DR-Encoder: Encode Low-rank Gradients with Random Prior for Large Language Models Differentially Privately

Huiwen Wu [1], Deyi Zhang [1], Xiaohan Li [1], Xiaogang Xu [2] *, Jiafei Wu [1] †, Zhe Liu [1]

[1] Research Center for Data Hub and Security, Zhejiang Laboratory, Hangzhou, China
[2] The Chinese University of Hong Kong, Hong Kong SAR, China
{whw,zhangdeyi,xiaohan}@zhejianglab.org, xiaogangxu00@gmail.com, {wujiafei,zhe.liu}@zhejianglab.org

Abstract

The emergence of the large language model (LLM) has shown its superiority in a wide range of disciplines, including language understanding and translation, relational logic reasoning, and even partial differential equations solving. The transformer is the pervasive backbone architecture for the foundation model construction. It is vital to research how to adjust the Transformer architecture to achieve an end-to-end privacy guarantee in LLM fine-tuning. This paper investigates three potential information leaks during a federated fine-tuning procedure for LLM (FedLLM). Based on the potential information leakage, we insert two-stage randomness into FedLLM to provide an end-to-end privacy guarantee solution. The first stage is to train a gradient auto-encoder with a Gaussian random prior based on the statistical information of the gradients generated by local clients. The second stage is fine-tuning the overall LLM with a differential privacy guarantee by adopting appropriate Gaussian noises. We show our proposed method's efficiency and accuracy gains with several foundation models and two popular evaluation benchmarks. Furthermore, we present a comprehensive privacy analysis with Gaussian Differential Privacy (GDP) and Rényi Differential Privacy (RDP).

1 Introduction

Large language models (LLMs) have demonstrated their strong capabilities in real-life applications, such as language understanding (Karanikolas et al. 2023), mathematical reasoning (Imani, Du, and Shrivastava 2023), and even differential equation solving (Herde et al. 2024). LLM-based applications provide a variety of convenient tools, including chatbots, virtual assistants, article writing, creative writing, and translation services. However, the extensive use of LLMs in daily work also poses a significant risk of unintentional leakage of personal information.

One typical approach to address privacy concerns in daily LLM usage is to use differential privacy during model training, achieved by adding extra Gaussian noise to the training data or intermediate gradients. However, these methods may lose their effectiveness when dealing with extensive input data and billions of model parameters in LLMs. This paper introduces a practical differential private gradient descent for fine-tuning pre-trained LLMs in a parallel manner with a pre-trained AutoEncoder to encode gradients based on a Gaussian prior. The aim is to achieve differential privacy at a client-level while preserving utility when using LLMs in downstream tasks.

In this paper, we propose a learning-based random prior gradient encoding method for federated LLM, called **DR-Encoder**. In our design of the pretraining AutoEncoder with a random prior, intermediate gradients are initially collected using low-rank decomposition (LoRA (Hu et al. 2021)) during individual training sessions. Afterward, we calculate the mean and variance from the low-rank gradients collected for each layer and training epoch. Subsequently, we create synthetic gradient data derived from the Gaussian distribution with the previously determined mean and standard deviations. This synthetic gradient-shaped data are then employed to train the Auto-Encoder for gradient compression and decompression. Next, we illustrate the process when applying the Auto-Encoder to fine-tune a Large Language Model (LLM). The first step involves placing the encoder and decoder on the client side. During federated LLM fine-tuning, original gradients are first perturbed, and then compressed into a subspace representation. This subspace gradient feature is transmitted to the global server, which then decompresses the gradients back into the LoRA shape and aggregates them. After aggregating the local gradients, the server sends back the aggregated gradients, which the client uses for the local gradient descent.

Compared to current methods that employ learning-based gradient compression (Wu et al. 2024; Lin et al. 2018; Li and Han 2019; Abrahamyan et al. 2021), our proposed method significantly enhances the privacy of the associated contributors. First, the gradient data used for training the AutoEncoder is synthesized from data generated using a Gaussian distribution rather than the real gradients, ensuring a strong privacy guarantee for the system. We rely solely on statistical information of gradients per layer and epoch for pre-training the AutoEncoder. Second, we apply a differential privacy mechanism to compressed gradients to maintain client-level differential privacy in the Federated Learning (FL) training setting. Finally, we offer accurate privacy for the entire federated system by considering randomness in client selection.

*corresponding author
†corresponding author

- We propose a novel compression training strategy for FL based on the mean and standard deviation gradients layer-wise. Experiments verifies the utility of the AutoEncoder trained via the synthetic data.
- A new differential privacy mechanism is devised to guarantee the privacy of local LoRA gradients.
- We adopt the Gaussian differential privacy and the Renyi differential privacy for a comprehensive privacy analysis both theoretically and numerically.

2 Related Work

2.1 Differential Privacy for LLMs

EW-Tune (Behnia et al. 2022) presents a differential privacy (DP) framework for fine-tuning LLMs using an edge-wise accountant. This approach ensures finite sample privacy guarantees through perturbation applied to the low-rank decomposition of the gradient matrix. The authors in (Shi et al. 2022) introduce a technique named Just Fine-tune Twice (JFT) that aims to achieve selective differential privacy (SDP) in large language models (LLMs) through two protective layers. The first layer, a low contextual detector, secures named entities, proper nouns, pronouns, and sentence components like subjects and objects, while the second layer, a high contextual detector, censors verbs to further enhance privacy. Whispered Tuning (Singh et al. 2024) is a multifaceted approach that integrates redaction of personally identifiable information, differential privacy techniques, and output filtering to improve privacy preservation in LLM. Split-N-Denoise (Mai et al. 2023) is a system crafted to protect user data privacy during the inference stage of large language models (LLM) by leveraging local differential privacy (LDP). It provides the user with a Transformer-based denoising model pre-trained on the server using public datasets and artificial noise. In (Charles et al. 2024), research study two variants of DP-SGD were investigated in the research study with sampling at the sample level and gradient clipping per sample to achieve differential privacy at the sample level and user-level sampling with clipping of gradient per user to achieve differential privacy at the user level.

