# MemDPT: Differential Privacy for Memory Efficient Language Models

**Anonymous ACL submission**

## Abstract

Large language models have consistently demonstrated remarkable performance across a wide spectrum of applications. Nonetheless, the deployment of these models can inadvertently expose user privacy to potential risks. The substantial memory demands of these models during training represent a significant resource consumption challenge. The sheer size of these models imposes a considerable burden on memory resources, which is a matter of significant concern in practice. In this paper, we present an innovative training framework MemDPT that not only reduces the memory cost of large language models but also places a strong emphasis on safeguarding user data privacy. MemDPT provides edge network and reverse network designs to accommodate various differential privacy memory-efficient fine-tuning schemes. Our approach not only achieves $2 \sim 3\times$ memory optimization but also provides robust privacy protection, ensuring that user data remains secure and confidential. Extensive experiments have demonstrated that MemDPT can effectively provide differential privacy efficient fine-tuning across various task scenarios.

## 1 Introduction

Large language models (LLMs) (Radford et al., 2019; Hoffmann et al., 2022a; 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 et al., 2022; Li et al., 2022a). 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 et al., 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., 2022b; 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 **Mem**ory efficient **D**ifferential **P**rivate **T**uning (**MemDPT**), a framework for training in scenarios that involve both privacy-protection and memory efficient transfer learning. In our framework, we explore two efficient methods for parameter-efficient fine-tuning, $\text{MemDPT}_{\text{side}}$ and $\text{MemDPT}_{\text{rev}}$, 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 MemDPT, 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 MemDPT framework on multiple datasets and models. The results show promising performance across various dimensions, with a substantial improvement in training memory.

## 2 Preliminaries

### 2.1 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:

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

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

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

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 $O(BT(p + d) + pd)$.

### 2.2 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 $(\epsilon, \delta)$-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

| 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.

optimizers such as SGD or Adam (Abadi et al., 2016; Mironov, 2017; Koskela et al., 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 et al., 2022) strategy, the gradient norm of each sample is calculated.

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

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 $\mathbf{G}_l = \boldsymbol{x}_{(l)}^\top \mathrm{diag}(\mathbf{C}) \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{(l)}}$.

## 3 Methodology

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

### 3.1 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 $\mathbf{W}_{\mathrm{p}}$, $x_0$, $y$, $\theta$. The model could formulated as:

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

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:

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

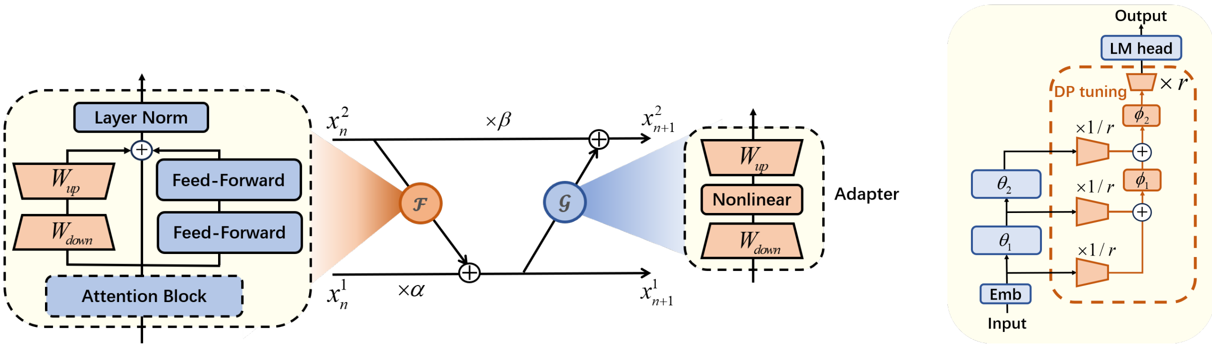
We hope to find a form that remains distinct from the original form when adjusting parameters. That is, there exists such a form:

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

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 edge networks, we adopt the LST (Sung et al., 2022) method. This involves passing the intermediate layer information from the pre-trained model to the edge network through a linear layer $f'$. We denote this method as $\mathrm{MemDPT}_{\mathrm{side}}$.




(a) Designs of MemDPT$_{\text{rev}}$.

(b) Designs of MemDPT$_{\text{side}}$.

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

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

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

### 3.2 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. MemDPT$_{\text{side}}$ 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 et al., 2023), the following form is usually satisfied.

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

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 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 MemDPT$_{\text{rev}}$.

