Differentially Private Subspace Fine-Tuning for Large Language Models

Lele Zheng[1], Xiang Wang[1], Tao Zhang[1]*, Yang Cao[2], Ke Cheng[1], Yulong Shen[1]

[1]School of Computer Science and Technology, Xidian University, Xi'an, China
[2]Department of Computer Science, Institute of Science Tokyo, Tokyo, Japan
{zhenglele, taozhang, chengke}@xidian.edu.cn, wxhygge@stu.xidian.edu.cn,
cao@c.titech.ac.jp, ylshen@mail.xidian.edu.cn

Abstract

Fine-tuning large language models on downstream tasks is crucial for realizing their cross-domain potential but often relies on sensitive data, raising privacy concerns. Differential privacy (DP) offers rigorous privacy guarantees and has been widely adopted in fine-tuning; however, naively injecting noise across the high-dimensional parameter space creates perturbations with large norms, degrading performance and destabilizing training. To address this issue, we propose DP-SFT, a two-stage subspace fine-tuning method that substantially reduces noise magnitude while preserving formal DP guarantees. Our intuition is that, during fine-tuning, significant parameter updates lie within a low-dimensional, task-specific subspace, while other directions change minimally. Hence, we only inject DP noise into this subspace to protect privacy without perturbing irrelevant parameters. In phase one, we identify the subspace by analyzing principal gradient directions to capture task-specific update signals. In phase two, we project full gradients onto this subspace, add DP noise, and map the perturbed gradients back to the original parameter space for model updates, markedly lowering noise impact. Experiments on multiple datasets demonstrate that DP-SFT enhances accuracy and stability under rigorous DP constraints, accelerates convergence, and achieves substantial gains over DP fine-tuning baselines.

Code — https://github.com/XidianNss/DP-SFT

Introduction

Large language models (LLMs), such as BERT and GPT, have achieved remarkable success in natural language processing and generation tasks (Devlin et al. 2019; Zhao et al. 2023; Alipour, Pendar, and Roy 2024). To optimize their performance on specific downstream tasks, fine-tuning is essential (Wang et al. 2025; Hu et al. 2021; Ding et al. 2023). As shown in Fig. 1, this process allows pre-trained models to be adapted to task-specific data, enhancing accuracy and effectiveness. However, fine-tuning often relies on datasets that contain sensitive information, such as personal identities, financial data, or private conversations. The potential leakage of such information poses serious threats to user privacy and security (Das, Amini, and Wu 2025). Therefore, balancing privacy protection with the performance of fine-tuning has become a critical challenge.

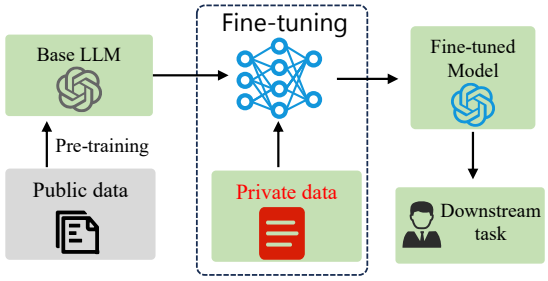

Figure 1: Fine-tuning Process of Large Language Models

Differential privacy (DP) (Dwork et al. 2006) provides strict privacy guarantees for sensitive data. The core principle of DP is to ensure that an algorithm's output remains consistent despite small changes in a single data sample. In other words, minor adjustments to an individual's data should not significantly alter the algorithm's results, thus ensuring the protection of user privacy. DP achieves this by introducing noise into the model, making it impossible for external observers to infer any individual data from the model's output. Recently, DP has become widely applied to large model training and fine-tuning, particularly in applications involving sensitive data. (Abadi et al. 2016) introduced DPSGD, which incorporates Gaussian noise into gradients during deep learning, laying the theoretical foundation for privacy protection in the fine-tuning of large language models. Building on this, (Li et al. 2024) proposed AnaDP, a method that dynamically allocates noise and privacy budgets based on the importance of model parameters during specific training steps. Their research demonstrated that AnaDP performs exceptionally well across multiple datasets, narrowing the performance gap between DP fine-tuning and standard fine-tuning while maintaining privacy.

Despite significant progress in the application of differential privacy to fine-tuning, existing methods still face the challenge of the curse of dimensionality. To ensure privacy protection, noise must typically be injected into the entire high-dimensional parameter space. This approach increases

*Corresponding author.

the noise magnitude substantially, leading to unnecessary performance degradation and training instability, which ultimately affects the model's effectiveness. For instance, given a fixed privacy parameter, the variance of the Gaussian noise added to the gradients increases linearly with the parameter dimensionality. The introduction of high-dimensional noise presents two major challenges. First, the randomness of the noise significantly disturbs each gradient update direction, making the training process more unpredictable and reducing the model's final performance. Second, the added noise negatively impacts the stability of training results. In the absence of noise, the model typically converges gradually over several iterations. However, with the introduction of high-dimensional noise, the training results often exhibit large fluctuations, making it difficult for the model to converge quickly. Therefore, achieving differential privacy protection in large language model fine-tuning without sacrificing model performance remains a significant challenge.

To address the challenges mentioned above, we draw inspiration from subspace differential privacy and propose a DP-SFT mechanism based on gradient subspace optimization. Our approach is founded on the observation that, despite the vast number of parameters in large language models, effective gradient updates during fine-tuning rely heavily on a few dominant gradient directions. These directions form a compact low-dimensional subspace that is significantly smaller than the original parameter space. Consequently, our method first constructs a gradient subspace by extracting the trajectory of full parameter fine-tuning. It then projects the training gradients onto this subspace, adds noise to meet the differential privacy requirements, and finally projects the noisy subspace back into the high-dimensional space to complete the gradient update. Intuitively, DP-SFT reduces the required noise by significantly lowering the dimensionality of the subspace (by several orders of magnitude). Specifically, our contributions are as follows:

1. We propose Differentially Private Subspace Fine-Tuning (DP-SFT), a novel two-stage framework that injects privacy noise only into a task-specific low-dimensional subspace of gradients. This approach significantly reduces noise magnitude while preserving formal DP guarantees, overcoming the dimensionality challenge in DP optimization for LLMs.

2. We demonstrate subspace transferability across tasks, enabling the subspace to be constructed on public or related datasets without consuming the privacy budget. Our analysis and experiments show that the subspace captures general task geometry rather than private sample-specific information, making DP-SFT practical and privacy-efficient.