2.2 Parameter Efficient Fine-tuning (PEFT)

Mixout approaches (Lee, Cho, and Kang 2019) integrate the standard network with the dropout network utilizing a specified probability. LoRA methods (Hu et al. 2021; Liu et al. 2024; Dettmers et al. 2024) decompose the gradient matrix and reconstruct it by multiplying the low-rank matrices. Adapter-based methods (Karimi Mahabadi, Henderson, and Ruder 2021; Mahabadi et al. 2021) introduce an additional adapter layer within the transformer layer, altering the network architecture. MagPruning methods (Han, Mao, and Dally 2015; Han et al. 2015; Lagunas et al. 2021) follow the principle that large weights are more important. By filtering out small weights in absolute values, it tunes the parameters with large absolute values only. DiffPruning (Mallya, Davis, and Lazebnik 2018; Guo, Rush, and Kim 2021) uses a Bernoulli random variable to represent the mask selection process and learns this variable through reparameterization methods. Child-Pruning (Xu et al. 2021; Mostafa and Wang 2019) trains in the full parameter space and calculates the projected mask to find the child network. In (Fu et al. 2023), the authors provide a unified sparse fine-tuning model containing random approaches, rule-based approaches, and projection-based approaches. Based on the proposed unified sparse fine-tuned model, it further provides comprehensive theoretical analysis for fine-tuning methods.

3 Methodology

3.1 Privacy Goal

The goal of our methods is to provide an end-to-end privacy guarantee for the gradient compression procedure in federated learning. We divide the whole gradients compress into two procedures. One is the pre-training of AutoEncoder to acquire the encoder and decoder for gradients compression. The second is federated fine-tuning with clients equipping encoder and server equipping with decoder. In the aforementioned two stages, there are several chances of information leakage. Firstly, when using local training gradients as input for AutoEncoder, the collection and transmission of the local gradients of the clients leak the sensitive information of the clients through the reconstruction attacker from the gradients to the original data (Petrov et al. 2024) and the membership inference from the gradients (Feng et al. 2024; Maini et al. 2024; Wei, Wang, and Jia 2024). Moreover, as we record the local training gradients as the input for training the AutoEncoder, the local data information is condensed into the models' weights of AutoEncoder. There is a chance to infer the original data information from the trained AutoEncoder, which are called model inversion attacks (Fredrikson, Jha, and Ristenpart 2015). We took the following steps to achieve the end-to-end privacy guarantee.

- Instead of transimission of the exact gradients from client to server, we transmit the stastics of gradients only. And we use the statistics to generate synthetic gradients for AutoEncoder pre-training.
- In the fine-tuning stage, instead of transmitting the exact gradients from client to server, we adopt differential privacy on the local gradients.
- A rigorous analysis of the privacy cost is presented to validate the privacy leakage in the entire federated system.

3.2 Pre-Training with Random Prior

In this section, we illustrate the process of training an AutoEncoder to grasp the statistical properties of the training gradients. Inspired by work on training deep neural networks for gradient compression (Li and Han 2019; Wu et al. 2024) and noise reduction (Mai et al. 2023) in conventional ML and LLM, we introduce a novel method to train the AutoEncoder by exclusively sharing the statistical details of the training gradients during the actual training phase. Thus, protecting the original gradient information during AutoEncoder training is crucial. To achieve this goal, we collect only the statistical information of the gradients, including the mean and standard deviation for each layer and epoch, and send them to the server. The information collected can be stored in the form $[\mathbf{m}^{\mathbf{t}}_{*,\mathbf{i}}, \mathbf{s}^{\mathbf{t}}_{*,\mathbf{i}}]^t_i$, where $*$ denotes the low-rank parts of $\mathbf{A}$ or $\mathbf{B}$, i represents the layer

Algorithm 1: RandomPrior($[\mathbf{A}_i^t, \mathbf{B}_i^t], \beta_1, \beta_2, h_1, h_2$)

Input: $[\mathbf{A}_i^t, \mathbf{B}_i^t]_{i\in[M],t\in[N]}$: Gradients tensor of client i at communication step t; hyper-parameters $\beta_1, \beta_2, h_1, h_2$;
Output $\mathbf{Enc}$: the pretrained gradients encoder; $\mathbf{Dec}$: the pretrained gradients decoder;
Client side:
for each client $i \in [M]$:

1: **Dynamically estimate mean epoch-wise:**

$$\mathbf{m}^{\mathbf{t}}_{\mathbf{A},\mathbf{i+1}} = \beta_1 \mathbf{m}^{\mathbf{t}}_{\mathbf{A},\mathbf{i}} + (1-\beta_1)\,\mathbf{A}^{\mathbf{t}}_{\mathbf{i}};$$

2: **Dynamically estimate variance:**

$$\mathbf{v_{i+1}} = \min\left(\max\left(\|\mathbf{A}^{\mathbf{t}}_{\mathbf{i}} - \mathbf{m}^{\mathbf{t}}_{\mathbf{A},\mathbf{i+1}}\|^2, h_1\right) h_2\right);$$

3: **Update estimate standard deviation:**

$$\left(\mathbf{s}^t_{i+1}\right)^2 = \beta_2 \left(\mathbf{s}^{\mathbf{t}}_{\mathbf{i}}\right)^2 + (1-\beta_2)\mathbf{v}^{\mathbf{t}}_{\mathbf{i+1}}, \forall \quad t \in [M];$$

4: **Compute the statistics for the counter part $\mathbf{B}_i^t$;**
5: **Send the collected statistics to Server.**

Server side:

1: **Generate the synthetic gradients:**

$$\hat{\mathbf{A}}^{\mathbf{t}}_{\mathbf{i}} \sim \mathcal{N}(\mathbf{m}^{\mathbf{t}}_{\mathbf{A},\mathbf{i}}, \mathbf{s}^{\mathbf{t}}_{\mathbf{A},\mathbf{i}}) \quad \hat{\mathbf{B}}^t_i \sim \mathcal{N}(\mathbf{m}^{\mathbf{t}}_{\mathbf{B},\mathbf{i}}, \mathbf{s}^{\mathbf{t}}_{\mathbf{B},\mathbf{i}});$$

