DP-MemArc: Differential Privacy Transfer Learning for Memory Efficient Language Models

Yanming Liu[1], Xinyue Peng[2], Yuwei Zhang[8], Xiaolan Ke[3], Songhang Deng[4], Jiannan Cao[5], Chen Ma[6], Mengchen Fu[7], Tianyu Du[1*], Sheng Cheng[1], Xun Wang[1], Jianwei Yin[1], Xuhong Zhang[1*]

[1]Zhejiang University
[2]Southeast University
[3]Harvard University
[4]University of California, Los Angeles
[5]Massachusetts Institute of Technology
[6]Renmin University of China
[7]The University of Tokyo
[8]Tongji University

{oceann24, zhangxuhong, zjradty, zjuyjw}@zju.edu.cn, xinyuepeng@seu.edu.cn, jiannan@mit.edu, songh00@ucla.edu

Abstract

Large language models have repeatedly shown outstanding performance across diverse applications. However, deploying these models can inadvertently risk user privacy. The significant memory demands during training pose a major challenge in terms of resource consumption. This substantial size places a heavy load on memory resources, raising considerable practical concerns. In this paper, we introduce DP-MemArc, a novel training framework aimed at reducing the memory costs of large language models while emphasizing the protection of user data privacy. DP-MemArc incorporates side network or reversible network designs to support a variety of differential privacy memory-efficient fine-tuning schemes. Our approach not only achieves about 2.5 times in memory optimization but also ensures robust privacy protection, keeping user data secure and confidential. Extensive experiments have demonstrated that DP-MemArc effectively provides differential privacy-efficient fine-tuning across different task scenarios.

Introduction

Large language models (LLMs) (Radford et al. 2019; Hoffmann et al. 2022; Chowdhery et al. 2023; Touvron et al. 2023) have already demonstrated their capabilities across various domains, excelling in a wide range of generation and comprehension tasks (Bang et al. 2023; Robinson, Rytting, and Wingate 2022; Li, Zhang, and Zhao 2022). However, complete training of LLMs demands significant computational resources and time, making it inconvenient to adapt the model in downstream tasks (Liu et al. 2022a). There exist several methods that offer solutions for parameter-efficient fine-tuning (Dettmers et al. 2024; Houlsby et al. 2019; Hu et al. 2021). These approaches achieve highly effective downstream task fine-tuning results by adjusting only a small number of parameters. The goal of such methods is to enable LLMs to adapt to small-scale features in a relatively small dataset, thereby accomplishing specific downstream tasks. Unfortunately, for LLMs, we often encounter situations where the available dataset is small and proprietary, raising concerns about privacy (Bu et al. 2024; Yu et al. 2021; Finlayson, Swayamdipta, and Ren 2024). Additionally, the training of LLMs requires substantial training memory (Wang et al. 2023), making it challenging to train on parameter-efficient fine-tuning.

A recent line of work that focuses on fine-tuning large models using differential privacy (DP) solutions, including both full parameter fine-tuning and parameter efficient fine-tuning approaches (Duan et al. 2024; Bu et al. 2024; Yu et al. 2021). These solutions employ a method called Differential Privacy Stochastic Gradient Descent (DP-SGD) (Yu et al. 2019). The training data is protected by implementing gradient clipping and adding Gaussian noise during each iteration to ensure privacy. Compared to traditional fine-tuning approaches, DP allows for downstream task handling with only a small loss in accuracy while maintaining a theoretical private guarantee (Yu et al. 2021). These approaches exhibit good performance across a variety of tasks and settings. However, these methods still have issues with training memory. In previous research, differential privacy has imposed larger computational and storage overheads, making training such large models challenging in resource-constrained scenarios. Additionally, existing efficient parameter fine-tuning with differential privacy schemes has only achieved marginal reductions in memory overhead, with insufficient optimization efficiency in memory resources (Li et al. 2022; Ke et al. 2024). As models continue to grow, the demand for both memory efficiency and privacy in such scenarios also increases.

To address this issue, we propose a solution called **D**ifferential **P**rivate **Mem**ory efficient transfer **Arc**hitecture (**DP-MemArc**), a framework for training in scenarios that involve both privacy-protection and memory efficient transfer learning. In our framework, we explore two efficient

*Corresponding Author.

Module	Forward pass	Back-propagation	Ghost norm in Book-Keeping	Opacus grad instantiation	Opacus sum of weighted grad
Time complexity	$2BTpd$	$4BTpd$	$2BT^2(p+d)$	$2BTpd$	$2Bpd$
Space complexity	$pd + BTd$	$BT(p+d) + pd$	$2BT^2$	Bpd	0

Table 1: The time and space complexity of the training process of a model under a single-layer MLP. While opacus is a codebase for vanilla differential private method implementation.

methods for parameter-efficient fine-tuning, DP-MemArc$_{\text{side}}$ and DP-MemArc$_{\text{rev}}$, which save memory usage from different perspectives. In this setup, our approach not only achieves competitive performance but also significantly reduces training memory usage. Experiments on different datasets and models have thoroughly demonstrated the effectiveness and potential of our approach. Therefore, our work effectively addresses the issue of insufficient memory in private fine-tuning for language models, while also providing alternative privacy fine-tuning solutions.

In summary, our contributions in this paper are as follows:

- We propose a framework called DP-MemArc, which enables efficient fine-tuning of language models with lower training memory in differential privacy fine-tuning. This framework contains two memory optimization methods, reducing the memory requirements for privacy training of language models.
- We conduct a systematic analysis of the relationship between training memory requirements and network architecture. We elucidate the characteristics of fine-tuning memory cost under different network architectures, demonstrating favorable downstream task performance in differential privacy.
- We evaluate our DP-MemArc framework on multiple datasets and models. The results show promising performance across various dimensions, with a substantial improvement in training memory.