3. Extensive experiments demonstrate that DP-SFT achieves near-non-private performance and consistently outperforms strong baselines under different privacy budgets. It sets a new state-of-the-art in privacy-utility trade-offs for differentially private fine-tuning of LLMs.

Related Works

A straightforward approach to ensuring differential privacy during fine-tuning is to inject noise into gradients, as demonstrated in DP-SGD (Abadi et al. 2016) and DP-Adam (Kingma and Ba 2014). However, due to the curse of dimensionality, directly perturbing gradients in large language models (LLMs) often leads to significant accuracy degradation. To alleviate this issue, (Yu et al. 2021c) proposes a reparameterized gradient perturbation, which reparameterizes each high-rank weight matrix into two low-rank matrices along with a residual matrix. Noise is then added only to the gradients of the low-rank components, and the perturbed gradients are projected back to update the original high-rank weights. However, applying such reparameterization at every update step can introduce training instability.

More recent work (Yu et al. 2021a) leverages parameter-efficient fine-tuning techniques, such as LoRA (Hu et al. 2021), Adapter (Houlsby et al. 2019), and Compacter (Karimi Mahabadi, Henderson, and Ruder 2021), to reduce the number of perturbed parameters. These methods inject noise only into the gradients of additional lightweight plug-in modules, rather than the full model. While this improves efficiency, (Li et al. 2021) argues that full-parameter fine-tuning remains essential in many domains. They introduce ghost clipping, a reparameterization-based technique that avoids explicitly instantiating per-example gradients. Nevertheless, existing low-rank reparameterization methods (Bondarenko, Del Chiaro, and Nagel 2024; Zhao et al. 2024) still require a large number of retained dimensions to maintain accuracy. For example, even after reparameterization, fine-tuning LLaMA-7B still involves updating hundreds of millions of parameters (Bondarenko, Del Chiaro, and Nagel 2024).

In parallel, several approaches in machine learning have been proposed to address the high-dimensional noise problem caused by differential privacy. Some methods (Kairouz et al. 2021; Bie 2024) leverage prior knowledge extracted from public datasets to improve training on private datasets, but they rely on a high degree of similarity between the public and private domains. Other works (Singhal and Steinke 2021; Tsfadia 2024) suggest using principal component analysis (PCA) to construct gradient subspaces, thereby reducing noise dimensionality. However, the high computational cost of PCA makes these methods difficult to scale. In contrast, our approach directly extracts the gradient subspace from training dynamics in a computationally efficient manner, enabling accuracy improvements with manageable resource consumption.

Preliminaries

Differential Privacy

Differential Privacy (DP) (Dwork et al. 2006) enables the analysis of population-level characteristics in a dataset without disclosing information about any individual. This is achieved by adding carefully calibrated noise to statistical queries or to the dataset itself, making it infeasible for an adversary to determine whether a particular individual's data

is present. Formally, we present the relevant concepts of differential privacy and its mathematical definition.

Definition 1 ((ε, δ)-Differential Privacy)**.** *For a given $\varepsilon \in \mathbf{R}_{\geq 0}$, an obfuscation mechanism $\mathcal{M}$ satisfies (ε, δ)-DP if and only if for any pair of neighboring datasets D, D', and any output set $S \in \mathcal{S}$ ($\mathcal{S}$ is the set of all possible outputs), the probability that outputs belong to the same set should be bounded by*

$$\Pr(\mathcal{M}(D) \subseteq S) \leq e^{\varepsilon} \Pr(\mathcal{M}(D') \subseteq S) + \delta, \qquad (1)$$

where ε represents the privacy budget, which is used to measure the degree of privacy protection. A smaller ε indicates stronger privacy protection but also requires adding more noise. δ represents the relaxation term, which measures the degree of non-satisfaction of differential privacy to some extent.

In practical applications, δ is usually set to $o\left(\frac{1}{n}\right)$ to ensure the overall protection effect of differential privacy, where n denotes the number of individuals.

Differential privacy exhibits two fundamental and advantageous properties: composition and post-processing invariance. The composition property quantifies the cumulative privacy loss incurred when multiple analyses are performed on the same dataset, enabling rigorous tracking of privacy budgets over time. In contrast, post-processing invariance guarantees that any data-independent computation applied to the output of a differentially private mechanism cannot weaken its privacy guarantees.

Theorem 1 (Composition)**.** *Let mechanisms M_i ($i \in \{1, 2, \ldots, T\}$) satisfy $(\varepsilon_i, \delta_i)$-DP, then their combination $(M_1, M_2, \ldots, M_T)$ satisfies $(\sum_{i=1}^{T} \varepsilon_i, \sum_{i=1}^{T} \delta_i)$-DP.*

Theorem 2 (Post-processing invariance)**.** *Given a deterministic function f and a (ε, δ)-differentially private mechanism $\mathcal{M}$, then the composite mechanism $f \circ \mathcal{M}$ satisfies (ε, δ)-DP.*

Gaussian Mechanism in Deep Learning

In deep learning, the Gaussian mechanism (Liu 2018) is widely used to implement differential privacy by adding calibrated noise to gradients during training. Specifically, each sample's gradient g is first clipped to a predefined norm bound C to control sensitivity. Gaussian noise with variance determined by C, ε, and δ is then added to the clipped gradients. The noisy gradients are used to update model parameters, enabling privacy-preserving optimization with a balanced trade-off between privacy and utility.

Gopi et al.'s research (Gopi, Lee, and Wutschitz 2021) utilizes Gaussian differential privacy (Balle and Wang 2018) to provide an effective method for calculating the Gaussian noise that should be added in deep learning. We abbreviate this method as $\mathrm{gdp}(\cdot)$, with the formal usage as follows:

Corollary 1 (Property of Gaussian noise)**.** *Given privacy budget ε, relaxation term δ, sampling probability q and training steps T, we obtain*

$$\sigma = \mathrm{gdp}(\varepsilon, \delta, q, T). \qquad (2)$$

For clipped gradients of each training data sample, $\|g\| \leq C$, Gaussian noise with distribution $\mathcal{N}(0, \sigma^2 C^2 I_d)$ is added to each gradient, where I_d denotes the d-dimensional identity matrix and d is the dimension of the gradient. Under these conditions, each sample is safeguarded by (ε, δ)-DP.