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:

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

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

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.

| | Memory(GB)↓ | | | MNLI↑ | | | QQP↑ | | | QNLI↑ | | | SST2↑ | | | Trainable param(%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\epsilon=1.6$ | $\epsilon=8$ | $\epsilon=\infty$ | $\epsilon=1.6$ | $\epsilon=8$ | $\epsilon=\infty$ | $\epsilon=1.6$ | $\epsilon=8$ | $\epsilon=\infty$ | $\epsilon=1.6$ | $\epsilon=8$ | $\epsilon=\infty$ | $\epsilon=1.6$ | $\epsilon=8$ | $\epsilon=\infty$ | |
| 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-LoRA | 12.68 | 12.24 | 7.12 | **82.89** | **88.28** | 90.83 | 83.85 | 88.73 | **91.95** | **87.51** | 91.38 | **94.76** | 93.58 | 95.20 | **96.25** | 3.82% |
| 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% |
| 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% |
| PromptDPSGD | 12.58 | 12.06 | 7.21 | 81.33 | 87.02 | 90.68 | 83.58 | 88.31 | 91.19 | 87.15 | 90.89 | 94.20 | 93.12 | 95.24 | 95.97 | 0.96% |
| $\text{MemDPT}_{\text{side}}$ | 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% |
| $\text{MemDPT}_{\text{rev}}$ | 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 MemDPT architectures as novel efficient memory privacy fine-tuning schemes.

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. As shown in Table 1, the first two parts mainly rely on the pre-trained model and the size of the additional parameters, while the Book-keeping strategy requires to store training memory of $O(4BT^2)$. Here, we also employ the BK algorithm to calculate the norm of the samples, thereby obtaining the corresponding gradient values. During training, we set batch sizes to 32. When dealing with tasks involving input sequence lengths of 128, which are medium sequence lengths of problems, MemDPT significantly reduces the memory required for training due to $pd \gg T^2$.

## 4 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 et al., 2022), and PromptDPSGD (Duan et al., 2024; Lester et al., 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 et al., 2019), which is to generates texts to evaluate a restaurant, to evaluate the quality of the model in generation tasks under privacy constraints. More details about the dataset are in Appendix A.

**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 conditions 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 5e-4 and used DP-Adam optimizer as the default optimizer for the model, while DP-SGD optimizer is employed for PromptDPSGD. 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.

| | Memory(GB)↓ | | | BLEU↑ | | | Rouge-L↑ | | | Perplexity↓ | | | Trainable param(%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\epsilon = 1.6$ | $\epsilon = 8$ | $\epsilon = \infty$ | $\epsilon = 1.6$ | $\epsilon = 8$ | $\epsilon = \infty$ | $\epsilon = 1.6$ | $\epsilon = 8$ | $\epsilon = \infty$ | $\epsilon = 1.6$ | $\epsilon = 8$ | $\epsilon = \infty$ | |
| 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-LoRA | 22.38 | 21.75 | 13.68 | 65.8 | 67.3 | 69.5 | 64.8 | **69.1** | 72.4 | 2.39 | 2.48 | 2.32 | 2.30% |
| DP-Adapters | 23.68 | 24.12 | 14.55 | 65.2 | 66.9 | **69.8** | **65.1** | 68.5 | 71.9 | 2.44 | 2.35 | 2.28 | 1.16% |
| 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% |
| 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 | 0.67% |
| $\text{MemDPT}_{\text{side}}$ | 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% |
| $\text{MemDPT}_{\text{rev}}$ | **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 $\epsilon = 1.6, 8, \infty$.

# 5 Experiments

## 5.1 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 MemDPT methods demonstrate competitive performance on text classification tasks using the RoBERTa-large model.

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

(2) Specifically, compared to the performance of DP-LoRA under privacy constraints, our $\text{MemDPT}_{\text{side}}$ 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 MemDPT 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 $\text{MemDPT}_{\text{rev}}$ 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, $\text{MemDPT}_{\text{rev}}$ improves accuracy by an average of **+3.1%** compared to DP-BiTFiT and performs better in scenarios with higher privacy constraints $\epsilon$, 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 edge network design excels in perplexity performance compared to other differential privacy parameter tuning methods. Additionally, $\text{MemDPT}_{\text{side}}$ shows outstanding performance on the BLEU metric. Comparing our method under differential privacy, when the parameter $\epsilon$ 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 $\text{MemDPT}_{\text{rev}}$ maintaining strong performance. This approach maintains relatively stable task accuracy under highly constrained training memory conditions.