Preliminaries

Memory Footprint on Language Model

We consider a N multilayer perception: $\boldsymbol{x}_N = f_N(f_{N-1}(...f_2(f_1(\boldsymbol{x}_0))))$, where x_0 as the initial PLM input, the i^{th} layer $\boldsymbol{x}_i = f_i(\boldsymbol{x}_{i-1}) = \sigma_i(\mathbf{W}_i\boldsymbol{x}_{i-1})$ consists of a weight matrix $\mathbf{W}_i$ and a nonelinear function $\boldsymbol{\sigma}_i$. For the format simplicity, the bias term is ignored. We denote $\boldsymbol{h}_i = \mathbf{W}_i\boldsymbol{x}_{i-1}$ as the hidden states for the pre-activation of i^{th} layer. In backpropagation with loss $\mathcal{L}$, the gradient of W_i is calculated with respect to x_i using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_N}\Big(\prod_{j=i+1}^{N} \frac{\partial \boldsymbol{x}_j}{\partial \boldsymbol{h}_j}\frac{\partial \boldsymbol{h}_j}{\partial \boldsymbol{x}_{j-1}}\Big)\frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{h}_i}\frac{\partial \boldsymbol{h}_i}{\partial \mathbf{W}_i} \tag{1}$$

Denoting the derivative of $\boldsymbol{\sigma}$ is $\boldsymbol{\sigma}'$, then the equation could simplified as:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_N}\Big(\prod_{j=i+1}^{N} \boldsymbol{\sigma}'_j\mathbf{W}_j\Big)\boldsymbol{\sigma}'_i\boldsymbol{x}_{i-1}. \tag{2}$$

Thus, in training memory, the core consumption lies in the states of model weights $\{\mathbf{W}_i\}_{i=1}^N$ and derivative activation functions state $\{\boldsymbol{\sigma}'\}_{i=1}^N$ along the backpropagation path, as well as the optimizer states used during gradient updates. The optimizer states are directly related to the updated model parameters $\{\Delta\mathbf{W}\}$.

Assuming the batch size is B, the length of the input sequence is T, the model input and output dimension is d and p, for a standard linear layer $\boldsymbol{x}_i = \sigma_i(\mathbf{W}_i\boldsymbol{x}_{i-1})$, the forward pass stores the intermediate states of the model and the model weights with the memory complexity of $O(pd + BTd)$, while the backward pass is responsible for storing the activation states during the gradient update process, the results of the output gradients, and the corresponding parameter gradients, with the total memory complexity of $O(BT(p+d) + pd)$.

Deep Learning with Differential Privacy

Differential Privacy (Dwork et al. 2006; Abadi et al. 2016) algorithms demonstrate that under this formulation, the model's output cannot significantly help determine whether an individual record exists in the input dataset through certain mathematical derivations. The formal definition is recalled as follows:

Definition 1 *(Differential Privacy). Given a domain $\mathcal{D}$, any two datasets D, $D' \subseteq \mathcal{D}$ that differ in exactly one record are called neighboring datasets. A randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is (ϵ, δ)-differential private if for all neighboring datasets D and D' and all $T \subseteq \mathcal{R}$,*

$$\Pr[\mathcal{M}(D) \subseteq T] \leq e^{\epsilon}\Pr[\mathcal{M}(\mathcal{D}') \subseteq T] + \delta.$$

DP-optimizer. To train a privacy-preserving language model, the current approach involves providing differential privacy guarantees when computing gradients and applying these guarantees to optimizers such as SGD or Adam (Abadi et al. 2016; Mironov 2017; Koskela, Jälkö, and Honkela 2020). This approach incorporates steps involving per-example gradient clipping $\mathbf{G}_l = \sum C_i \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}_{(l)}}$ and adding Gaussian noise $\mathcal{N}(0, \mathbf{I})$ on gradient $\mathbf{G}$. Where C_i is the per-sample clipping factor.

Book-keeping. To avoid the significant memory overhead caused by storing gradients for each sample during initialization, Bu et al. (2023) proposed a method BK utilizing gradient norms. Using the GhostClip (Goodfellow 2015; Bu, Mao, and Xu 2022) strategy, the gradient norm of each sample is calculated.

$$\left\| \frac{\partial \mathcal{L}_i}{\partial \mathbf{W}} \right\|_{\mathrm{F}}^2 = \mathrm{vec}\Big(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_i} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_i}^{\top} \Big) \cdot \mathrm{vec}\left(\boldsymbol{x}_i \boldsymbol{x}_i^{\top} \right) \tag{3}$$

Based on the gradient norms, clipping factors C_i and clipping matrices $\mathbf{C}$ are generated, which are then used to compute the sum of clipped gradient in batches $\mathbf{G}_l = \boldsymbol{x}_{(l)}^{\top} \mathrm{diag}(\mathbf{C}) \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{(l)}}$. It is necessary to retain the complexity of the original DP to avoid issues related to length-dependent time and memory complexity when handling long-text question answering. BK-MixOpt strikes a balance between the two. It compares the theoretical complexity in terms of both dimension and context length, and selects the optimal memory complexity $O(min\{2BT^2, Bpd\})$ as the basis for computation.

Methodology

To address the issue of excessive memory consumption during differential privacy training, we have designed two methods: DP-MemArc$_{\text{side}}$ and DP-MemArc$_{\text{rev}}$. These methods help reduce training memory usage in different aspects.

Side Network Design

In general, most of the memory consumption comes from the model weights and the states of activation functions in the backward propagation path. By minimizing the consumption of these two parts as much as possible, the memory usage during training can be correspondingly reduced. This necessitates finding a reasonable design to address this situation.