2: **Train the AutoEncoder with synthetic gradients:**

$$\min \|\mathbf{Dec} \circ \mathbf{Enc}(\left[\hat{\mathbf{A}}^t_i, \hat{\mathbf{B}}^t_i\right]) - \left[\hat{\mathbf{A}}^t_i, \hat{\mathbf{B}}^t_i\right] \|;$$

3: **Send $\mathbf{Enc}$ to all clients.**

Algorithm 2: **DR-Encoder**($\mathbf{W}_0, \sigma, p, T, \mathbf{Enc}, \mathbf{Dec}$)

Input: initial foundation model parameters $\mathbf{W}^0$; $\mathbf{Enc}$: pretrained encoder with random prior; $\mathbf{Dec}$: pretrained decoder with random prior; σ: DP noise multiplier; η^t: learning rate; p: client selection probability ; T: iteration step;
Output Differential private model parameters $\mathbf{w}_i^T$.
Initialize: $\mathbf{W}_i^0 = \mathbf{w}^0$.
for communication round $t \in [N]$:
Client side:
for each client $i \in [M]$:

1: **Sample a subset of samples $\mathcal{I}_t$ with probability p;**
2: **for each client $k \in \mathcal{I}_t$ do**
3: **Compute low rank gradient per client:**

$$\mathbf{G}^{\mathbf{t}}_{\mathbf{i}} \approx \mathbf{A}^{\mathbf{t}}_{\mathbf{i}} \mathbf{B}^{\mathbf{t}}_{\mathbf{i}};$$

4: **Compress local gradients:**

$$\dot{\mathbf{A}}^{\mathbf{t}}_{\mathbf{i}} = \mathbf{Enc}(\mathbf{A}^{\mathbf{t}}_{\mathbf{i}}), \quad \dot{\mathbf{B}}^{\mathbf{t}}_{\mathbf{i}} = \mathbf{Enc}(\mathbf{B}^{\mathbf{t}}_{\mathbf{i}})$$

5: **Clip local gradients with Eq.** (1) **and Eq.** (2)**;**
6: **Noise local gradients with Eq.** (3) **and Eq.** (4)**;**
7: **Send the local gradients to server;**
8: **end for**

Server side:

1: **Aggregate per client gradients according to Eq.** (5)**;**
2: **Decode the compressed gradients with $\mathbf{Dec}$;**
3: **Do gradient descent with decompressed gradients as Eq.** (6)**;**
4: **Send the updated model parameters back to client.**

index and t indicates the epoch index. We adopt a dynamical way to compute the mean and standard deviation of local gradients by layer and epoch, which follows steps 1 to 4 on the client side as shown in Algorithm 1. The hyperparameters $\beta_1, \beta_2, h_1, h_2$ are small scalars. In our experiments, we use $\beta_1 = 0.99, \beta_2 = 0.9, h_1 = 10^{-5}$ and $h_2 = 10^{-3}$. When server receiving the collected mean and standard deviation, it generates synthetic gradients with Gaussian distribution. The synthetic gradients are in the form $[\hat{\mathbf{G}}^{\mathbf{t}}_{\mathbf{i}}] = [\hat{\mathbf{A}}^{\mathbf{t}}_{\mathbf{i}}, \hat{\mathbf{B}}^{\mathbf{t}}_{\mathbf{i}}], \hat{\mathbf{A}}^{\mathbf{t}}_{\mathbf{i}} \sim \mathcal{N}(\mathbf{m}^{\mathbf{t}}_{\mathbf{A},\mathbf{i}}, \mathbf{s}^{\mathbf{t}}_{\mathbf{i}}), \mathbf{B}^{\mathbf{t}}_{\mathbf{i}} \sim \mathcal{N}(\mathbf{m}^{\mathbf{t}}_{\mathbf{B},\mathbf{i}}, \mathbf{s}^{\mathbf{t}}_{\mathbf{B},\mathbf{i}})$. The server uses synthetic gradients to train the AutoEncoder with loss of ℓ_2 reconstruction. Once the server completes training the AutoEncoder, it separates the AutoEncoder into an encoder and a decoder, dispatching the encoder to all clients while retaining the decoder exclusively. We present the training details in Algorithm 1. We use a fundamental architecture for AutoEncoder and present the details in the Supplementary Material Sect.3.

3.3 Differentially Private Federated Fine-tuning with LoRA gradients

First of all, we lay the foundation algorithm for fine-tuning the LLM model with differential LoRA gradients privately. Several works consider training LLMs according to differential privacy constraint. For example, in (Charles et al. 2024), the author proposes a sample-level differential privacy and a user-level differential privacy method. In our work, our goal is to provide differential client-level privacy for the entire FL system. In each client at iteration t, the gradient is decomposed to a low-rank decomposition for light transmission. $\mathbf{G}_i^t = \mathbf{A}_i^t \mathbf{B}_i^t, \quad \text{where} \quad \mathbf{A}_i^t \in \mathbb{R}^{n\times r}, \mathbf{B}_i^t \in \mathbb{R}^{r\times n}$. We first compress the gradients with the $\mathbf{Enc}$ trained in Algorithm 1. We then clip the gradient.

$$\bar{\mathbf{A}}^t_i \;\; = \mathrm{Clip}(\dot{\mathbf{A}}^{\mathrm{t}}_{\mathrm{i}}) \;\; = \|\dot{\mathbf{A}}^t_i\| \min(1, \frac{C}{\|\dot{\mathbf{A}}^t_i\|_F}); \qquad (1)$$

$$\bar{\mathbf{B}}^t_i \;\; = \mathrm{Clip}(\dot{\mathbf{B}}^{\mathrm{t}}_{\mathrm{i}}) \;\; = \|\dot{\mathbf{B}}^t_i\| \min(1, \frac{C}{\|\dot{\mathbf{B}}^t_i\|_F}). \qquad (2)$$

Next, we exert an random noise on the gradient before transmitted to the global server.

$$\tilde{\mathbf{A}}^t_i \;\; = \bar{\mathbf{A}}^t_i + \mathcal{N}(0, 4\sigma_A^2/K^2); \qquad (3)$$

$$\tilde{\mathbf{B}}^t_i \;\; = \bar{\mathbf{B}}^t_i + \mathcal{N}(0, 4\sigma_B^2/K^2). \qquad (4)$$

