusts the magnitude of updates with accumulated empirical second moments). We also identified that this implementation is also the primary reason that a prior work’s code does not reproduce good results when run in full precision.²⁹

U Fine-tuning with RGP and the Text-infilling Objective

Recall Section 3.2 and Table 1 showed that private full fine-tuning with a text-infilling objective leads to improved classification results. Here, we show that the infilling objective is also helpful when one privately fine-tunes with the RGP method for a classification task. To study this, we reimplemented the RGP method and tuned the hyperparameters in full precision. Fixing all hyperparameters, we compared models trained with and without infilling. Table 14 confirms that fine-tuning with RGP based on infilling is generally helpful across the considered tasks. Note the aim of this experiment is not obtain state-of-the-art performance, but rather to study the effect of using the text-infilling objective. Thus, we expect these results could generally be further improved with more extensive hyperparameter search.

Table 14: Text-infilling objective improves the performance of RGP for classification. Numbers are averaged over three independent runs.

<table>
<thead>
<tr>
<th>Method</th>
<th>ε = 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MNLI-(m/mm)</td>
</tr>
<tr>
<td>RGP (RoBERTa-base)</td>
<td>79.79/80.40</td>
</tr>
<tr>
<td>RGP + infilling (RoBERTa-base)</td>
<td>81.97/82.28</td>
</tr>
</tbody>
</table>

V Which method shall I use for private fine-tuning?

While we were able to obtain similar results on the E2E and DART datasets with full fine-tuning and LoRA (Tables 1 and 8), we encountered significant difficulties in fine-tuning for dialog generation (even non-privately) when not updating the embedding layer and language modeling head – perplexity was much worse, and generations frequently contained seemingly arbitrary characters. This result suggests that while full fine-tuning may be more computationally intensive than lightweight approaches at times, its simplicity (involving few design decisions) makes it an attractive first option when compute resource is sufficient.