Assume the base model is $\mathbf{F}$, the model's pre-trained weights, input, output, and parameters are $\mathbf{W}_{\mathrm{p}}$, x_0, y, θ. The model could formulated as:

$$y = \mathbf{F}(\mathbf{W}_{\mathrm{p}}, \theta; x_0). \tag{4}$$

Traditional parameter-efficient fine-tuning methods cannot avoid the memory consumption associated with the model weights of frozen parameters in the backward propagation path, which can formulated as:

$$y = \mathbf{F}(\mathbf{W}_{\mathrm{p}} + \Delta\mathbf{W}, \theta + \Delta\theta; x_0). \tag{5}$$

We hope to find a form that remains distinct from the original form when adjusting parameters. That is, there exists such a form:

$$y = \alpha \mathbf{F}_1(\mathbf{W}_{\mathrm{p}}, \theta; x_0) + \beta \mathbf{F}_2(\Delta\mathbf{W}, \Delta\theta; x_0). \tag{6}$$

In this form, Side-tuning (Zhang et al. 2020) meets the requirements. Side-tuning introduces a side network that learns the knowledge and features of new tasks, relying on the knowledge contained in the trained model parameters, thus supporting the processing of downstream tasks.

Assuming the input and output dimension of the side network is r, we add a liner layer at the last layer of the side network to ensure dimension consistency. We use Book-keeping (Bu et al. 2023) for differential privacy fine-tuning. The memory cost includes both the forward and backward propagation processes. For the forward process, the bilateral forward propagation memory consumption needs to be taken into account, with the complexity of $O(pd+r^2+BT(d+r))$. For the backward process, gradients need to be computed only in the side network, with a complexity of $O(2BTr + r^2)$ for gradients and $O(2BT^2)$ for Ghost Norm.

When $r \ll d$, the side tuning approach significantly reduces the training memory required for privacy fine-tuning. However, at sufficiently small r, the performance of side tuning also deteriorates significantly. To integrate the information of a trained model effectively into side networks, we adopt the LST (Sung, Cho, and Bansal 2022) method. This involves passing the intermediate layer information from the pre-trained model to the si de through a linear layer f'. We denote this method as DP-MemArc$_{\text{side}}$.

$$y = \mathbf{F}_2(\Delta\mathbf{W}, \Delta\theta; y_{i-1} + f_i'(x_i), y_0 = x_0). \tag{7}$$

Using the DP-MemArc$_{\text{side}}$ method, we can maintain a good performance in fine-tuning our side network with differential privacy. When $d/r = 8$, LST (Sung, Cho, and Bansal 2022) and DP-MemArc$_{\text{side}}$ achieves an empirically optimal ratio of training memory to performance.

Reversible Network Design

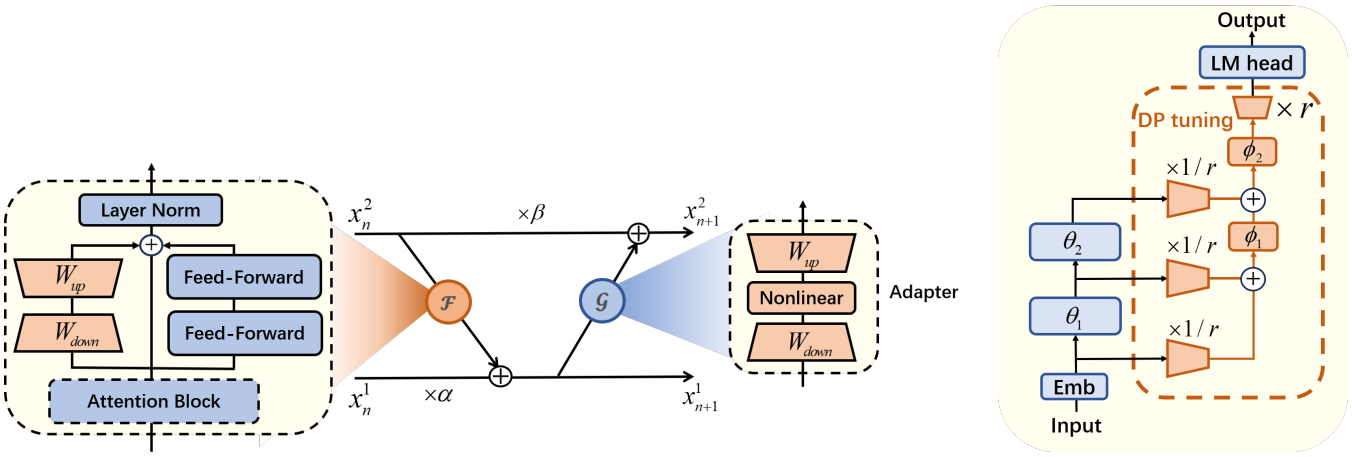
Due to the significant amount of training memory required to store the state of activation functions during batch processing, a large portion of memory is consumed by saving activation states $\{\boldsymbol{\sigma_i}\}_{i=1}^N$. Regular parameter-efficient fine-tuning methods cannot effectively address this issue. DP-MemArc$_{\text{side}}$ reduces the memory needed to store activation functions by compressing the dimensions of the activation functions. However, this method still consumes some memory. If we could deduce the intermediate states from the output results in reverse, we could further reduce the memory demand for storing activation states.

For reversible networks (Gomez et al. 2017; Liao, Tan, and Monz 2023), the following form is usually satisfied.

$$\begin{aligned} \boldsymbol{x}_{i+1}^1 &= \alpha \boldsymbol{x}_i^1 + \mathcal{F}_i(\boldsymbol{x}_i^2), \\ \boldsymbol{x}_{i+1}^2 &= \beta \boldsymbol{x}_i^2 + \mathcal{G}_i(\boldsymbol{x}_{i+1}^1), \\ \boldsymbol{x}_i^2 &= (\boldsymbol{x}_{i+1}^2 - \mathcal{G}_i(\boldsymbol{x}_{i+1}^1))/\beta, \\ \boldsymbol{x}_i^1 &= (\boldsymbol{x}_{i+1}^1 - \mathcal{F}_i(\boldsymbol{x}_i^2))/\alpha. \end{aligned} \tag{8}$$

We can obtain the corresponding activation function values $\boldsymbol{\sigma_i} = \sigma_i(\mathbf{W}_i \boldsymbol{x}_{i-1})$ from the intermediate states $\{\boldsymbol{x_i}\}_{i=1}^N$ of the model and calculate their derivatives, thus avoiding the need to store each activation function value.