Methods

In this section, we present the proposed DP-SFT framework in detail, as illustrated in Fig. 2. The goal of DP-SFT is to protect sensitive data used during the fine-tuning of large language models under differential privacy constraints. The core idea of DP-SFT is to significantly reduce the dimensionality of noise injection by restricting it to a task-specific low-dimensional subspace of the gradient space. This strategy improves model utility and training stability while providing formal differential privacy guarantees.

DP-SFT consists of two main stages: Subspace Construction and Private Subspace Training. In the first stage, we identify a low-dimensional subspace that captures the principal directions of task-specific gradient variation. In the second stage, during each training step, incoming gradients are projected onto the identified subspace, Gaussian noise calibrated to satisfy the DP constraint is added, and the perturbed gradients are then mapped back to the original parameter space for model updates.

Stage 1: Subspace Construction

The first stage of DP-SFT aims to construct a task-specific low-dimensional subspace that captures the principal directions of parameter updates during fine-tuning, as shown in Algorithm 1. Under differential privacy (DP) constraints, directly injecting noise into the full high-dimensional parameter space can severely degrade model performance. To mitigate this, we identify a compact subspace that retains meaningful update directions, thereby improving training stability and model utility. Similarly, the recent work Mosformer (Cheng et al. 2025) in secure multi-party computation also employs dimensionality reduction strategies to effectively control communication and computational overhead.

Concretely, we perform one epoch of full-parameter fine-tuning on the private dataset $\mathcal{D}$ under (ε, δ)-DP constraints, and periodically record model parameters. Based on these snapshots, we construct a trajectory matrix:

$$A = \left[\Theta^{(1)} - \Theta^{(0)}, \Theta^{(2)} - \Theta^{(0)}, \ldots, \Theta^{(m)} - \Theta^{(0)}\right] \in \mathbb{R}^{m \times d}, \qquad (3)$$

where $\Theta^{(j)}$ denotes model parameters at step j ($j \in \{0, 1, \ldots, m\}$), d is the parameter dimensionality, and m is the number of recorded steps. The matrix A captures the evolution of model weights during early-stage fine-tuning.

We then apply Singular Value Decomposition (SVD) to obtain:

$$A = U \Sigma W^{\top}, \qquad (4)$$

where the columns of $W \in \mathbb{R}^{d \times d}$ represent orthogonal directions in the parameter space. We retain the top-k right singular vectors corresponding to the largest singular values and construct an orthogonal projection matrix $P \in \mathbb{R}^{d \times k}$, with $k \ll d$. This projection matrix defines a task-specific

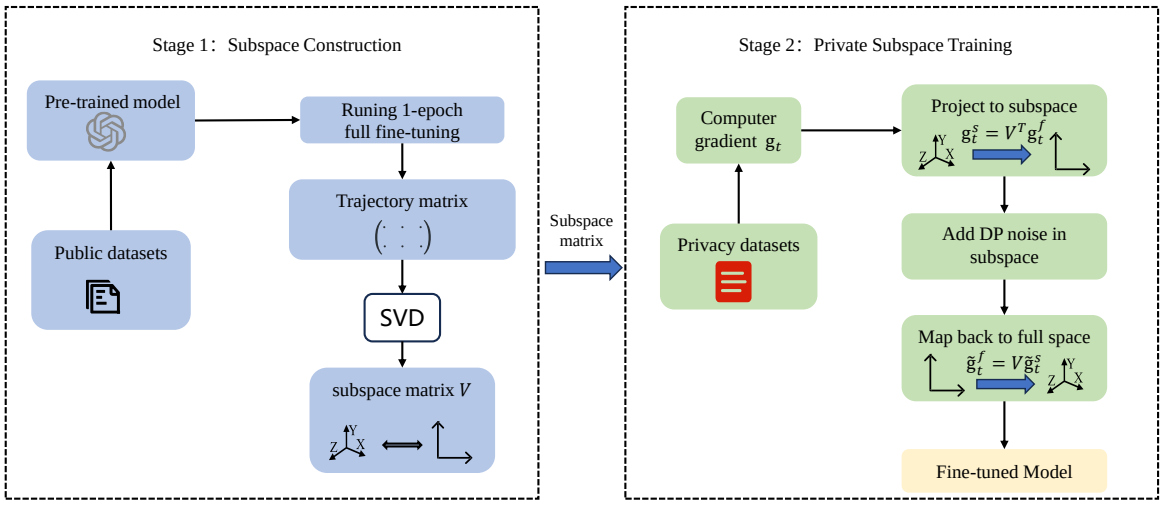

Figure 2: An overview of the proposed DP-SFT. DP-SFT consists of two stages: (1) Subspace Construction: a set of model snapshots during private training is collected and stacked into a trajectory matrix, from which a task-specific subspace is extracted via SVD. (2) Private Subspace Training: gradients are computed and projected into the subspace, where calibrated DP noise is added before mapping them back to the full parameter space for model updates.

subspace that captures the most significant optimization trajectories and serves as a dimensionality reduction operator between the full parameter space $\mathbb{R}^d$ and the subspace $\mathbb{R}^k$.

Importantly, subspace construction requires only short training trajectories to be effective. In our experiments, we further demonstrate that performing SVD on trajectory matrices collected from a brief fine-tuning process is sufficient to yield high-quality subspace projection matrices. These matrices support strong performance during subsequent private training. As a result, the subspace construction stage incurs negligible computational overhead, improving the overall practicality and scalability of our approach.

Stage 2: Private Subspace Training

After constructing the task-specific subspace, DP-SFT performs differentially private fine-tuning by restricting gradient updates to this subspace. The detailed procedure is presented in Algorithm 2. Unlike traditional DP-SGD, which injects noise into the entire high-dimensional parameter space $g_h \in \mathbb{R}^d$, our method operates solely in the low-dimensional subspace with projected gradients $g_z \in \mathbb{R}^k$, where $k \ll d$.

At each training step t, we compute the full per-sample gradient $g_t(x_i) = \nabla_{\Theta_t}\mathcal{L}(\Theta_t, x_i)$, flatten it, and project it to the subspace using an orthogonal matrix $P \in \mathbb{R}^{d\times k}$:

$$g_t^s(x_i) = P^\top g_t^f(x_i). \tag{5}$$

This projection compresses the gradient into a low-dimensional representation while preserving task-relevant directions.