(3) Compared to full differential privacy fine-tuning, $\text{MemDPT}_{\text{rev}}$ 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 MemDPT framework for generative tasks, maintaining lower training memory requirements even at larger batch sizes.

## 5.2 Analysis

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

**Book-Keeping in MemDPT.** In the setup of these two architectures, we use the BK method for differential privacy training. BK reduces the required training memory by using Ghostnorm to compute

| | Privacy Constrains | $\text{MemDPT}_{\text{side}}$ | $\text{MemDPT}_{\text{rev}}$ |
|---|---|---|---|
| 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 |
| Book-Keeping | $\epsilon = 1.6, \delta = 2 \times 10^{-5}$ | 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 MemDPT.

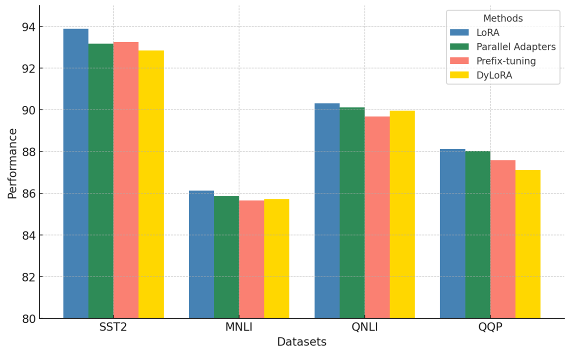



Figure 2: Performance of different reversible network sub-function $\mathcal{F}$ design. The private constraint is $\epsilon = 8.0$.

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 exhibits the best performance in the following scenarios. From the ablation experiments, the BK method reduces training memory consumption by $1.5 \sim 2\times$ in privacy-preserving computation. This highlights the importance of using BK 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 $\text{MemDPT}_{\text{side}}$ and $\text{MemDPT}_{\text{rev}}$ is $O(Bpd/64)$ and $O(4Bpr)$. This still requires a considerable amount of computation time, and in $\text{MemDPT}_{\text{side}}$, the gradient calculation for the upsampling and downsampling matrices also needs to be considered. Meanwhile, we can observe that when $r$ is relatively small, the Opcaus method requires relatively less computational memory. To ensure the overall model's accuracy in downstream tasks, the BK method remains the most efficient choice.

**Reversible Network Functions.** In the design of $\text{MemDPT}_{\text{rev}}$, we include two sub-functions that are used to achieve the reversible design of reversible networks. Section 3.2 elaborates on the principles of the reversible network's inversion. This scheme leverages the similarity processing of learnable parameters in the initial setup. 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. These PEFT methods have been widely used in non-privacy scenarios. In the privacy scenario, we select different methods and incorporate them with $\text{MemDPT}_{\text{rev}}$ in terms of accuracy and training memory consumption.

We have selected several classic and efficient parameter fine-tuning methods to replace the sub-function $\mathcal{F}$ 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 $\text{MemDPT}_{\text{rev}}$.

### 5.3 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, $\text{MemDPT}_{\text{side}}$, and $\text{MemDPT}_{\text{rev}}$ on the GPT2-large model, evaluating their BLEU scores. During training, each batch size is set to 32, and the model is trained for 40K steps, observing the performance and changes in BLEU scores. The privacy parameters of the model are set to $\epsilon = 8$ and $\delta = \frac{1}{42061}$. The result is shown in Figure 3.

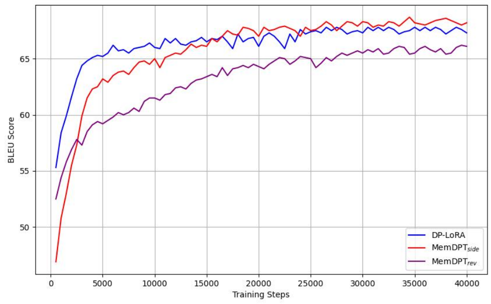



Figure 3: The experiment is conducted on the E2E dataset. The BLEU scores of different methods are based on the number of training steps of the model.

From the results, $\text{MemDPT}_{\text{side}}$ and $\text{MemDPT}_{\text{rev}}$ require more training steps to reach stable values compared to DP-LoRA. Considering the architecture of the models themselves, $\text{MemDPT}_{\text{side}}$ 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. Additionally, since we employ differential privacy methods for training, although the BLEU scores during training fluctuate, they remain relatively balanced, which aligns with our expectations for using differential privacy.