To enable the two modules of the reversible network to both acquire new features and retain the knowledge of the pre-trained model, For module $\mathcal{F}$, we introduced the LoRA (Hu et al. 2021) architecture into the FFN layer of the model, continuing the traditional LoRA approach. Meanwhile, for



(a) Designs of DP-MemArc$_{\text{rev}}$.

(b) Designs of DP-MemArc$_{\text{side}}$.

Figure 1: Two different DP-MemArc designs, the left represents reversible network design, and the right represents side network design. The trainable parameters are fine-tuned using the differential privacy BK-MixOpt method.

module $\mathcal{G}$, we used Adapters (Houlsby et al. 2019) as trainable parameters to adapt to downstream tasks. Since the network is reversible, we only need to use constant reproducible space to compute $\boldsymbol{x}_i^2$ and $\boldsymbol{x}_i^1$ for each layer, which satisfies the requirements for the subsequent backpropagation calculations. We denote this method as DP-MemArc$_{\text{rev}}$.

For reversible networks, we have the following derivation steps. At the beginning of training, when the output of the adapter output is close to 0. $x_n \approx \mathcal{F}_n(\boldsymbol{x}_{n-1})$. Assume that x_0^1 and x_0^2 comes from the initial input x, we have:

$$\boldsymbol{x}_1^1 = \alpha \boldsymbol{x}_0^1 + \mathcal{F}_1(\boldsymbol{x}_0^2) \approx \alpha x_0 + x_1, \tag{9}$$

$$\boldsymbol{x}_1^2 = \beta \boldsymbol{x}_0^2 + \mathcal{G}_1(\boldsymbol{x}_1^1) = \beta x_0 + \mathcal{G}_1(\boldsymbol{x}_1^1) \approx \beta \boldsymbol{x}_0. \tag{10}$$

When $\alpha \to 0$, we have $\boldsymbol{x}_1^1 = x_1$, $\boldsymbol{x}_1^2 = \beta x_0$. We achieve a relatively stable state of the reversible network by exchanging output values $\boldsymbol{x}_1^1 = \beta \boldsymbol{x}_0$, $\boldsymbol{x}_1^2 = x_1$. Through iterative computation like this, the model can be satisfied as $\boldsymbol{x}_n^1 \approx \beta \boldsymbol{x}_{n-1}$, $\boldsymbol{x}_n^2 \approx x_n$. We generate the final output as $x = (\boldsymbol{x}_N^1 + \boldsymbol{x}_N^2)/2$. In this way, when training reversible models, the continuity of the model's representation can be maintained, and inference and learning for downstream tasks can be facilitated based on pre-trained models.

During the backpropagation process in our reversible network, the intermediate states of the model can be obtained by computing the reverse steps. As a result, the training memory required for activation values can be reduced by reusing a fixed-size replaceable memory. The primary training memory consumption of the model comes from storing the output gradients, storing the parameter gradients, and the computational memory required by the Ghostnorm method. Here, we also employ the BK-MixOpt algorithm to calculate the norm of the samples, thereby obtaining the corresponding gradient values. During training, we set batch sizes to 32.

Experimental Setup

We designed a series of experiments covering different models and datasets to evaluate the performance of our methods. The specific experimental design is as follows.

Models. We used the RoBERTa-large (Liu et al. 2019), GPT-2-large (Radford et al. 2019) model as our base models. These models will be fine-tuned according to the corresponding downstream tasks, and the performance of the fine-tuned models will be evaluated under different privacy constraints.

Baselines. We compare the two methods against multiple baselines, including DP-LoRA (Hu et al. 2021; Yu et al. 2021), DP-Adapter (Houlsby et al. 2019; Yu et al. 2021), DP-BiTFiT (Bu et al. 2024; Zaken, Goldberg, and Ravfogel 2022), and PromptDPSGD (Duan et al. 2024; Lester, Al-Rfou, and Constant 2021). These methods are all privacy-preserving fine-tuning approaches with opacus DP (Yousefpour et al. 2021), and we test them on the same training data to ensure fairness of comparison.

Datasets. We conduct experiments on five datasets. Four from the GLUE benchmarks (Wang et al. 2018), which cover different NLP tasks. **MNLI**: the MultiGenre Natural Language Inference Corpus. **QQP**: the Quora Question Pairs2 dataset. **QNLI**: the Stanford Question Answering dataset. **SST2**: the Stanford Sentiment Treebank dataset. We also select an NLG task **E2E** dataset (Duvsek, Novikova, and Rieser 2019), which is to generates texts to evaluate a restaurant, to evaluate the quality of the model in generation tasks under privacy constraints.

Implementation Details. To standardize the training process, we partition each dataset as follows: The text classification dataset includes 50k samples for training, 1k samples for validation, and the remaining data for testing. The E2E dataset includes 42061 samples for training and 4672 samples for validation. We set different privacy constraint condi-

	Memory(GB)↓			MNLI↑			QQP↑			QNLI↑			SST2↑			Trainable param(%)
	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	
Differential Private on Adaptive Parameter Transfer Learning																
DP-LoRA	12.65	12.21	7.14	**81.49**	87.07	90.81	83.46	88.53	**91.75**	**87.32**	91.34	94.33	93.43	95.14	95.88	1.88%
DP-Adapters	13.29	13.07	7.38	80.84	86.93	90.15	84.20	87.98	91.37	86.17	90.28	94.36	92.87	95.33	95.82	1.86%
PromptDPSGD	12.44	12.12	7.25	81.16	87.13	90.48	83.58	88.46	91.22	87.23	90.87	94.11	93.13	95.29	95.93	1.92%
DP-MemArc$_{\text{side}}$	**6.18**	**6.23**	**5.66**	81.30	**87.16**	**90.91**	**84.56**	**88.92**	91.66	86.95	**91.56**	**94.40**	**93.60**	**95.44**	**95.94**	2.10%
Differential Private on Fixed Parameter Transfer Learning																
DP-Full FT	26.12	26.83	10.93	51.45	84.23	**90.65**	61.37	84.98	**92.30**	59.55	84.48	**95.13**	75.74	86.20	**96.18**	100%
DP-BiTFiT	**5.12**	5.88	4.82	75.36	83.74	89.19	78.92	85.20	90.65	83.43	87.57	93.56	89.12	93.02	95.38	0.08%
DP-MemArc$_{\text{rev}}$	5.48	**5.65**	**4.78**	**80.29**	**86.12**	90.21	**82.57**	**88.12**	91.25	**85.89**	**90.31**	94.10	**91.78**	**93.89**	95.32	3.92%