We then perform ℓ_2-norm clipping with threshold C, followed by Gaussian noise injection:

$$\widetilde{g}_t^s = \text{Clip}(g_t^s, C) + \mathcal{N}(0, \sigma^2 C^2 I_k), \tag{6}$$

where σ is calibrated to the target privacy parameters (ε, δ). Since noise is injected in $\mathbb{R}^k$ rather than $\mathbb{R}^d$ where $k \ll d$ (with k being orders of magnitude smaller than d), the noise dimension is significantly reduced from d to k in comparison with standard DP-SGD.

The noisy gradient is then mapped back to the original parameter space:

$$\widetilde{g}_t^f = P \cdot \widetilde{g}_t^s \in \mathbb{R}^d, \tag{7}$$

reshaped into the original model parameter format, and used to update the model with a standard optimizer such as SGD or Adam.

From the perspective of differential privacy, subspace projection serves as a geometric safeguard. The projection matrix P effectively filters out gradient components that are orthogonal to the subspace, ensuring that only the most relevant directions are exposed to noise. For any gradient vector x, the transformation $P^\top x$ extracts its coordinates in the subspace basis, while Px reconstructs its subspace-aligned component in the original space.

Because the Gaussian mechanism is applied to the clipped gradient in the subspace (with bounded sensitivity), and all subsequent steps such as projection back and parameter reshaping are post-processing operations, the entire training procedure remains (ε, δ)-differentially private according to the composition and post-processing invariance properties of differential privacy. Compared to standard DP-SGD, DP-SFT achieves improved training stability and model utility by narrowing the noise injection domain to a low-dimensional, task-relevant subspace.

Privacy Guarantee

We formally prove that DP-SFT provides (ε, δ)-differential privacy for each training sample, based on the properties

Algorithm 1: Subspace Construction.

Require: Model parameters Θ_0 with d dimensions, subspace dimension k, training set S, learning rate η, loss function $\mathcal{L}$, training steps T, batch size B, clipping threshold C, privacy budget ε and relaxation term δ.

Ensure: Trained model parameters Θ^*, Projection matrix P.

Calculate standard deviation $\sigma = \text{gdp}\left(\varepsilon, \delta, \frac{B}{|S|}, T\right)$;
Calculate trajectory recording period $y = \lfloor \frac{T}{k} \rfloor$;
Create an empty matrix A;
for $t = 1$ to T **do**
  Sample a batch $\mathcal{B}_t$ of size B from training set S;
  for x_i in $\mathcal{B}_t$ **do**
    Calculate gradients $g_t(x_i) = \nabla_{\Theta_t}\mathcal{L}(\Theta_t, x_i)$;
    Clip $\overline{g_t}(x_i) = g_t(x_i) / \max\left(1, \frac{\|g_t(x_i)\|_2}{C}\right)$;
  end for
  Aggregate gradient: $\bar{g}_t \leftarrow \frac{1}{B}\sum_{x_i \in \mathcal{B}_t} \overline{g_t}(x_i)$;
  Add noise: $\widetilde{g}_t \leftarrow \bar{g}_t + \mathcal{N}(0, \sigma^2 C^2 I_d)$;
  if $t \mid y$ **then**
    Flatten trajectory $s = \text{flat}(\Theta_t - \Theta_0)$;
    Append s as a new row to matrix A;
  end if
  $\Theta_{t+1} \leftarrow Adam(\Theta_t, \widetilde{g}_t, \eta)$;
end for
$P \leftarrow \text{SVD}(A, k)$
return Θ^*, P.

Algorithm 2: Private Subspace Training.

Require: Model parameters Θ^* with d dimensions, matrix $P_{d\times k}$ with column vectors as standard orthogonal basis, training set S, learning rate η, loss function $\mathcal{L}$, training steps T, batch size B, clipping threshold C, privacy budget ε and relaxation term δ.

Ensure: Trained model parameters Θ.

Calculate standard deviation $\sigma = \text{gdp}\left(\varepsilon, \delta, \frac{B}{|S|}, T\right)$;
for $t = 1$ to T **do**
  Sample a batch $\mathcal{B}_t$ of size B from training set S;
  for x_i in $\mathcal{B}_t$ **do**
    Compute gradients $g_t(x_i) = \nabla_{\Theta_t}\mathcal{L}(\Theta_t, x_i)$;
    Flatten $g_t^f(x_i) = \text{flat}(g_t(x_i))$;
    Project $g_t^s(x_i) = P^\top \cdot g_t^f(x_i)$;
    Clip $\overline{g_t^s}(x_i) = g_t^s(x_i) / \max\left(1, \frac{\|g_t^s(x_i)\|_2}{C}\right)$;
  end for
  Aggregate gradient: $\bar{g_t^s} \leftarrow \frac{1}{B}\sum_{x_i \in \mathcal{B}_t} \overline{g_t^s}(x_i)$;
  Add noise: $\widetilde{g_t^s} \leftarrow \bar{g_t^s} + \mathcal{N}(0, \sigma^2 C^2 I_k)$;
  Finish projection $\widetilde{g_t^f} = P \cdot \widetilde{g_t^s}$;
  Recover shape $\widetilde{g}_t = \text{flat}^{-1}\left(\widetilde{g_t^f}\right)$;
  $\Theta_{t+1} \leftarrow Adam(\Theta_t, \widetilde{g}_t, \eta)$;
end for
return Θ.

of the Gaussian mechanism and the composition and post-processing invariance theorems of differential privacy.

Theorem 3 (Differential Privacy of DP-SFT)**.** *For any $p_1, p_2 \in (0, 1)$ such that $p_1 + p_2 = 1$, DP-SFT satisfies (ε, δ)-differential privacy for each data sample $x_i \in \mathcal{D}$.*

We decompose the training process into two stages and analyze the privacy guarantee of each:

(1) Subspace Construction. This phase involves performing a brief phase of full-parameter fine-tuning to extract optimization trajectories. Gaussian noise is added to gradient updates, which satisfies $(p_1\varepsilon, p_1\delta)$-DP according to the standard privacy analysis of DP-SGD (Abadi et al. 2016). As the singular value decomposition (SVD) and subspace construction are deterministic post-processing steps, the post-processing invariance property (Theorem 2) ensures that this phase remains $(p_1\varepsilon, p_1\delta)$-DP.