# 6 Related Work

## 6.1 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 et al., 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 et al., 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). MemDPT employs efficient and memory-friendly designs at both the model and algorithm levels, thereby reducing training memory requirements while maintaining original performance.

## 6.2 Parameter Efficient Transfer Learning

Training and inference for a large language model require substantial computational resources, which are often limited in many scenarios (Hoffmann et al., 2022b). To reduce the demand for computational resources during training, parameter-efficient fine-tuning methods are applied to transfer learning. This approach involves fine-tuning a small subset of new parameters and integrating them into the model for plug-and-play inference. 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 et al., 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 et al., 2022; Liao et al., 2023). Our framework offers two methods, edge networks, and inverse networks, to reduce the memory required during training.

# 7 Conclusion

In this paper, we introduce a framework called MemDPT, 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 the design of MemDPT, language models can perform downstream tasks under corresponding privacy constraints across various tasks. Multiple experiments have demonstrated the effectiveness of our approach, achieving significant optimization in training memory. We hope that our method will contribute to future private efficient memory optimization for fine-tuning large language models and be applicable to different training tasks.

## Limitation

Our approach offers a solution to the efficient memory tuning problem in differential privacy training, alleviating the issue of insufficient training memory in privacy scenarios. However, our method also has certain limitations. Firstly, for longer context texts, since the BK algorithm relies on the context length for its time complexity, the memory optimization might be inadequate. Additionally, due to the limitations of large language models, private fine-tuning may result in hallucination issues due to the inherent knowledge deficiencies of the language model, leading to diminished effectiveness. Secondly, during batch training, even the forward pass already occupies a significant amount of memory, and current open-source large language models (such as the Llama series) still require a substantial amount of training memory for batch training. We can consider using distributed training methods to address this issue and reduce training memory requirements. Moreover, our privacy protection scenario targets the differential privacy fine-tuning of all content in the training data. In certain specific scenarios, we may only need to protect certain entities or fields. In the future, we will explore solutions for partial information privacy protection and attempt to apply our framework to these types of problems.

## Ethics Statement

The data and code we use are all sourced from public information, and our training data does not contain any specific personal or organizational information or privacy. Our method provides a privacy-protecting training approach that can help entities or organizations prevent the leakage of sensitive information during model training. The information and content we use comply with relevant open-source protocols and licenses.

## References


Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

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

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2023. Differentially private optimization on large model at small cost. In *International Conference on Machine Learning*, pages 3192–3218. PMLR.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. 2024. Differentially private bias-term only fine-tuning of foundation models. In *International Conference on Machine Learning*. PMLR.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Minxin Du, Xiang Yue, Sherman SM Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. 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*, pages 2665–2679.

Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. 2024. Flocks of stochastic parrots: Differentially private prompt learning for large language models. *Advances in Neural Information Processing Systems*, 36.

Ondrej Duvsek, Jekaterina Novikova, and Verena Rieser. 2019. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *Computational Linguistics*, 1(1).

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 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*, pages 265–284. Springer.

Matthew Finlayson, Swabha Swayamdipta, and Xiang Ren. 2024. Logits of api-protected llms leak proprietary information. *arXiv preprint arXiv:2403.09539*.

Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. 2017. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30.

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

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022a. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022b. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35:30016–30030.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Shuqi Ke, Charlie Hou, Giulia Fanti, and Sewoong Oh. 2024. On the convergence of differentially-private fine-tuning: To linearly probe or to fully fine-tune? *arXiv preprint arXiv:2402.18905*.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Junlong Li, Zhuosheng Zhang, and Hai Zhao. 2022a. Self-prompting large language models for open-domain qa. *arXiv preprint arXiv:2212.08635*.

Xiang Lisa Li and Percy Liang. 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)*, pages 4582–4597.

Xuechen Li, Daogao Liu, Tatsunori B Hashimoto, Huseyin A Inan, Janardhan Kulkarni, Yin-Tat Lee, and Abhradeep Guha Thakurta. 2022b. When does differentially private learning not suffer in high dimensions? *Advances in Neural Information Processing Systems*, 35:28616–28630.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*.

Baohao Liao, Shaomu Tan, and Christof Monz. 2023. Make pre-trained model reversible: From parameter to memory efficient fine-tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 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)*, pages 61–68.

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

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

Jekaterina Novikova, Ondvre Duvsek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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

Joshua Robinson, Christopher Michael Rytting, and David Wingate. 2022. Leveraging large language models for multiple choice question answering. *arXiv preprint arXiv:2210.12353*.