Table 2: Experiments on the RoBERTa-large model. We evaluate the accuracy(%) results and profile to compute the training memory(GB) with privacy constraints at $\epsilon = 1.6, 8, \infty$. We propose two DP-MemArc architectures as novel efficient memory privacy fine-tuning schemes. Adaptive and Fixed are used to differentiate whether the trainable parameters can be adjusted.

	Memory(GB)↓			BLEU↑			Rouge-L↑			Perplexity↓			Trainable param(%)
	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	$\epsilon = 1.6$	$\epsilon = 8$	$\epsilon = \infty$	
Differential Private on Adaptive Parameter Transfer Learning													
DP-LoRA	22.21	21.72	13.73	65.4	67.1	69.1	64.4	68.3	72.1	2.42	2.45	2.31	1.15%
DP-Adapters	23.68	24.12	14.55	65.2	66.9	**69.2**	**64.9**	68.2	71.9	2.44	2.35	2.28	1.16%
PromptDPSGD	22.12	20.96	14.18	64.2	66.5	69.1	65.0	68.3	72.0	2.60	2.54	2.39	1.33%
DP-MemArc$_{\text{side}}$	**11.68**	**11.44**	**10.17**	**66.4**	**68.2**	68.9	64.6	**68.5**	**72.7**	**2.32**	**2.38**	**2.24**	1.28%
Differential Private on Fixed Parameter Transfer Learning													
DP-Full FT	58.96	62.23	20.45	62.2	66.8	69.3	63.4	67.8	**72.6**	**2.46**	**2.23**	**1.85**	100%
DP-BiTFiT	9.59	**9.71**	8.62	61.7	65.2	68.6	62.9	66.4	71.3	2.83	2.58	2.77	0.05%
DP-MemArc$_{\text{rev}}$	**9.45**	9.88	**8.39**	**65.1**	**66.1**	**69.8**	**64.2**	**68.1**	71.6	2.71	2.65	2.58	2.15%

Table 3: Experiments on the GPT-2-large model. We evaluate the BLEU(%), Rouge-L(%) and Perplexity scores results on E2E dataset and profile to compute the training memory(GB) with privacy constraints at $\epsilon = 1.6, 8, \infty$.

tions specifically as $\epsilon = \{1.6, 8, \infty\}$ and $\delta = 1/|\mathcal{D}_{train}|$ to assess performance variations among different methods under these constraints. We chose a learning rate of $5\mathrm{e}{-4}$ and used DP-Adam optimizer as the default optimizer for the model, while DP-SGD optimizer is employed for Prompt-DPSGD. For evaluation metrics, we utilize a profiler to track the model's training memory usage, evaluating the mean memory consumption during training. Default LoRA and Adapters ranks are set to $r = 64$. For text classification tasks, we compare accuracy. For generation tasks, we employed perplexity, BLEU (Papineni et al. 2002), and ROUGE-L (Lin 2004) as evaluation metrics to comprehensively assess generation quality. In our experiments, we conduct training with a batch size of 32 and sequence length of 128 in FP16.

Experiments

Main Results

We evaluate various baseline methods on multiple task datasets and organized the results of RoBERTa and GPT2 separately according to the task type.

Text classification on RoBERTa-large. As shown in Table 2, the two DP-MemArc methods demonstrate competitive performance on text classification tasks using the RoBERTa-large model.

(1) The side network design achieves the best results compared to other adaptive baseline methods in most of the accuracy evaluations. The average performance on DP-MemArc$_{\text{side}}$ is similar to DP-LoRA, but the side network design method requires less training memory than DP-LoRA.

(2) Specifically, compared to the performance of DP-LoRA under privacy constraints, our DP-MemArc$_{\text{side}}$ achieves nearly $2 \sim 3\times$ optimization in training memory. Simultaneously, we can observe that when further memory savings during training are required, the reversible network design of DP-MemArc offers an ideal choice.

(3) Compared to the current most memory-efficient method, DP-BiTFiT, our method consistently performs better in downstream tasks while maintaining similar training memory usage. This indicates that DP-MemArc$_{\text{rev}}$ can better learn the characteristics of downstream tasks and perform gradient clipping based on computable activation function values while preserving privacy.

(4) In terms of average performance, DP-MemArc$_{\text{rev}}$ improves accuracy by an average of **+3.1%** compared to DP-BiTFiT and performs better in scenarios with higher privacy constraints ϵ, suggesting that the model better captures the gradient changes of the training data and adapts to downstream tasks.

Text Generation on GPT-2-large. For generative tasks, we employ three metrics to assess the quality of animal generation and simultaneously utilize profiles to record the changes in training memory. Experiments on Table 3 indicate that our approach demonstrates performance comparable to text

classification tasks in generative tasks.

(1) Our side network design excels in perplexity performance compared to other differential privacy parameter tuning methods. Additionally, DP-MemArc$_{\text{side}}$ shows outstanding performance on the BLEU metric. Comparing our method under differential privacy, when the parameter ϵ is set to 1.6 indicating higher privacy demands, performance in the BLEU metric only drops by 3.5%. This suggests our method better learns the characteristics and paradigms of generative tasks, yielding relatively accurate outputs.