(2) Private Subspace Training. During training, each per-sample gradient $\nabla_{\Theta_t}\mathcal{L}(\Theta_t, x_i)$ is projected into the subspace using $P^\top$, clipped to norm C, and perturbed with Gaussian noise drawn from $\mathcal{N}(0, \sigma^2 C^2 I_k)$, where $k \ll d$. According to the Gaussian mechanism (Dwork et al. 2006), this step satisfies $(p_2\varepsilon, p_2\delta)$-DP. The reconstruction step using P is again post-processing and does not affect the privacy bound.

By the basic composition theorem (Dwork et al. 2006), the total privacy loss is at most $((p_1 + p_2)\varepsilon, (p_1 + p_2)\delta) = (\varepsilon, \delta)$, completing the proof.

Subspace Transferability

In practice, constructing the task-specific subspace does not strictly require access to the private dataset used for downstream fine-tuning. Instead, a semantically similar public dataset can be used to extract gradient trajectories and compute the subspace matrix. This is feasible because related tasks often share dominant optimization directions in the parameter space.

To validate this, we design a series of cross-task transfer experiments, where the subspace is constructed using non-sensitive public datasets (e.g., IMDB or SST-2) without applying differential privacy and then transferred to sensitive target tasks for private fine-tuning. Results show that even when the subspace is not derived from the target task, the performance difference remains negligible. This indicates that the extracted subspace primarily reflects general task structure and optimization geometry, rather than memorizing individual samples or private information.

Following standard practice in the differential privacy literature, we treat subspace extraction as a public preprocessing step that incurs no privacy cost. Moreover, this transferability highlights the subspace's role as a reusable structural prior, both effective for downstream training and compliant with privacy guarantees.

Experiments

In this section, we evaluate the effectiveness of our proposed method, DP-SFT, under different privacy budgets and benchmark tasks. We compare it with strong baselines cov-

ering both full-parameter and parameter-efficient differentially private fine-tuning methods.

Experimental Setup

Datasets To evaluate the effectiveness of DP-SFT, we conduct experiments on four widely used natural language understanding benchmarks: SST-2 (Socher et al. 2013), IMDB (Maas et al. 2011), QNLI (Wang et al. 2018), and MNLI (Williams, Nangia, and Bowman 2017). SST-2 is a binary sentiment classification task consisting of short movie reviews, commonly used to assess fine-tuning stability and sample efficiency. IMDB is another sentiment classification dataset, featuring longer, noisier, and more user-generated reviews, making it more privacy-sensitive. QNLI is a question–answer entailment task derived from the SQuAD dataset, where the model determines whether a candidate sentence correctly answers a given question. And MNLI is a large natural language inference benchmark involving sentence pairs from diverse genres labeled as entailment, contradiction, or neutral. Together, these datasets cover a range of task types and input lengths, enabling a comprehensive evaluation of model utility, robustness, and generalization under strict privacy constraints.

Base Model All experiments are conducted using the RoBERTa-base model (Liu et al. 2019), a widely adopted and robust pre-trained language model built on the Transformer architecture. RoBERTa-base consists of 12 encoder layers, each with 768 hidden dimensions and 12 self-attention heads, resulting in approximately 125 million parameters. Its strong performance and stability make it a suitable backbone for evaluating differentially private fine-tuning methods across diverse NLP tasks.

Baselines To validate the effectiveness of the proposed method, we compare DP-SFT with several representative baselines: (1) **Full-tuning** (Liu et al. 2019), which updates all model parameters without privacy protection; (2) **Full-DP** (Abadi et al. 2016) , a fully private variant using DP-SGD to ensure (ε, δ)-DP; (3) **LoRA-DP** (Yu et al. 2021b), a parameter-efficient method applying DP-SGD to low-rank adaptation modules; (4) **Adapter-DP** (Yu et al. 2021b), which inserts and trains adapter modules under DP constraints.

Metrics To evaluate the effectiveness of differentially private fine-tuning methods, we report accuracy on the validation sets of each dataset.

Implementation Details Our implementation is based on HuggingFace's Transformers library (Wolf et al. 2020). To ensure reproducibility and fair comparison, we use consistent settings across all methods: the batch size is fixed at 32; SST-2 and QNLI have a maximum input sequence length of 128, whereas IMDB and MNLI have a length of 256. For all DP-enabled methods (Full-DP, LoRA-DP, and Adapter-DP), we adopt a learning rate of 5×10^{-4} and a clipping threshold of 10. For the proposed DP-SFT, the subspace dimension is set to 32 for SST-2, QNLI, and IMDB, and 64 for MNLI. Privacy parameters are configured as $\delta = 10^{-5}$ for SST-2, QNLI, and IMDB, and $\delta = 10^{-6}$ for MNLI due to their dataset sizes, which is the same as (Yu et al. 2021d). The subspace construction stage runs for 1 epoch with a learning rate of 5×10^{-4}. Privacy parameters are configured as $\delta = 10^{-5}$. We evaluate the methods under two privacy budgets: $\varepsilon \in \{1, 4\}$. More implementation details can be found in the Supp. 1.

Method	SST-2	IMDB	QNLI	MNLI
Full-Tuning (Non-DP)	0.9507	0.9424	0.9249	0.8777
Full-DP	0.7592	0.7891	0.5995	0.3348
LoRA-DP (r=16)	0.8876	0.8886	0.8054	0.4327
Adapter-DP (r=48)	0.8909	0.8752	0.8056	0.7566
DP-SFT (Ours)	**0.9323**	**0.9352**	**0.9207**	**0.8678**

Table 1: Accuracy comparison of all methods under standard privacy constraint ($\varepsilon = 4$). *Note*: Bold values indicate the best performance among differentially private methods.

Method	SST-2	IMDB	QNLI	MNLI
Full-Tuning (Non-DP)	0.9507	0.9424	0.9249	0.8777
Full-DP	0.7500	0.7752	0.5854	0.3220
LoRA-DP (r=16)	0.8693	0.8394	0.7807	0.3424
Adapter-DP (r=48)	0.8624	0.6952	0.7820	0.7242
DP-SFT (Ours)	**0.9327**	**0.9290**	**0.9209**	**0.8652**

Table 2: Accuracy comparison of all methods under extreme privacy constraint ($\varepsilon = 1$). *Note*: Bold values indicate the best performance among differentially private methods.

