

## A APPENDIX

### A.1 BASELINES

We compare our proposed method Delta-LoRA with Fine-Tuning and prior works of LoRA, AdaLoRA, and DyLoRA. For PEFT methods, we only train the incremental updates for  $\mathbf{W}_V$  and  $\mathbf{W}_Q$ , following the setup as used in LoRA’s paper. For Fine-Tuning methods, we use two extra training paradigms: (1) freeze the embedding and train all the other parameters as Fine-Tuning †; (2) train  $\mathbf{W}_V$  and  $\mathbf{W}_Q$  only as Fine-Tuning ‡.

**Fine-Tuning.** In the past few years, fine-tuning has become the mainstream paradigm for both NLP and CV tasks. However, fine-tuning full parameters is subject to potential drawbacks including overfitting and training instability (Huang et al., 2022). Therefore, freezing a subset of network layers and fine-tuning the rest has become a popular choice (Tan et al., 2018). In our experiments, we compare with full fine-tuning, fine-tuning with embedding layers frozen (Fine-tuning †) and fine-tuning query and value matrices only (Fine-tuning ‡).

**LoRA** (Hu et al., 2022) uses multiplication of two low-rank matrices to learn the incremental updates with reduced GPU memory cost. We follow their setups to reproduce experimental results for fair comparison.

**DyLoRA** (Valipour et al., 2023) randomly chooses a rank  $r$  for LoRA modules during learning.

**AdaLoRA** (Zhang et al., 2022) focuses on the challenge of determining the optimal rank for incremental updates. It employs an adaptive approach to singular value pruning, tailoring the rank selection to the magnitude of each singular value. Consequently, distinct ranks are employed for different layers.

### A.2 THE COMPARISON BETWEEN LoRA AND DELTA-LoRA WITH LLaMA-7B

#### A.2.1 TRAINING AND INFERENCE ARGUMENTS USED IN OUR METHOD AND BASELINE

We choose LLaMA-7B to evaluate our method and LoRA. Here, we set the learning rate  $\gamma = 1e-4$ , batch size to 128,  $r = 8$ ,  $\alpha = 16$ , and training epochs to 3 for both two methods. Following the LoRA’s paper, we only tune  $\mathbf{W}_Q$  and  $\mathbf{W}_V$ . For Delta-LoRA, we choose start steps  $K = 100$  and  $\lambda = 0.25$ . When inference, we set the no\_repeat\_ngram\_size = 10, temperature = 0 and beam size = 4 to get a certain answer.

#### A.2.2 THE EVALUATION FOR OUR METHOD AND BASELINES

Current LLMs obtain the training data from the Internet, which may unintentionally cause data leakage. Therefore, using the mainstream NLP datasets to evaluate the effectiveness of Large Language Model is not reasonable and wisdom. Inspired by evaluation approach proposed by Liu et al. (2023), we decided to use GPT-4 to judge the text generated by which method is accurate. First, we ask GPT-4 to generate 1,000 different questions. Second, we use the LLaMA-7B trained by two methods to generate the texts. Finally, we ask GPT-4 to give decision to tell us which text is accurate. It can choose from three options: *a. Choice 1* (LoRA generates accurate text), *b. Choice 2* (Delta-LoRA generates accurate text) and *c. Both Choice 1 and 2* (Both LoRA and Delta-LoRA generate accurate texts). The prompt we used for evaluation:

*Help me to determine which text is accurate for the given instruction and question. The answer can be chosen from a. Choice 1 is accurate, b. Choice 2 is accurate or c. both Choice 1 and 2 are accurate. Give me a certain answer and this is a choice question. Please don’t give reasons and the answer must be shorter than 20 words.*

*Question: ""*

*(Choice 1): ""*

*(Choice 2): ""*

**Algorithm 1:** Delta-LoRA

---

**Input:** Learning rate  $\eta$ ; weight decay  $\beta$ ; total training iterations  $T$ ; low rank  $r$ ; scale factor  $\alpha$ ; start steps  $K$ ; update ratio  $\lambda$ .  
 $\mathbf{A}$  is initialized by Kaiming Initialization,  $\mathbf{B} = \mathbf{0}$  and  $\mathbf{W}$  is initialized with pre-trained weights.  
**for**  $t = 0, \dots, T - 1$  **do**  
    Sample a mini-batch and compute gradients for  $\{\mathbf{A}, \mathbf{B}\}$  in each Delta-LoRA module.  
    Update the first and second moments maintained by the optimizer with the computed gradients, and get the normalized gradients  $\hat{g}_\mathbf{A}$  and  $\hat{g}_\mathbf{B}$ .  
     $\mathbf{A}^{(t+1)} \leftarrow \mathbf{A}^{(t)} - \eta \hat{g}_\mathbf{A} - \eta \beta \mathbf{A}^{(t)}$   
     $\mathbf{B}^{(t+1)} \leftarrow \mathbf{B}^{(t)} - \eta \hat{g}_\mathbf{B} - \eta \beta \mathbf{B}^{(t)}$   
    **if**  $t > K$  **do**  
         $\mathbf{W}^{(t+1)} \leftarrow \mathbf{W}^{(t)} + \lambda \cdot \frac{\alpha}{r} \cdot (\mathbf{A}^{(t+1)} \mathbf{B}^{(t+1)} - \mathbf{A}^{(t)} \mathbf{B}^{(t)})$   
    **end if**  
**end for**  
**Output:** the fine-tuned parameters  $\{\mathbf{W}^{(T)}, \mathbf{A}^{(T)}, \mathbf{B}^{(T)}\}$