Gaspar Rochette, Andre Manoel, and Eric W Tramel. 2020. Efficient per-example gradient computations in convolutional neural networks. In *Workshop on Theory and Practice of Differential Privacy (TPDP)*.

Weiyan Shi, Aiqi Cui, Evan Li, Ruoxi Jia, and Zhou Yu. 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*, pages 2848–2859.

Weiyan Shi, Ryan Shea, Si Chen, Chiyuan Zhang, Ruoxi Jia, and Zhou Yu. 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*, pages 6327–6340.

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

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 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*, pages 3274–3287.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Sid Wang, John Nguyen, Ke Li, and Carole-Jean Wu. 2023. Read: Recurrent adaptation of large transformers. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

Jie Xu, Karthikeyan Saravanan, Rogier van Dalen, Haaris Mehmood, David Tuckey, and Mete Ozay. 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*.

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

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. 2021. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*.

Lei Yu, Ling Liu, Calton Pu, Mehmet Emre Gursoy, and Stacey Truex. 2019. Differentially private model publishing for deep learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 332–349. IEEE.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 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)*, pages 1–9.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 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*, pages 698–714. Springer.

# A Details of Datasets

**GLUE benchmarks.** The General Language Understanding Evaluation (GLUE) benchmark(Wang et al., 2018) represents a comprehensive suite of natural language understanding tasks aimed at advancing the field of machine learning in linguistic applications. We use the following datasets selected from GLUE:

- **MNLI Datasets:** The Multi-Genre Natural Language Inference Corpus is a crowd-sourced collection of sentence pairs with textual entailment annotations. It contains 392K samples of the tasks. It involves predicting whether a premise sentence entails, contradicts, or neither affects a hypothesis sentence. These entailment predictions are categorized as entailment, contradiction, or neutral. The premise sentences are collected from ten different sources, such as transcribed speech, fiction, and government reports.

- **QQP Datasets:** The Quora Question Pairs dataset is a collection of question pairs from the community question-answering website Quora, which has 364k samples. The task associated with this dataset is to determine whether a pair of questions are semantically equivalent.

- **QNLI Datasets:** The Stanford Question Answering Dataset is a question-answering dataset consisting of question-paragraph pairs, where one of the sentences in the paragraph, drawn from Wikipedia, contains the answer to the corresponding question written by an annotator. The task is converted into sentence pair classification by forming a pair between each question and each sentence in the corresponding context and filtering out pairs with low lexical overlap between the question and the context sentence. The task is to determine whether the context sentence contains the answer to the question. This modified version of the original task removes the requirement that the model select the exact answer and the simplifying assumptions that the answer is always present in the input and that lexical overlap is a reliable cue. This process of recasting existing datasets into NLI is similar to methods introduced and expanded upon. The converted dataset is called QNLI (Question-answering NLI), which has 104k samples.

- **SST2 Datasets:** The Stanford Sentiment Treebank consists of sentences from movie reviews and human annotations of their sentiment. The task involves predicting the sentiment of a given sentence which includes 67k samples.

**E2E benchmarks.** The E2E dataset(Novikova et al., 2017) is a valuable resource for training end-to-end, data-driven natural language generation (NLG) systems in the restaurant domain. It contains template-like information in the restaurant domain, which is used for mapping to natural language through end-to-end training. The dataset consists of 42061 training samples, 4672 validation samples, and 4693 test samples.

## B More Details on Implementation

In the experiment, we conduct experiments under three privacy constraints: $\{1.6, 8, \infty\}$. Since in the reverse network model $\text{MemDPT}_{\text{rev}}$, the activation function obtains tensor $\boldsymbol{x}$ through reverse computation during backpropagation, we need to replace the part of the gradient calculation code that originally calls the intermediate state $\boldsymbol{x}$ with the calculation formula `grad_rev()` of the reverse network. Additionally, for the edge model network $\text{MemDPT}_{\text{side}}$, we select part of the pre-trained model in each layer as the initialization parameters for the edge network. This initialization approach enhances the performance of the edge network in downstream tasks. The privacy parameter settings $\delta = 1/|D_{train}|$ are same from other works(Bu et al., 2024; Yu et al., 2021). When calculating BK, the required intermediate state information is also obtained through the calculation formula `grad_rev()` . We iteratively use a single storage space to retain the intermediate state of the calculation. Thus, when the number of layers in the LM is $L$, the training memory optimizes from $L \times O(BTd)$ to $1 \times O(BTd)$. We conduct training with a batch size of 32 and sequence length of 128 in FP16.