The magnitude of noise exerted on each client's gradient is computed according to the differential privacy mechanism. In our design, we use a homogeneous σ for $\mathbf{A}$ and $\mathbf{B}$, as

$$\sigma_A = \sigma_B = \mathbf{DPAccountant}(\epsilon, \mathrm{T}, \mathrm{p}).$$

The privacy accountant **DPAccountant** can be chosen utilizing Gaussian Differential Privacy (GDP) (Dong, Roth, and Su 2019) or Rényi Differential Privacy (RDP) (Mironov 2017). We elaborate on the calculation procedure in

Sect. 4.2. After adding noise, the client sends the differential private gradients to the server. When the server receives the differential private gradients, it performs the aggregation and is followed by the denoising process.

$$\tilde{\mathbf{G}}^t = \frac{1}{N}\sum_{i=1}^{N}\tilde{\mathbf{G}}_i^t = \frac{1}{N}\sum_{i=1}^{N}\tilde{\mathbf{A}}_i^t\tilde{\mathbf{B}}_i^t. \tag{5}$$

Then the gradient descent is implemented with the aggregated gradients.

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta^t\mathbf{G}^t. \tag{6}$$

After that, the server sends the updated model parameters to each selected client. Then the system runs into the next iteration. This loop continues until we run out the privacy budget ϵ or the system diverges due to the accumulated random noise. We describe the procedure formally in Algorithm 2.

4 Theoretical Analysis

Here, we derive the theoretical analysis to show how to achieve client-level differential privacy based on our proposed methods.

4.1 Priliminary

Definition 4.1. *(ℓ_2-sensitivity (Dwork, Roth et al. 2014)). The ℓ_2-sensitivity of a function $f : \mathcal{D} \to \mathbb{R}^d$ is:*

$$\Delta_2 f = \max_{x,y\in\mathcal{D},\|x-y\|=1} \|f(x) - f(y)\|_2.$$

Definition 4.2. *(The Gaussian Mechanism (Dwork, Roth et al. 2014)) Given a function $f : \mathcal{D} \to \mathbb{R}^d$ on a data set $\mathcal{D}$, the Gaussian mechanism is defined as:*

$$\mathcal{M}_G(x, f(\cdot), \epsilon) = f(x) + (Y_1, \cdots, Y_k)$$

where Y_i are i.i.d. random variables drawn from $\mathcal{N}\left(\sigma^2\Delta_2 f^2\right)$ and $\sigma = \frac{2\ln(1.25/\delta)}{\epsilon}$.

Theorem 4.1. *(Dwork, Roth et al. 2014) The Gaussian mechanism defined in Definition 4.2 preserves $(\epsilon, \delta)-$differential privacy.*

4.2 Privacy of DR-Encoder

First, we show that, with clip value 1, the sensitivity of the composition of gradient aggregation in Eq. (5) is $2/K$.

Lemma 4.1. *With clip value 1, the sensitivity of gradient aggregation defined in Eq.* (5) *is $2/K$, with K be the number of selected clients.*

Then we show with sensitivity equal 1 and the noise multiplier σ, the privacy loss for per iteration is $G_{1/\sigma}$-DP.

Lemma 4.2 (Privacy per iteration)**.** *Suppose* Noise *with random variable sampled from Gaussian mechanism $\mathcal{N}(0, 4\sigma^2/K^2)$. Then **DR-Encoder** (Algorithm 2) for per gradient update satisfies $G_{1/\sigma}$-DP, where $G_{1/\sigma}(\alpha) = \Phi(\Phi^{-1}(1-\alpha) - 1/\sigma)$ and Φ denotes the standard normal cumulative distribution function.*

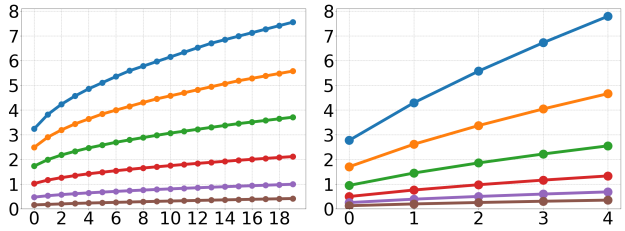


Figure 1: The graph illustrates the accumulation of privacy loss through Rényi Differential Privacy (RDP). The x-axis represents the communication steps, while the y-axis shows the accumulated privacy budget. Different colors on the graph correspond to varying privacy budget usage under different noise magnitudes (σ), as detailed in Table 3.

By applying Lemma 4.2 to mechanism defined in Algorithm 2, we have the privacy analysis of **DR-Encoder** for per gradient update is $G_{1/\sigma}-$DP. After that, one can prove the privacy analysis of training equipped with **DR-Encoder**, with subsampling amplification (for SGD or mini-batch SGD) and the central limit theorem of composition over iteration T via GDP (Bu et al. 2019). According to the Central Limit Theorem (Theorem 5 (Bu et al. 2019)), we have approximated G_μ for the accumulated privacy loss of **DR-Encoder**. Furthermore, we convert the accumulated $\mu-$GDP back to $(\epsilon, \delta)-$DP with the primal-dual result (Corollary 2.13 in (Dong, Roth, and Su 2019)). Then we have the main result of the privacy analysis of **DR-Encoder**, which preserves $(\epsilon, \delta)-$DP with noise multiplier $2\sigma/K$, number of iterations T, and subsampling rate p.

Theorem 4.2 (Gaussian Differential Privacy (Bu et al. 2019))**.** *Suppose Algorithm 2 run with number of steps T and Poisson sampling without replacement with probability $p = K/M$, which satisfy $p\sqrt{T} \to \nu$. Then $C_p\left(G_{1/\sigma}\right)^{\otimes T} \to G_\mu$ uniformly as $T \to \infty$ where*

$$\mu = \nu \cdot \sqrt{(e^{1/\sigma^2} - 1)}. \tag{7}$$

By solving Eq. (7) inversely, we get σ via predefined ϵ and δ. We present the detailed relation between σ and ϵ, δ in Sect. 5.2. Furthermore, we display the accumulation procedure of privacy loss via RDP in Figure 1. Due to the limited page, we show all the related proofs in the Supplementary Material Sect.2.