(2) Compared to DP-BiTFiT, reversible network design exhibits competitive training memory consumption requirements, with DP-MemArc$_{\text{rev}}$ maintaining strong performance. This approach maintains relatively stable task accuracy under highly constrained training memory conditions.

(3) Compared to full differential privacy fine-tuning, DP-MemArc$_{\text{rev}}$ saves approximately $6 \sim 8\times$ the training memory in high privacy $\epsilon = 1.6$ scenarios. These results underscore the promising outlook of our proposed DP-MemArc framework for generative tasks, maintaining lower training memory requirements even at larger batch sizes.

Analysis

We conduct a deep analysis of two DP-MemArc methods and perform ablation experiments on the corresponding modules, including the differential private algorithm and alternative model setting.

Book-Keeping in DP-MemArc. In the setup of these two architectures, we use the BK-MixOpt method for differential privacy training. BK-MixOpt reduces the required training memory by using Ghostnorm to compute the normalized formulation. To evaluate the impact of different differential privacy methods during the training process, we conducted experiments on these two model designs and measured the average memory consumption during the training process. The results are shown in Table 4.

BK-MixOpt exhibits the best performance in the following scenarios. From the ablation experiments, the BK-MixOpt method reduces training memory consumption by $1.5 \sim 2\times$ in privacy-preserving computation. This highlights the importance of using BK-MixOpt within our framework. When there are no privacy constraints as $\epsilon = \infty$, all three methods degrade into the standard gradient descent process. Under the condition of privacy constraints, if the Opcaus method of calculating gradients for each sample is adopted, the time complexity for calculating the sample gradient in a single layer under the two architectures DP-MemArc$_{\text{side}}$ and DP-MemArc$_{\text{rev}}$ is $O(Bpd/64)$ and $O(4Bpr)$. This still requires a considerable amount of computation time, and in DP-MemArc$_{\text{side}}$, the gradient calculation for the upsampling and downsampling matrices also needs to be considered.

Reversible Network Functions. In the design of DP-MemArc$_{\text{rev}}$, we include two sub-functions that are used to achieve the reversible design of reversible networks. Section elaborates on the principles of the reversible network's inversion. Therefore, we can modify the internal design while ensuring that each sub-function fulfills its respective role. To further understand the differences between various designs, we fix the sub-function $\mathcal{G}$ and change the internal architecture of sub-function $\mathcal{F}$, replacing it with different parameter-efficient fine-tuning(PEFT) methods. In the privacy scenario, we select different methods and incorporate them with DP-MemArc$_{\text{rev}}$ in terms of accuracy and training memory consumption.

We have selected several classic and efficient parameter fine-tuning methods to replace the subfunction $\mathcal{F}$ here, including LoRA (Hu et al. 2021), Parallel Adapters (He et al. 2021), Prefix tuning (Li and Liang 2021) and dyLoRA (Valipour et al. 2023), and set the constraint $\epsilon = 8.0$. The result is shown in Figure 2.

LoRA is superior to other candidate architectures as a reversible network sub-function. Compared to other methods, using $\mathcal{F} = \mathbf{F}(\mathbf{W}_{\text{p}} + LR, \theta + \Delta\theta; x_i^2)$ results in a slight **+0.71** improvement in accuracy. Given the simplicity of LoRA's network architecture and the similarity in training memory usage across various methods, we finally adopt LoRA as the reversible network sub-function for DP-MemArc$_{\text{rev}}$.

	Privacy Constrains	DP-MemArc$_{\text{side}}$	DP-MemArc$_{\text{rev}}$
Opcaus	$\epsilon = 1.6, \delta = 2 \times 10^{-5}$	7.45	10.66
	$\epsilon = 8.0, \delta = 2 \times 10^{-5}$	7.33	10.98
	$\epsilon = \infty, \delta = 2 \times 10^{-5}$	5.60	4.82
GhostClip	$\epsilon = 1.6, \delta = 2 \times 10^{-5}$	9.72	8.52
	$\epsilon = 8.0, \delta = 2 \times 10^{-5}$	9.54	8.43
	$\epsilon = \infty, \delta = 2 \times 10^{-5}$	5.72	4.75
BK-MixOpt	$\epsilon = 1.6, \delta = 2 \times 10^{-5}$	6.18	5.48
	$\epsilon = 8.0, \delta = 2 \times 10^{-5}$	6.23	5.65
	$\epsilon = \infty, \delta = 2 \times 10^{-5}$	5.66	4.78

Table 4: Evaluations of Different DP methods on DP-MemArc.

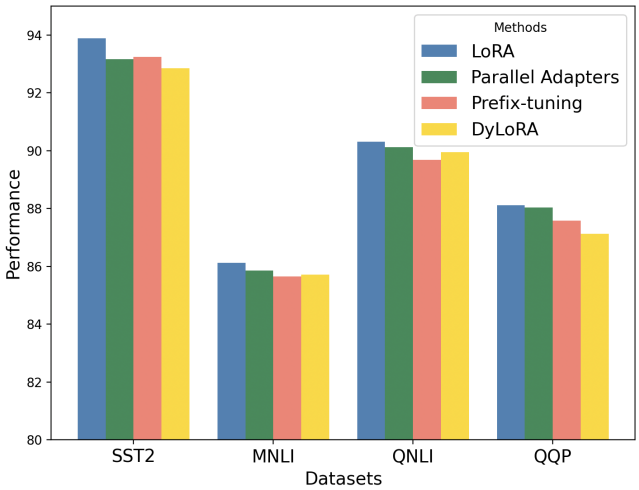


Figure 2: Performance of different reversible network sub-function $\mathcal{F}$ design. The private constraint is $\epsilon = 8.0$.