---

## A.3 ALGORITHM OF DELTA-LoRA

A.4 THE EXPANSION OF  $\Delta \mathbf{AB}$ 

In the real training process, we need to consider a variety of training arguments, such as optimizer and the regularization for  $\Delta \mathbf{AB}$ . Suppose that we use the AdamW (Loshchilov & Hutter, 2019) and  $L_2$  regularization, the  $\Delta \mathbf{AB}$  can be expanded in the following equation:

$$\begin{aligned}
\Delta \mathbf{AB} &= \mathbf{A}^{(t+1)} \mathbf{B}^{(t+1)} - \mathbf{A}^{(t)} \mathbf{B}^{(t)} \\
&= (\mathbf{A}^{(t)} - \eta \hat{g}_\mathbf{A} - \eta \beta \mathbf{A}^{(t)}) \cdot (\mathbf{B}^{(t)} - \eta \hat{g}_\mathbf{B} - \eta \beta \mathbf{B}^{(t)}) - \mathbf{A}^{(t)} \mathbf{B}^{(t)} \\
&= \mathbf{A}^{(t)} \mathbf{B}^{(t)} - \eta \mathbf{A}^{(t)} \hat{g}_\mathbf{B} - \eta \beta \mathbf{A}^{(t)} \mathbf{B}^{(t)} - \eta \hat{g}_\mathbf{A} \mathbf{B}^{(t)} + \eta^2 \hat{g}_\mathbf{A} \hat{g}_\mathbf{B} + \eta^2 \beta \hat{g}_\mathbf{A} \mathbf{B}^{(t)} \\
&\quad - \eta \beta \mathbf{A}^{(t)} \mathbf{B}^{(t)} + \eta^2 \beta \mathbf{A}^{(t)} \hat{g}_\mathbf{B} + \eta^2 \beta^2 \mathbf{A}^{(t)} \mathbf{B}^{(t)} - \mathbf{A}^{(t)} \mathbf{B}^{(t)} \\
&= -\eta \mathbf{A}^{(t)} \hat{g}_\mathbf{B} - \eta \beta \mathbf{A}^{(t)} \mathbf{B}^{(t)} - \eta \hat{g}_\mathbf{A} \mathbf{B}^{(t)} + \eta^2 \hat{g}_\mathbf{A} \hat{g}_\mathbf{B} + \eta^2 \beta \hat{g}_\mathbf{A} \mathbf{B}^{(t)} \\
&\quad - \eta \beta \mathbf{A}^{(t)} \mathbf{B}^{(t)} + \eta^2 \beta \mathbf{A}^{(t)} \hat{g}_\mathbf{B} + \eta^2 \beta^2 \mathbf{A}^{(t)} \mathbf{B}^{(t)} \\
&\approx -\eta \mathbf{A}^{(t)} \hat{g}_\mathbf{B} - \eta \hat{g}_\mathbf{A} \mathbf{B}^{(t)}
\end{aligned} \tag{8}$$

where  $\eta$  is the learning rate,  $\beta$  is weight decay. What's more, for pre-trained weight  $\mathbf{W}$ ,  $\Delta \mathbf{W} = \eta \hat{g}_\mathbf{W} + \eta \beta \mathbf{W}^{(t)}$ . As a consequence,  $\Delta \mathbf{AB}$  is not equal to  $\Delta \mathbf{W}$  in the training process.

## A.5 THE PARAMETER SENSITIVITY STUDY

Table 8: The parameter sensitivity study of update ratio  $\lambda$  for our proposed Delta-LoRA on E2E Challenge dataset. The best results are **boldfaced**.

$\lambda$	BLEU	NIST	METEOR	ROUGE-L	CIDEr
0	68.94	8.73	45.27	70.81	2.41
1	69.77	8.81	45.99	71.58	2.46
2	<b>70.84</b>	<b>8.91</b>	<b>46.47</b>	<b>72.24</b>	<b>2.53</b>
3	70.14	8.84	46.39	71.45	2.45
4	70.03	8.83	46.21	71.56	2.47
5	70.13	8.85	46.35	71.72	2.48

**Parameter Sensitivity.** Here, we explore the hyper-parameter  $K$  in Algorithm 1 and  $\lambda$  in Equation 6. For the hyper-parameter  $K$ , we select it from 0 to 1000 with the interval of 100. From Table 9, we find that our Delta-LoRA could not bring in any improvement before  $K = 400$ , and it will keep a relatively good performance when  $K$  is larger than 500. What is more, we choose different numbers for  $\lambda$ , ranging from 0 to 5. According to Table 8, the 5 metrics rise rapidly after  $\lambda = 0$  and reach best at  $\lambda = 2$ , while the performance has small drops on 5 evaluation scores if  $\lambda$  is chosen from 3 to 5.