Experiment Results

Performance of DP-SFT To comprehensively evaluate the effectiveness of our proposed framework, we report DP-SFT's performance under different privacy budgets across four benchmark datasets: SST-2, IMDB, QNLI, and MNLI. As shown in Tables 1 and 2, DP-SFT consistently achieves the best performance among all DP methods under both settings. Under $\varepsilon = 4$, DP-SFT outperforms the strongest baseline (Adapter-DP) by up to 11.51% on QNLI and 11.12% on MNLI, while maintaining a small gap of only 1.84% to the non-private full-tuning upper bound. This demonstrates that DP-SFT delivers near-lossless utility even with privacy constraints. Under the more challenging $\varepsilon = 1$ setting, performance degradation becomes more pronounced for all methods, but DP-SFT remains robust. It outperforms Adapter-DP by up to 13.89% on QNLI and 23.38% on IMDB, showing strong resistance to utility loss. Remarkably, DP-SFT retains over 86.5% accuracy on MNLI, while other methods drop below 73%. These results validate our hypothesis that confining gradient perturbation to a task-specific low-dimensional subspace not only reduces noise impact but also leads to significantly improved model utility and training stability.

Effectiveness of Short Trajectories for Subspace Construction To investigate whether short training trajectories can effectively capture task-specific optimization patterns for subspace construction, we perform an ablation study comparing subspace extraction using 1 epoch and

Training Scheme	SST-2	IMDB	QNLI	MNLI
Full-tuning	0.9507	0.9424	0.9249	0.8777
4-epoch subspace	0.9492	0.9406	0.9232	0.8717
1-epoch subspace	**0.9438**	**0.9388**	**0.9191**	**0.8665**
1-epoch vs Full	**-0.69%**	**-0.36%**	**-0.58%**	**-1.12%**

Table 3: Accuracy comparison between the few-epoch subspace in stage 1 and full fine-tuning

Transfer Direction	Accuracy	Δ vs Ideal
IMDB $\rightarrow$ SST-2	0.9071	-3.67%
SST-2 $\rightarrow$ IMDB	0.9100	-2.88%

Table 4: Performance of subspace transferability in non-DP settings

4 epochs of full-parameter fine-tuning, as well as standard 32-epoch full-tuning. As shown in Table 3, subspace models trained using only 1 epoch of the trajectory still achieve strong performance. Specifically, the 1-epoch subspace model achieves 94.38% on SST-2, 93.88% on IMDB, 91.91% on QNLI, and 86.65% on MNLI. These results demonstrate that short training trajectories (e.g., 1 epoch) suffice to extract meaningful subspaces for effective optimization under privacy constraints. Importantly, this approach reduces subspace construction time by 96.9%, thereby markedly enhancing the efficiency and applicability of the proposed method while maintaining stability.

Subspace Transferability In Tables 4 and 5, arrows such as "IMDB $\rightarrow$ SST-2" denote that the subspace is constructed using the source task (IMDB) and then transferred to the target task (SST-2) for differentially private training. The reported accuracy corresponds to model performance on the target task using the transferred subspace. To evaluate whether subspaces can leverage shared structural patterns for cross-task transfer, we design experiments under both non-private (Non-DP) and differentially private (DP, $\varepsilon = 4$) settings. Our hypothesis is that similar datasets share latent feature distributions, allowing effective subspace reuse without leaking private data. The results in Table 4 confirm that the subspace can encode transferable structure. In Table 5, DP training with transferred subspaces still maintains high performance, outperforming the Full-DP baseline by 14.22% on SST-2 and 11.99% on IMDB, despite small drops compared to ideal training. These findings verify two key properties: (1) subspaces can generalize across related tasks; (2) transferred subspaces retain model stability and utility under DP noise. Overall, subspace transfer presents an efficient and privacy-preserving approach to reusing task representations in large language model fine-tuning.

Impact of Noisy Trajectories To assess the effect of noise on subspace quality, we design an experiment where training trajectories are generated using DP-SGD with a privacy budget of $\varepsilon = 3$ in stage 1, followed by low-dimensional DP training with $\varepsilon = 1$ in stage 2, yielding a total budget of $\varepsilon_{\text{total}} = 4$. As shown in Table 6, although DP-SFT (Noisy) consistently outperforms the Full-DP baseline across all datasets, achieving up to +7.05% improvement on IMDB, its accuracy remains substantially lower than non-private baselines. For example, performance on QNLI (65.71%) and MNLI (35.75%) exhibits a sharp decline compared to non-DP references (92.49% and 87.77%). These findings highlight a fundamental limitation: injecting noise during subspace construction significantly reduces the fidelity of the learned subspace in representing the model's dominant update directions. This motivates our core design principle, which is to construct transferable subspaces using public data. As a result, we avoid early-stage noise contamination while preserving privacy guarantees and improving model utility during subsequent private fine-tuning. We leave the integration with secure inference systems like L-SecNet (Song et al. 2024) for future work.

Transfer Direction	Accuracy	Δ vs Ideal	Δ vs Full-DP
IMDB $\rightarrow$ SST-2	0.9014	-4.24%	+14.22%
SST-2 $\rightarrow$ IMDB	0.9090	-2.98%	+11.99%

Table 5: Performance of subspace transferability under DP ($\varepsilon = 4$)

Dataset	DP-SFT (Noisy)	Full-DP	Δ vs Full-DP
SST-2	0.7890	0.7592	**+2.98%**
IMDB	0.8596	0.7891	**+7.05%**
QNLI	0.6571	0.5995	**+5.76%**
MNLI	0.3575	0.3348	**+2.27%**

Table 6: Performance degradation from noisy trajectory construction ($\varepsilon_{\text{total}} = 4$)

Conclusion

We proposed DP-SFT, a differentially private subspace fine-tuning framework that mitigates the curse of dimensionality in LLM optimization under privacy constraints. By restricting noise injection to a task-specific low-dimensional subspace extracted via SVD on short training trajectories, DP-SFT substantially reduces the amount of noise required for privacy while preserving formal (ε, δ)-DP guarantees. Extensive experiments across multiple NLP benchmarks demonstrate that DP-SFT consistently outperforms existing full and parameter-efficient DP fine-tuning methods in both accuracy and training stability. Furthermore, we show that the extracted subspaces are transferable across tasks and can be constructed with negligible computational overhead, offering a scalable and practical solution for privacy-preserving adaptation of large models.