	Time Complexity(sec)	Memory Complexity(GB)	QQP scores↑
DP-Full FT	626.51	26.83	84.98
DP-LoRA	295.62	12.68	88.73
DP-Adapters	301.96	13.07	87.98
DP-BiTFiT	285.98	5.88	85.20
PromptDPSGD	293.65	12.06	88.31
DP-MemArc$_{\text{side}}$	291.56	6.23	**88.92**
DP-MemArc$_{\text{rev}}$	**284.69**	**5.65**	88.12

Table 5: Experiments on time efficiency for different methods on Roberta-large. The privacy constraints are set to $\epsilon = 8$.

Time Efficiency

To better validate the time efficiency theory in Table 1, we compare the time efficiency of our method with the baseline method to ensure that DP-MemArc maintains consistent time consumption while being memory-efficient. Since the inference speed of a model is positively correlated with the size of the training parameters in most cases, when we set the same number of trainable parameters, the impact of parameter size is reduced, and the difference in consumption is mainly reflected in the different frameworks. The corresponding results are shown in the Table 5. The experimental results indicate that compared to other baseline methods, DP-MemArc is more efficient in reasoning across various downstream tasks.

Training Scale Analysis

To understand and compare the training process and accuracy variations of different methods under differential privacy, we use checkpoints to record the training process of the model. We test three methods: DP-LoRA, DP-MemArc$_{\text{side}}$, and DP-MemArc$_{\text{rev}}$ on the GPT2-large model, evaluating their BLEU scores. From the results, DP-MemArc$_{\text{side}}$ and DP-MemArc$_{\text{rev}}$ require more training steps to reach stable values compared to DP-LoRA. Considering the architecture of the models themselves, DP-MemArc$_{\text{side}}$ needs to be tuned for the entire side network to adapt to the corresponding time for downstream tasks. Training the low-rank matrices of DP-LoRA is relatively simpler. As for the reversible network, due to the use of approximation methods for learning, more training data helps to mitigate the performance loss caused by approximation by adjusting the reversible gradients.

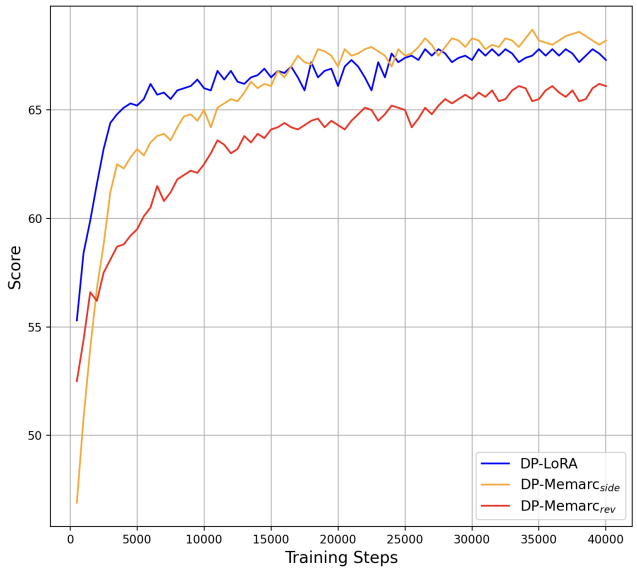


Figure 3: The experiment of training steps is conducted on the E2E dataset.

Related Work

Differential Private Fine-tuning

To ensure the privacy needs of the model, differential privacy fine-tuning methods offer a feasible solution with strong theoretical guarantees (Abadi et al. 2016; Song, Chaudhuri, and Sarwate 2013). In terms of model structure, PEFT methods can be transferred to differential privacy schemes (Yu et al. 2021; Bu et al. 2024; Xu et al. 2024). In methods design, the selected differential privacy (Shi et al. 2022a,b) approach can provide stronger differential privacy constraints more specifically for designated information. In algorithm design, it includes a series of studies (Rochette, Manoel, and Tramel 2020; Du et al. 2023) on the computational graph during the differential privacy propagation process. Techniques like Ghostnorm (Goodfellow 2015; Li et al. 2021) and Book-Keeping (Bu et al. 2023) provide unified batch norm computation and batch processing for gradient clipping. Although differential privacy offers very strong theoretical protections, reducing the memory requirements for training under differential privacy scenarios remains a significant challenge (Du et al. 2023).

Parameter Efficient Transfer Learning

To reduce the demand for computational resources during training, parameter-efficient fine-tuning methods are applied to transfer learning. Common methods include training low-rank matrices (Hu et al. 2021; Valipour et al. 2023; Dettmers et al. 2024), adding adapters (Houlsby et al. 2019; He et al. 2021), and performing prefix tuning (Li and Liang 2021; Liu et al. 2022b) or prompt tuning (Lester, Al-Rfou, and Constant 2021) on the inputs of the original model. While most parameter-efficient fine-tuning methods reduce time and space consumption, they still require significant training memory due to the state of activation functions (Sung, Cho, and Bansal 2022; Liao, Tan, and Monz 2023). Our framework offers side networks and reversible networks designs to reduce the memory required during training.

Conclusion

In this paper, we introduce a framework called DP-MemArc, which encompasses two methods aimed at addressing the issue of excessive memory consumption during training in privacy-sensitive scenarios. In this process, we reduce the training memory consumption of models in privacy environments using the BK method. With DP-MemArc, LLMs can perform downstream tasks under corresponding privacy constraints across various tasks. We hope that our method will contribute to future private efficient memory optimization for fine-tuning LLMs.

Acknowledgements

This work was partly supported by the NSFC under No. 62402418 and No. 62102360. This work was also partly supported by the Key R&D Program of Ningbo under No. 2024Z115.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 308–318.

Bang, Y.; Cahyawijaya, S.; Lee, N.; Dai, W.; Su, D.; Wilie, B.; Lovenia, H.; Ji, Z.; Yu, T.; Chung, W.; et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Bu, Z.; Mao, J.; and Xu, S. 2022. Scalable and efficient training of large convolutional neural networks with differential privacy. *Advances in Neural Information Processing Systems*, 35: 38305–38318.

Bu, Z.; Wang, Y.-X.; Zha, S.; and Karypis, G. 2023. Differentially private optimization on large model at small cost. In *International Conference on Machine Learning*, 3192–3218. PMLR.