5 Experiments

In this segment, we perform extensive experiments to address the following research inquiries.

- **RQ1** How does the **DR-Encoder** train by Random Prior work in the fine-tuning scenario?
- **RQ2** How much noise do we exert on the gradient to guarantee DP with various budgets?
- **RQ3** How does the random mechanism to achieve DP influence the FedLLM fine-tuning?
- **RQ4** What is the difference in performance between the AutoEncoder trained with the informative prior and with the random prior?

Methods	Communication Rounds	Client Selection Fraction	Learning Rates	Low Rank	# Train Samples	# Test Samples
LlaMa-Dolly	20	0.05	1.5×10^{-4}	8	149	13948
Qwen-MMLU	4	1.0	3×10^{-4}	8	285	14042

Table 1: Hyperparameters Details of Fine-tuning Examples

Methods	Stem	Social	Humanities	Others	Avgerage	Avg(Hard)
Qwen7B-DR-Encoder-MMLU	46.93	65.42	51.17	63.75	56.13	
Qwen7B-FedCG-MMLU	**47.67**	**66.01**	50.86	64.33	56.44	
Qwen7B-LoRA-MMLU	47.61	65.62	**52.09**	64.4	**56.77**	
Qwen7B-Cent-MMLU	47.32	65.58	51.43	**64.95**	56.6	
Qwen7B-Base-MMLU	46.62	65.52	51.2	63.69	56.07	
LlaMa7B-DR-Encoder-CEval	26.6	26.5	25.5	25.7	26.2	26.8
LlaMa7B-FedCG-CEval	**26.8**	26	**26.8**	**26.5**	**26.6**	**26.9**
LlaMa7B-LoRA-CEval	25.9	**27.6**	25.2	24.5	25.8	24.8
LlaMa7B-Cent-CEval	24.5	25.6	25.5	24.4	24.9	23.4
LlaMa7B-Base-CEval	21.6	23.4	23.9	23.3	22.8	20.3

Table 2: Fine-tuning advancements for the Qwen and LlaMa models assessed on MMLU and C-Eval, respectively. The MMLU evaluation comprises five subjects: 'Stem', 'Social', 'Humanities', 'Others', and 'Average'; whereas the C-Eval includes an additional subject termed 'Avg(Hard)'.

	LlaMa			**Qwen**	
σ	RDP ϵ	GDP ϵ	σ	RDP ϵ	GDP ϵ
0.00	∞	∞	0.00	∞	∞
0.46	**7.55**	8.00	1.10	**7.79**	8.00
0.52	**5.57**	4.00	1.63	4.66	**4.00**
0.60	3.71	**2.00**	2.64	2.55	**2.00**
0.75	2.12	**1.00**	4.47	1.33	**1.00**
1.00	1.00	**0.50**	7.70	0.69	**0.50**
1.45	0.42	**0.25**	13.24	0.36	**0.25**

Table 3: Comparison of noise multiplier σ and privacy budget ϵ as calculated by RDP and GDP.

- **RQ5** What is the gain in communication efficiency of the proposed method?

5.1 Experimental Settings

Now we present the details of experimental settings including foundation models, datasets, evaluation benchmarks, and hyper-parameters that we utilized throughout all papers.

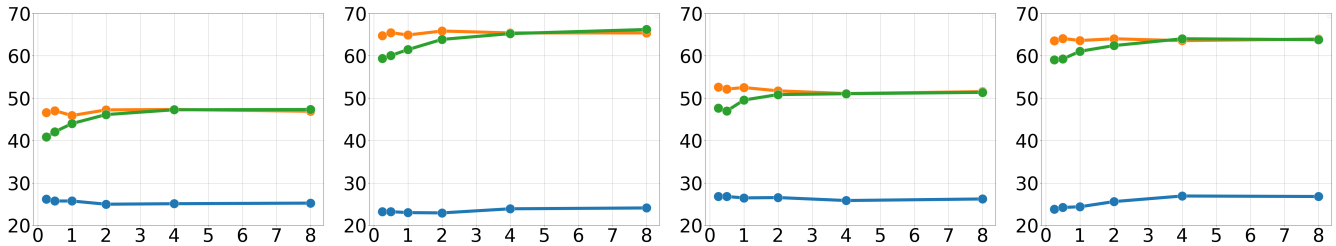
For the foundational models, we adopt the fine-tuning strategy based on LlaMa-7B and Qwen-7B. LlaMa is an open and efficient LLM foundation model developed by Meta with parameter sizes ranging from 7B to 65B (Touvron et al. 2023). Qwen is developed and released by Alibaba Group (Bai et al. 2023). Due to the limited computational resource, we carry out the experiments on the 7B version. For the fine-tuning of our large language models (LLMs), we leveraged two primary datasets: MMLU-train (Hendrycks et al. 2020) and Databricks-dolly-15k (Conover et al. 2023). The MMLU-train dataset encompasses 57 tasks in diverse disciplines, such as elementary mathematics, United States history, computer science, legal studies, and more. Meanwhile, Databricks-dolly-15k is structured into eight categories, including brainstorming, classification, closed QA, creative writing, general QA, information extraction, open QA, and summarization. To evaluate the performance of our LLM, we used the C-Eval (Zhang et al. 2023b) and MMLU (Hendrycks et al. 2020) benchmarks, which provide comprehensive, multidisciplinary assessments designed specifically to evaluate LLM capabilities. Drawing on established research, we select the hyperparameters for FedLLM (Zhang et al. 2023b,a). Details on the fine-tuning process and the hyperparameters chosen are presented in Table 1. The compared methods are '**Cent**', which stands for centralized fine-tuning, '**LoRA**', representing subspace decomposition of local gradient parameters, '**FedCG**', indicating gradient compression with a knowledgeable prior developed in (Wu et al. 2024), and '**DR-Encoder**', referring to our approach involving gradient compression with random prior. Each experiment is labeled in the format '**A-B-C**', where **A** represents the base model, **B** signifies the tuning technique, and **C** indicates the evaluation criteria.