Table 9: The parameter sensitivity study of start steps  $K$  for our proposed Delta-LoRA on E2E Challenge dataset. The best results are **boldfaced**.

$K$	BLEU	NIST	METEOR	ROUGE-L	CIDEr
0	69.10	8.75	45.54	71.31	2.41
100	69.97	8.84	46.07	71.40	2.46
200	69.72	8.83	45.82	71.41	2.43
300	69.73	8.86	45.98	71.09	2.46
400	70.18	8.89	46.30	71.66	2.49
500	70.84	8.91	46.47	<b>72.24</b>	<b>2.53</b>
600	70.38	8.86	46.38	71.70	2.47
700	70.61	8.89	46.43	72.13	2.51
800	70.70	8.89	46.30	71.97	2.51
900	<b>71.00</b>	<b>8.92</b>	<b>46.47</b>	72.04	2.52
1000	70.87	8.89	46.31	72.06	2.50

#### A.6 HYPER-PARAMETER USED IN OUR EXPERIMENTS

We report the hyper-parameter that used in our experiments. Table 10 and Table 11 show the hyper-parameter that we used for the training and evaluation on E2E Challenge and WebNLG Challenge 2017 dataset. The Table 12 and Table 13 are the training and evaluation hyper parameter for XSum dataset, and the Table 14 consists of hyper-parameters for 8 datasets in GLUE benchmark.

Table 10: The training hyper-parameter used for E2E Challenge and WebNLG Challenge 2017 dataset.

Hyper-Parameter	E2E Challenge	WebNLG Challenge 2017
Learning Rate $\eta$	2e-4	2e-4
Batch Size	8	8
Number of Epochs	5	5
Weight Decay $\beta$	0.01	0.01
Resid_pdrop	0	0.09
Attn_pdrop	0	0.09
Embd_pdrop	0	0
Label Smooth	0	0
Start Steps $K$	500	500
Update Ratio $\lambda$	2	5
Rank $r$	4	4
Alpha $\alpha$	32	32
Trainable Matrices	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$
LR Scheduler	Linear	Linear
Warmup Steps	500	500

Table 11: The hyper-parameter for evaluation used for E2E Challenge and WebNLG Challenge 2017 dataset.

Hyper-Parameter	E2E Challenge	WebNLG Challenge 2017
Beam Size	10	5
Penalty	0.8	1.0
No Repeat Ngram Size	4	4

Table 12: The training hyper-parameter used for XSum dataset.

Hyper-Parameter	Xsum
Learning Rate $\eta$	2e-4
Batch Size	64
Number of Epochs	25
Weight Decay $\beta$	0
Activation Dropout	0
Dropout	0
Classifier Dropout	0
Start Steps $K$	1000
Update Ratio $\lambda$	0.5
Rank $r$	4
Alpha $\alpha$	32
Trainable Matrices	$\mathbf{W}_Q, \mathbf{W}_V$
LR Scheduler	Linear
Warmup Steps	3000

Table 13: The hyper-parameter for evaluation used for XSum dataset.

Hyper-Parameter	Xsum
Beam Size	8
Penalty	1.0
No Repeat N-gram Size	4

Table 14: The training hyper-parameters of our proposed Delta-LoRA on GLUE benchmark. We adopt the most of hyper-parameters in LoRA’s paper and implement our method based on the codes given by LoRA’s repository.

Hyper-Parameter	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Learning Rate $\eta$	5e-4	5e-4	4e-4	4e-4	4e-4	4e-4	4e-4	4e-4
Batch Size	128	128	128	64	128	128	128	128
Number of Epochs	30	60	30	80	25	25	80	40
Weight Decay $\beta$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Max Sequence Length	256	256	256	256	256	256	512	256
Start Steps $K$	2000	400	10	100	800	400	200	200
Update Ratio $\lambda$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Rank $r$	8	8	8	8	8	8	8	8
Alpha $\alpha$	16	16	16	16	16	16	16	16
LR Scheduler	Linear	Linear	Linear	Linear	Linear	Linear	Linear	Linear
Trainable Matrices	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$	$\mathbf{W}_Q, \mathbf{W}_V$
Warmup Ratio	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.06
Evaluation Metrics	Accuracy	Accuracy	Accuracy	Matthews Correlation	Accuracy	Accuracy	Accuracy	Pearson