Bu, Z.; Wang, Y.-X.; Zha, S.; and Karypis, G. 2024. Differentially private bias-term only fine-tuning of foundation models. In *International Conference on Machine Learning*. PMLR.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Du, M.; Yue, X.; Chow, S. S.; Wang, T.; Huang, C.; and Sun, H. 2023. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2665–2679.

Duan, H.; Dziedzic, A.; Papernot, N.; and Boenisch, F. 2024. Flocks of stochastic parrots: Differentially private prompt learning for large language models. *Advances in Neural Information Processing Systems*, 36.

Duvsek, O.; Novikova, J.; and Rieser, V. 2019. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computational Linguistics*, 1(1).

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, 265–284. Springer.

Finlayson, M.; Swayamdipta, S.; and Ren, X. 2024. Logits of API-Protected LLMs Leak Proprietary Information. *arXiv preprint arXiv:2403.09539*.

Gomez, A. N.; Ren, M.; Urtasun, R.; and Grosse, R. B. 2017. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30.

Goodfellow, I. 2015. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*.

He, J.; Zhou, C.; Ma, X.; Berg-Kirkpatrick, T.; and Neubig, G. 2021. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*.

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

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, 2790–2799. PMLR.

Hu, E. J.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W.; et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Ke, S.; Hou, C.; Fanti, G.; and Oh, S. 2024. On the Convergence of Differentially-Private Fine-tuning: To Linearly Probe or to Fully Fine-tune? *arXiv preprint arXiv:2402.18905*.

Koskela, A.; Jälkö, J.; and Honkela, A. 2020. Computing tight differential privacy guarantees using fft. In *International Conference on Artificial Intelligence and Statistics*, 2560–2569. PMLR.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3045–3059.

Li, J.; Zhang, Z.; and Zhao, H. 2022. Self-prompting large language models for open-domain qa. *arXiv preprint arXiv:2212.08635*.

Li, X.; Liu, D.; Hashimoto, T. B.; Inan, H. A.; Kulkarni, J.; Lee, Y.-T.; and Guha Thakurta, A. 2022. When does differentially private learning not suffer in high dimensions? *Advances in Neural Information Processing Systems*, 35: 28616–28630.

Li, X.; Tramer, F.; Liang, P.; and Hashimoto, T. 2021. Large Language Models Can Be Strong Differentially Private Learners. In *International Conference on Learning Representations*.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4582–4597.

Liao, B.; Tan, S.; and Monz, C. 2023. Make Pre-trained Model Reversible: From Parameter to Memory Efficient Fine-Tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Liu, H.; Tam, D.; Muqeeth, M.; Mohta, J.; Huang, T.; Bansal, M.; and Raffel, C. A. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35: 1950–1965.

Liu, X.; Ji, K.; Fu, Y.; Tam, W.; Du, Z.; Yang, Z.; and Tang, J. 2022b. P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 61–68.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

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

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Robinson, J.; Rytting, C. M.; and Wingate, D. 2022. Leveraging large language models for multiple choice question answering. *arXiv preprint arXiv:2210.12353*.

Rochette, G.; Manoel, A.; and Tramel, E. W. 2020. Efficient Per-Example Gradient Computations in Convolutional Neural Networks. In *Workshop on Theory and Practice of Differential Privacy (TPDP)*.

Shi, W.; Cui, A.; Li, E.; Jia, R.; and Yu, Z. 2022a. Selective Differential Privacy for Language Modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2848–2859.

Shi, W.; Shea, R.; Chen, S.; Zhang, C.; Jia, R.; and Yu, Z. 2022b. Just Fine-tune Twice: Selective Differential Privacy for Large Language Models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 6327–6340.

Song, S.; Chaudhuri, K.; and Sarwate, A. D. 2013. Stochastic gradient descent with differentially private updates. In *2013 IEEE global conference on signal and information processing*, 245–248. IEEE.

Sung, Y.-L.; Cho, J.; and Bansal, M. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35: 12991–13005.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Valipour, M.; Rezagholizadeh, M.; Kobyzev, I.; and Ghodsi, A. 2023. DyLoRA: Parameter-Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low-Rank Adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 3274–3287.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*.

Wang, S.; Nguyen, J.; Li, K.; and Wu, C.-J. 2023. READ: Recurrent Adaptation of Large Transformers. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Xu, J.; Saravanan, K.; van Dalen, R.; Mehmood, H.; Tuckey, D.; and Ozay, M. 2024. DP-DyLoRA: Fine-Tuning Transformer-Based Models On-Device under Differentially Private Federated Learning using Dynamic Low-Rank Adaptation. *arXiv preprint arXiv:2405.06368*.

Yousefpour, A.; Shilov, I.; Sablayrolles, A.; Testuggine, D.; Prasad, K.; Malek, M.; Nguyen, J.; Ghosh, S.; Bharadwaj, A.; Zhao, J.; et al. 2021. Opacus: User-Friendly Differential Privacy Library in PyTorch. In *NeurIPS 2021 Workshop Privacy in Machine Learning*.

Yu, D.; Naik, S.; Backurs, A.; Gopi, S.; Inan, H. A.; Kamath, G.; Kulkarni, J.; Lee, Y. T.; Manoel, A.; Wutschitz, L.; et al. 2021. Differentially Private Fine-tuning of Language Models. In *International Conference on Learning Representations*.

Yu, L.; Liu, L.; Pu, C.; Gursoy, M. E.; and Truex, S. 2019. Differentially private model publishing for deep learning. In *2019 IEEE symposium on security and privacy (SP)*, 332–349. IEEE.

Zaken, E. B.; Goldberg, Y.; and Ravfogel, S. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 1–9.

Zhang, J. O.; Sax, A.; Zamir, A.; Guibas, L.; and Malik, J. 2020. Side-tuning: a baseline for network adaptation via additive side networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 698–714. Springer.