5.2 Foundation Models Improvement (RQ1)

In this section, we demonstrate how various fine-tuning strategies enhance model performance over the foundational ones. We detail the fine-tuning of the Qwen model with MMLU datasets and its evaluation in MMLU, as well as the fine-tuning with dolly-15k datasets and evaluation on C-Eval, in Table 2. From Table 2, it is evident that fine-tuning methods significantly improve accuracy compared to foundational methods. For instance, in the second part (C-Eval) of Table 2, the accuracy improvement of **FedCG** is 3.8 for the 'Average' subject and 6.3 for 'Avg(Hard)' relative to the foundational model. In the first part (MMLU) of Table 2, the **LoRA** method exhibits an accuracy increment of 0.7 in the 'Average' subject while **FedCG** shows an increment of 0.37 compared to the **Qwen7B-Base-MMLU** model. Moreover, the **DR-Encoder** with an autoencoder trained using random prior information demonstrates a comparable fine-

	Methods	Stem	Social Sciences	Humanities	Others	Average	Avg(hard)
	MMLU						
1	**Qwen-LoRA-ϵ-0.25**	26.2	23.24	26.8	23.82	25.22	
	Qwen-LoRA-ϵ-0.5	25.75	23.23	26.8	24.23	25.21	
	Qwen-LoRA-ϵ-1.0	25.75	23.01	26.48	24.39	25.09	
	Qwen-LoRA-ϵ-2.0	24.99	22.94	26.56	25.61	25.21	
	Qwen-LoRA-ϵ-4.0	25.11	23.91	25.86	26.93	25.5	
	Qwen-LoRA-ϵ-8.0	25.25	24.11	26.23	26.81	25.67	
2	**Qwen-FedCG-ϵ-0.25**	46.62	64.8	**52.64**	63.56	56.37	
	Qwen-FedCG-ϵ-0.5	47.06	65.48	52.17	64.11	56.58	
	Qwen-FedCG-ϵ-1.0	45.92	64.93	52.53	63.63	56.22	
	Qwen-FedCG-ϵ-2.0	47.25	65.87	51.75	64.04	56.55	
	Qwen-FedCG-ϵ-4.0	47.35	65.42	51.11	63.59	56.16	
	Qwen-FedCG-ϵ-8.0	46.97	65.42	51.56	63.98	56.31	
3	**Qwen-DR-Encoder-ϵ-0.25**	40.84	59.4	47.65	59.09	51.23	
	Qwen-DR-Encoder-ϵ-0.5	42.08	60.09	47.01	59.28	51.48	
	Qwen-DR-Encoder-ϵ-1.0	44.02	61.48	49.58	61.08	53.48	
	Qwen-DR-Encoder-ϵ-2.0	46.14	63.89	50.86	62.43	55.22	
	Qwen-DR-Encoder-ϵ-4.0	47.28	65.25	51.07	64.04	56.2	
	Qwen-DR-Encoder-ϵ-8.0	47.35	**66.23**	51.39	63.82	56.48	
4	**Qwen-DR-Encoder-ϵ-∞**	46.93	65.42	51.17	63.75	56.13	
	Qwen-FedCG-ϵ-∞	**47.67**	66.01	50.86	64.33	56.44	
	Qwen-LoRA-ϵ-∞	47.61	65.62	52.09	64.4	**56.77**	
	Qwen-Cent-ϵ-∞	47.32	65.58	51.43	**64.95**	56.6	
	Qwen-Base	46.62	65.52	51.2	63.69	56.07	
	C-Eval						
5	**LlaMa-FedLoRA-ϵ-4.0**	24.3	24.2	24.4	23.9	24.2	23.8
	LlaMa-FedLoRA-ϵ-8.0	24.5	25.7	26.1	25.9	25.4	24
6	**LlaMa-FedCG-ϵ-0.25**	**26.8**	25.3	**26.4**	**26.6**	**26.4**	**27.2**
	LlaMa-FedCG-ϵ-0.5	26.6	26.5	25.5	25.7	26.2	26.8
	LlaMa-FedCG-ϵ-1.0	24.9	24.8	24.3	24.5	24.7	25.8
	LlaMa-FedCG-ϵ-8.0	21.7	23.3	23.8	23.3	22.8	20.3
7	**LlaMa-DR-Encoder-ϵ-0.25**	26.6	26.5	25.4	25.7	26.1	26.8
	LlaMa-DR-Encoder-ϵ-0.5	22.2	24	24.4	23.9	23.3	20.3
	LlaMa-DR-Encoder-ϵ-2.0	24.7	25.3	24.8	23.9	24.7	25
	LlaMa-DR-Encoder-ϵ-8.0	26.6	26.2	25.6	25.8	26.1	26.8
8	**LlaMa-DR-Encoder-ϵ-∞**	26.6	26.5	25.5	25.7	26.2	26.8
	LlaMa-FedCG-ϵ-∞	26.6	26.5	25.5	25.7	26.2	26.8
	LlaMa-LoRA-ϵ-∞	25.9	**27.6**	25.2	24.5	25.8	24.8
	LlaMa-Cent-ϵ-∞	24.5	25.6	25.5	24.4	24.9	23.4
	LlaMa-Base	21.6	23.4	23.9	23.3	22.8	20.3

Table 4: Tests across Various Privacy Budget Levels, assessed through MMLU (Rows 1 to 4) and C-Eval (Rows 5 to 8). The MMLU assessment includes 5 subjects, whereas the C-Eval includes an additional one called 'Avg(hard)'.



(a) Stem (b) Social Sciences (c) Humanities (d) Others

Figure 2: MMLU Performance Analysis across Four Disciplines. The y-axis shows the evaluation score, while the x-axis shows the privacy budget. The yellow line denotes **FedCG**, the green line denotes **DR-Encoder**, and the blue line denotes **LoRA**.

tuning performance to **FedCG** with an autoencoder trained on raw gradients, outperforming **LoRA** without gradient compression. For example, in the C-Eval evaluation, the highest score of 26.6 for the 'Average' subject is achieved by **FedCG**, while the second highest score of 26.2 is achieved by **DR-Encoder**, which surpasses **LoRA**'s performance of 25.8. A similar trend is observed for the 'Avg(Hard)' subject.