Acknowledgments

This paper is supported by the National Natural Science Foundation of China (62572373, 62220106004, 62402358, 92467201, 62502363, U24A20238), the National Key R&D Program of China (2023YFB3107500), the Key R&D Pro-

gram of Shandong Province of China (2023CXPT056, 2025CXPT089), the Young Talent Fund of Association for Science and Technology in Shaanxi, China (20240138), the Natural Science Basic Research Program of Shaanxi Province (2025JC-YBQN-869), the Aeronautical Science Foundation of China (20181981006), the Open Topics from the Lion Rock Labs of Cyberspace Security (#LRL24004), the Fundamental Research Funds for the Central Universities (ZDRC2202, ZYTS25081, KYFZ25005), the Xidian University Specially Funded Project for Interdisciplinary Exploration (TZJHF202502), the China Scholarship Council (CSC), the Double First-Class Overseas Research Project of Xidian University, and JSPS KAKENHI JP23K24851, JST PRESTO JPMJPR23P5, JST CREST JPMJCR21M2, JST NEXUS JPMJNX25C4.

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.

Alipour, H.; Pendar, N.; and Roy, K. 2024. Chatgpt alternative solutions: Large language models survey. *arXiv preprint arXiv:2403.14469*.

Balle, B.; and Wang, Y.-X. 2018. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, 394–403. PMLR.

Bie, A. 2024. *Private Distribution Learning with Public Data*. Master's thesis, University of Waterloo.

Bondarenko, Y.; Del Chiaro, R.; and Nagel, M. 2024. Low-Rank Quantization-Aware Training for LLMs. *arXiv preprint arXiv:2406.06385*.

Cheng, K.; Xia, Y.; Song, A.; Fu, J.; Qu, W.; Shen, Y.; and Zhang, J. 2025. Mosformer: Maliciously Secure Three-Party Inference Framework for Large Transformers. In *Proceedings of the 2025 on ACM SIGSAC Conference on Computer and Communications Security*.

Das, B. C.; Amini, M. H.; and Wu, Y. 2025. Security and privacy challenges of large language models: A survey. *ACM Computing Surveys*, 57(6): 1–39.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186.

Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.-M.; Chen, W.; et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3): 220–235.

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.

Gopi, S.; Lee, Y. T.; and Wutschitz, L. 2021. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34: 11631–11642.

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.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Kairouz, P.; Diaz, M. R.; Rush, K.; and Thakurta, A. 2021. (Nearly) Dimension Independent Private ERM with AdaGrad Rates via Publicly Estimated Subspaces. In *Conference on Learning Theory*, 2717–2746. PMLR.

Karimi Mahabadi, R.; Henderson, J.; and Ruder, S. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34: 1022–1035.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, X.; Tramer, F.; Liang, P.; and Hashimoto, T. 2021. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*.

Li, X.; Zmigrod, R.; Ma, Z.; Liu, X.; and Zhu, X. 2024. Fine-Tuning Language Models with Differential Privacy through Adaptive Noise Allocation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 8368–8375.

Liu, F. 2018. Generalized gaussian mechanism for differential privacy. *IEEE Transactions on Knowledge and Data Engineering*, 31(4): 747–756.

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*.

Maas, A.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 142–150.

Singhal, V.; and Steinke, T. 2021. Privately learning subspaces. *Advances in neural information processing systems*, 34: 1312–1324.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631–1642.

Song, A.; Fu, J.; Mu, X.; Zhu, X.; and Cheng, K. 2024. L-secnet: Towards secure and lightweight deep neural network inference. *Journal of Networking and Network Applications*, 3(4): 171–181.

Tsfadia, E. 2024. On Differentially Private Subspace Estimation Without Distributional Assumptions. *arXiv preprint arXiv:2402.06465*.

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. *arXiv preprint arXiv:1804.07461*.

Wang, L.; Chen, S.; Jiang, L.; Pan, S.; Cai, R.; Yang, S.; and Yang, F. 2025. Parameter-efficient fine-tuning in large language models: a survey of methodologies. *Artificial Intelligence Review*, 58(8): 227.

Williams, A.; Nangia, N.; and Bowman, S. R. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 38–45.

Yu, D.; Naik, S.; Backurs, A.; Gopi, S.; Inan, H. A.; Kamath, G.; Kulkarni, J.; Lee, Y. T.; Manoel, A.; Wutschitz, L.; et al. 2021a. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*.

Yu, D.; Naik, S.; Backurs, A.; Gopi, S.; Inan, H. A.; Kamath, G.; Kulkarni, J.; Lee, Y. T.; Manoel, A.; Wutschitz, L.; et al. 2021b. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*.

Yu, D.; Zhang, H.; Chen, W.; Yin, J.; and Liu, T.-Y. 2021c. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning*, 12208–12218. PMLR.

Yu, D.; Zhang, H.; Chen, W.; Yin, J.; and Liu, T.-Y. 2021d. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning*, 12208–12218. PMLR.

Zhao, J.; Zhang, Z.; Chen, B.; Wang, Z.; Anandkumar, A.; and Tian, Y. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

Supplementary Material

1.Implementation Details

Hardware Configuration All experiments were conducted on a server equipped with 8 NVIDIA A100 GPU (80GB memory per unit). This GPU has strong parallel computing capabilities, which can meet the needs of large-scale neural network training and high-dimensional matrix operations (such as singular value decomposition in subspace construction). The server is equipped with an Intel Xeon Platinum 8468 CPU, featuring 96 cores and 192 threads, providing sufficient computing power for CPU-intensive tasks such as data preprocessing and model loading.Moreover, the server memory is 2TB, which can efficiently support the loading and caching of large-scale training data, avoiding training interruptions due to insufficient memory.

Software Environment The experiments were conducted on Ubuntu 22.04 LTS, a stable and compatible operating system that ensures smooth experimental execution. The core software and frameworks used include: Python 3.12 as the programming environment; PyTorch 12.6 with CUDA 12.7 for constructing and training neural networks, leveraging GPU acceleration to enhance training efficiency; NumPy 2.3.1 for numerical computing tasks, such as matrix operations (e.g., singular value decomposition); Opacus 1.5.4 for implementing differential privacy mechanisms to ensure compliance with privacy constraints during training; and Hugging Face Transformers 4.51.3 for loading pre-trained models and performing text tokenization.

Handling Randomness To ensure the reproducibility of experimental results, this study strictly controls the seeds of all operations involving randomness. Specifically, the random number seed is fixed as follows:

In the initialization phase, the global random number seed for the PyTorch framework is set by calling:

```
torch.manual_seed(config.seed)
```

This ensures that random operations on the CPU (e.g., data shuffling, weight initialization) are deterministic.