5.3 Privacy Accounting via RDP or GDP (RQ2)

This section discusses how to account for privacy loss in the FedLLM system. We employ the RDP (Mironov 2017) and GDP (Dong, Roth, and Su 2019) methods to account for loss of privacy. We provide a table showing the one-to-one relationship between the privacy budget ϵ and the noise multiplier for both RDP and GDP (Table 3). The calculation of GDP follows the steps described in Sect. 4.2. Initially, we set the privacy budget from the set $\{2^k | k \in \mathcal{N}, -3 \leq k \leq 2\}$ (columns 2 and 5). Next, we determine the magnitude of the noise multiplier for each client denoted as σ. Each client has an evenly distributed amount of Gaussian noise added. After obtaining the added noise (columns 1 and 4), we use RDP to aggregate the privacy cost measured in RDP according to established research (Mironov 2017; Balle and Wang 2018) and derive the RDP ϵ (columns 3 and 6). From Table 3, we note that when σ is small, RDP indicates a lower loss of privacy than GDP. In contrast, GDP results in a lower privacy loss when σ is large and the privacy budget is small. In subsequent sections, we use ϵ derived via GDP. It should also be noted that RDP demonstrates a lesser privacy loss when ϵ is large. Furthermore, we illustrate the privacy loss accumulation process in Figure 1. Additional details on training hyperparameters and privacy parameters can be found in the Supplementary Materials, Sect.1 and Sect.2.

5.4 Overall Influence of Differential Privacy (RQ3)

In Table 4, we present the training results with various privacy budgets of the set $\{0.25, 0.5, 1.0, 2.0, 4.0, 8.0\}$. Smaller privacy budgets ensure stronger privacy guarantees. Based on the results shown in Table 4, we observe that the inclusion of Gaussian noise in the low-rank decomposition of gradients to provide differential privacy leads to a decline in model performance. As illustrated in the first row of Table 4, adding Gaussian noise directly to the LoRA gradients results in the 'Average' score dropping from 56.77 to 25.67. This shows that Gaussian noise has a negative impact on model training. For the LlaMa model, there are no evaluation results for LoRA methods with small ϵ due to the substantial noise applied to the gradients.

5.5 Comparison Between FedCG and DR-Encoder (RQ4)

With the use of an informative pre-trained AutoEncoder, **FedCG** achieves performance comparable to the nonprivate scenario. For example, the scores obtained by **Qwen-FedCG-ϵ-0.5** are close to those of **Qwen-LoRA-ϵ-∞**. This is attributed to the strong denoising capability of the informative AutoEncoder. A similar pattern can be observed for the LlaMa example. Moreover, **FedCG** displays consistent performance across different privacy budgets. This consistency is likely a result of the denoising capabilities of the informative AutoEncoder. In contrast to **FedCG**, **DR-Encoder** exhibits a downward trend in evaluation scores as privacy budgets become more stringent. This aligns with the general understanding that a smaller privacy budget results in more significant Gaussian noise being added, thereby impairing overall model performance.

	#Parameters	Storage
FedCG	167,772,160	160 MB
DR-Encoder	10,240	0.078 MB
CEG	6.10×10^{-5}	

Table 5: Efficiency Improvement of **DR-Encoder** During the AutoEncoder Pretraining Phase.

5.6 DR-Encoder with Large Privacy Budget (RQ4)

Most notably, even when employing an AutoEncoder trained with a Random Prior, **DR-Encoder** efficiently alleviates the performance degradation caused by differential privacy for moderate privacy budgets of $\epsilon = 1.0$ and 2.0. Interestingly, with looser privacy budgets of $\epsilon = 4.0$ and 8.0, **DR-Encoder** surpasses the non-privacy benchmarks, such as **Compress-ϵ-∞** and **Cent-ϵ-∞**. This happens because looser privacy budgets lead to a smaller amount of random noise, thus improving the generalizability of the fine-tuned model.

5.7 Communication Efficiency Gain (RQ5)

In this section, we demonstrate the communication efficiency gain (CEG) of our method **DR-Encoder** in comparison to **FedCG**. In the initial stage, which involves collecting gradients and transmitting them to the server for autoencoder pre-training, for the LlaMa model with low-rank parameters $r = 8$, **FedCG** requires $4096 \times 8 \times 2 \times 4 \times 32 \times 20 = 167,772,160 \approx 160MB$ of parameters. In contrast, for **DR-Encoder**, we only send the mean and standard deviation for each layer and the gradients of each epoch, with a total parameter count of $2 \times 2 \times 4 \times 32 \times 20 = 10,240 \approx 0.078MB$. We reduce the communication complexity in the AutoEncoder preprocessing stage to 6.10×10^{-5}, illustrating the substantial efficiency gain of our proposed AutoEncoder training with Random Prior. We summarize this comparison of model parameters, storage, and efficiency gain in Table 5.

6 Conclusion

In this paper, we introduce a practical learning-based gradient compression method to achieve client-level differential privacy during fine-tuning of LLMs. Our approach offers high privacy guarantees, minimal communication overhead, and manageable computational demands. We utilize statistical data of the local gradients to train an AutoEncoder, preventing reverse attacks from the AutoEncoder on raw gradients as well as from raw gradients to the original input

data. In addition, we apply Gaussian noise to compressed gradients to ensure user-level differential privacy. We quantify privacy loss using GDP and RDP, offering a thorough privacy analysis. We propose a one-stage fine-tuning algorithm by integrating AutoEncoder training with LLM fine-tuning into a cohesive system. Moreover, our method can be extended to the Large Vision Model (LVM) and Large Multimodal Model (LMM) in the future.

Acknowledgments

This work is supported by the **Zhejiang Province Key Research and Development Plan** (Grant No. 2024SSYS0010) and the **National Natural Science Foundation of China** (Grant No. NSFC62132008).

References

Abrahamyan, L.; Chen, Y.; Bekoulis, G.; and Deligiannis, N. 2021. Learned gradient compression for distributed deep learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Balle, B.; and Wang, Y.-X. 2018. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. *arXiv preprint arXiv:1805.06530*.