If the experimental environment supports CUDA acceleration (i.e., `torch.cuda.is_available()` returns True), the same random number seed is set for all available GPU devices by calling:

```
torch.cuda.manual_seed_all(config.seed)
```

This guarantees that parallel computing on the GPU (e.g., CUDA kernel execution order, random number generation) produces consistent results.

The configuration parameter `config.seed` is fixed at 42, a commonly used benchmark seed value in machine learning research, ensuring reproducible results across different runs and devices.

This seed setting controls key operations such as data sample shuffling, random initialization of model parameters, and generation of differential privacy noise (e.g., noise sampling in the Opacus library). This effectively eliminates the impact of randomness on experimental outcomes.

Algorithm 3: DP-SFT.

Require: Model parameters Θ_0 with d dimensions, subspace dimension k, training set S, learning rate η_1, η_2, loss function $\mathcal{L}$, training epochs E_1, E_2, batch size B_1, B_2, clipping threshold C_1, C_2, privacy budget ε, relaxation term δ, privacy weight p_1, p_2.

Ensure: Trained model parameters Θ.

1: Training steps $T_1 = \left\lceil \frac{E_1|S|}{B_1 k} \right\rceil \cdot k$, $T_2 = \left\lceil \frac{E_2|S|}{B_1} \right\rceil$;
2: $\Theta^*, A = \mathrm{SC}(\Theta_0, d, S, \eta_1, \mathcal{L}, T_1, B_1, C_1, p_1\varepsilon, p_1\delta)$;
3: Construct matrix $P = \mathrm{SVD}(A, \mathrm{k})$
4: $\Theta = \mathrm{PST}(\Theta^*, P, S, \eta_2, \mathcal{L}, T_2, B_2, C_2, p_2\varepsilon, p_2\delta)$;
5: **return** Θ.

2. Procedure of DP-SFT

Algorithm 3 outlines the overall workflow of the proposed DP-SFT framework. For clarity, we refer to the `Subspace Construction` and `Private Subspace Training` stages as $\texttt{SC}(\cdot)$ and $\texttt{PST}(\cdot)$, respectively, in the pseudocode. The procedure begins by determining the number of training steps for each stage: T_1 for subspace construction and T_2 for subspace fine-tuning. To facilitate efficient matrix stacking during the first stage, T_1 is adjusted to be a multiple of the subspace dimension k, while T_2 corresponds to the standard number of full-parameter update steps (Line 1). Next, the $\texttt{SC}(\cdot)$ module is invoked to perform privacy-preserving training and collect parameter snapshots, resulting in intermediate parameters Θ^* and a trajectory matrix A (Line 2). Truncated singular value decomposition (SVD) is then applied to A to extract the top-k singular vectors, which constitute the orthonormal basis P of the task-specific subspace (Line 3). Finally, the $\texttt{PST}(\cdot)$ module performs differentially private training by constraining the gradient updates within the learned subspace, yielding the final model parameters Θ (Line 4).

Calculating T_1 and T_2 T_1 denotes the total number of update steps in the $\texttt{SC}(\cdot)$. It is computed by estimating the number of steps based on the dataset size, training epochs, and batch size, followed by applying the ceiling function and multiplying by the subspace dimension k. This ensures that parameter snapshots are collected at uniform intervals, facilitating the stable construction of the trajectory matrix. In contrast, T_2 represents the total number of training steps in the $\texttt{PST}(\cdot)$. It is computed based on the dataset size, the number of training epochs E_2, and the batch size B_2, using standard epoch-based scheduling. This design aligns with conventional training configurations, enabling a fair comparison with baseline methods.

Construction of Projection Matrix P Matrix A in $\texttt{SC}(\cdot)$ comprises parameter difference snapshots $\{\Theta_t - \Theta_0\}$ sampled at intervals of k steps. The formulation intentionally subtracts the initial parameters Θ_0 to eliminate the influence of the pre-trained base model's task-agnostic parameter distribution. By focusing on the delta between intermediate parameters and the initial state, this approach isolates the direction of parameter evolution specific to the

current task. Consequently, the extracted subspace emphasizes task-specific parameter variations while filtering out initialization-induced noise.

Singular Value Decomposition The operation $\texttt{SVD}(A, k)$ performs truncated singular value decomposition on matrix A and retains the top-k singular vectors to form the orthonormal basis P, which spans the low-dimensional subspace that captures the most significant task-specific variations.

Optimization in Private Subspace Training In the second stage, optimization performed by $\texttt{PST}(\cdot)$, differentially private (DP) updates are applied exclusively within the k-dimensional subspace defined by P, instead of the full d-dimensional parameter space. This restriction reduces the amount of noise required to ensure privacy by a factor of d/k, compared to full-parameter DP training. The total privacy budget $\varepsilon_{\text{total}} = \varepsilon$ is respected by allocating it between the two stages according to the privacy weights p_1 and p_2, such that $p_1 + p_2 = 1$.

3. Limitations and Future Work

Despite its effectiveness in balancing privacy preservation and model utility, the proposed DP-SFT method has several notable limitations that merit further investigation.

Sensitivity to Subspace Dimension k The performance of DP-SFT is highly sensitive to the choice of subspace dimension k. Although current method can be used to identify near-optimal values for specific tasks, such tuning often lacks generalizability across different model architectures and datasets. For example, smaller models with limited parameter capacity may require a larger k to retain sufficient task-relevant information, whereas larger models may maintain competitive performance with a smaller k, resulting in lower computational cost. However, the relationship between model scale and optimal subspace dimensionality remains insufficiently understood.

To address this challenge, future work could investigate dynamic k-selection mechanisms that adaptively determine the appropriate subspace dimensionality based on model size, dataset complexity, and task requirements. Techniques such as Bayesian optimization or reinforcement learning may offer promising solutions.

Evaluation Limitation The current evaluation of DP-SFT is confined to text classification tasks, and its applicability to other natural language processing tasks has yet to be explored. Tasks such as sequence labeling, machine translation, and generative modeling may involve different privacy-utility trade-offs and optimization dynamics, which could influence the effectiveness of the proposed method in more diverse settings.

Future work could focus on extending DP-SFT to a broader range of tasks beyond text classification, including generative applications such as privacy-constrained text generation and structured prediction tasks such as named entity recognition. Such efforts would require identifying and implementing task-specific adaptations to ensure both privacy preservation and performance across varying task characteristics.