Behnia, R.; Ebrahimi, M. R.; Pacheco, J.; and Padmanabhan, B. 2022. EW-Tune: A Framework for Privately Fine-Tuning Large Language Models with Differential Privacy. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*, 560–566.

Bu, Z.; Dong, J.; Long, Q.; and Su, W. J. 2019. Deep learning with Gaussian differential privacy. *arXiv preprint arXiv:1911.11607*.

Charles, Z.; Ganesh, A.; McKenna, R.; McMahan, H. B.; Mitchell, N.; Pillutla, K.; and Rush, K. 2024. Fine-Tuning Large Language Models with User-Level Differential Privacy. *arXiv preprint arXiv:2407.07737*.

Conover, M.; Hayes, M.; Mathur, A.; Xie, J.; Wan, J.; Shah, S.; Ghodsi, A.; Wendell, P.; Zaharia, M.; and Xin, R. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. *Company Blog of Databricks*.

Dettmers, T.; Pagnoni, A.; Holtzman, A.; and Zettlemoyer, L. 2024. Qlora: Efficient finetuning of quantized llms.

Dong, J.; Roth, A.; and Su, W. J. 2019. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4): 211–407.

Feng, Q.; Kasa, S. R.; Yun, H.; Teo, C. H.; and Bodapati, S. B. 2024. Exposing Privacy Gaps: Membership Inference Attack on Preference Data for LLM Alignment. arXiv:2407.06443.

Fredrikson, M.; Jha, S.; and Ristenpart, T. 2015. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, 1322–1333. New York, NY, USA: Association for Computing Machinery. ISBN 9781450338325.

Fu, Z.; Yang, H.; So, A. M.-C.; Lam, W.; Bing, L.; and Collier, N. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 12799–12807.

Guo, D.; Rush, A.; and Kim, Y. 2021. Parameter-Efficient Transfer Learning with Diff Pruning. In *ACL*.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. *NeurIPS*, 28.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2020. Measuring massive multitask language understanding. In *ICLR*.

Herde, M.; Raonić, B.; Rohner, T.; Käppeli, R.; Molinaro, R.; de Bézenac, E.; and Mishra, S. 2024. Poseidon: Efficient Foundation Models for PDEs. arXiv:2405.19101.

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. In *ICLR*.

Imani, S.; Du, L.; and Shrivastava, H. 2023. Mathprompter: Mathematical reasoning using large language models. In *ACL*.

Karanikolas, N.; Manga, E.; Samaridi, N.; Tousidou, E.; and Vassilakopoulos, M. 2023. Large Language Models versus Natural Language Understanding and Generation. In *Proceedings of the 27th Pan-Hellenic Conference on Progress in Computing and Informatics*.

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.

Lagunas, F.; Charlaix, E.; Sanh, V.; and Rush, A. M. 2021. Block Pruning For Faster Transformers. In *EMNLP*, 10619–10629.

Lee, C.; Cho, K.; and Kang, W. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models.

Li, H.; and Han, T. 2019. An end-to-end encrypted neural network for gradient updates transmission in federated learning. In *Data Compression Conference (DCC)*.

Lin, Y.; Han, S.; Mao, H.; Wang, Y.; and Dally, W. J. 2018. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *ICLR*.

Liu, S.-Y.; Wang, C.-Y.; Yin, H.; Molchanov, P.; Wang, Y.-C. F.; Cheng, K.-T.; and Chen, M.-H. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. In *ICML*.

Mahabadi, R. K.; Ruder, S.; Dehghani, M.; and Henderson, J. 2021. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*.

Mai, P.; Yan, R.; Huang, Z.; Yang, Y.; and Pang, Y. 2023. Split-and-Denoise: Protect large language model inference with local differential privacy. *arXiv preprint arXiv:2310.09130*.

Maini, P.; Jia, H.; Papernot, N.; and Dziedzic, A. 2024. LLM Dataset Inference: Did you train on my dataset? arXiv:2406.06443.

Mallya, A.; Davis, D.; and Lazebnik, S. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 67–82.

Mironov, I. 2017. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 263–275. IEEE.

Mostafa, H.; and Wang, X. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *ICML*, 4646–4655. PMLR.

Petrov, I.; Dimitrov, D. I.; Baader, M.; Müller, M. N.; and Vechev, M. 2024. DAGER: Exact Gradient Inversion for Large Language Models. arXiv:2405.15586.

Shi, W.; Shea, R.; Chen, S.; Zhang, C.; Jia, R.; and Yu, Z. 2022. Just fine-tune twice: Selective differential privacy for large language models. *arXiv preprint arXiv:2204.07667*.

Singh, T.; Aditya, H.; Madisetti, V. K.; and Bahga, A. 2024. Whispered tuning: Data privacy preservation in fine-tuning llms through differential privacy. *Journal of Software Engineering and Applications*, 17(1): 1–22.

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

Wei, J. T.-Z.; Wang, R. Y.; and Jia, R. 2024. Proving membership in LLM pretraining data via data watermarks. arXiv:2402.10892.

Wu, H.; Li, X.; Zhang, D.; Xu, X.; Wu, J.; Zhao, P.; and Liu, Z. 2024. CG-FedLLM: How to Compress Gradients in Federated Fune-tuning for Large Language Models. arXiv:2405.13746.

Xu, R.; Luo, F.; Zhang, Z.; Tan, C.; Chang, B.; Huang, S.; and Huang, F. 2021. Raise a Child in Large Language Model: Towards Effective and Generalizable Fine-tuning. In *EMNLP*, 9514–9528.

Zhang, J.; Kuo, M.; Zhang, R.; Wang, G.; Vahidian, S.; and Chen, Y. 2023a. Shepherd: A Lightweight GitHub Platform Supporting Federated Instruction Tuning. https://github.com/JayZhang42/FederatedGPT-Shepherd.

Zhang, J.; Vahidian, S.; Kuo, M.; Li, C.; Zhang, R.; Wang, G.; and Chen, Y. 2023b. Towards Building the Federated GPT: Federated Instruction Tuning. arXiv:2305.05644.