Supplementary Material for "Sensitivity-Aware Efficient Fine-Tuning via Compact Dynamic-Rank Adaptation"

Model & Method	# Trainable Parameters	SST-2 (Acc.)	MRPC (Acc.)	CoLA (MCC)	QNLI (Acc.)	RTE (Acc.)	STS-B (PCC)	Avg.
RoB_{base}(FF)	125M	94.8	90.2	63.6	92.8	78.7	91.2	85.2
RoB_{base}(BitFit)	0.1M	93.7	**92.7**	62	91.8	81.5	90.8	85.4
RoB_{base}(Adpt^{D})	0.3M	$94.2_{\pm 0.1}$	$88.5_{\pm 1.1}$	$60.8_{\pm 0.4}$	$93.1_{\pm 0.1}$	$71.5_{\pm 2.7}$	$89.7_{\pm 0.3}$	83.0
RoB_{base}(Adpt^{D})	0.9M	$94.7_{\pm 0.3}$	$88.4_{\pm 0.1}$	$62.6_{\pm 0.9}$	$93.0_{\pm 0.2}$	$75.9_{\pm 2.2}$	$90.3_{\pm 0.1}$	84.2
RoB_{base}(LoRA)	0.3M	$95.1_{\pm 0.2}$	$89.7_{\pm 0.7}$	$63.4_{\pm 1.2}$	$\mathbf{93.3}_{\pm 0.3}$	$78.4_{\pm 0.8}$	$\mathbf{91.5}_{\pm 0.2}$	85.2
RoB_{base}(AdaLoRA)	0.3M	$94.5_{\pm 0.2}$	$88.7_{\pm 0.5}$	$62.0_{\pm 0.6}$	$93.1_{\pm 0.2}$	$81.0_{\pm 0.6}$	$90.5_{\pm 0.2}$	85.0
RoB_{base}(DyLoRA)	0.3M	$94.3_{\pm 0.5}$	$89.5_{\pm 0.5}$	$61.1_{\pm 0.3}$	$92.2_{\pm 0.5}$	$78.7_{\pm 0.7}$	$91.1_{\pm 0.6}$	84.5
RoB_{base}(**OURS**)	0.4M	$\mathbf{95.8}_{\pm 0.3}$	$90.2_{\pm 0.3}$	$\mathbf{66.5}_{\pm 0.5}$	$92.9_{\pm 0.1}$	$\mathbf{82.3}_{\pm 0.5}$	$91.2_{\pm 0.2}$	**86.4**
$\text{RoB}_{\text{large}}$(FF)	356M	96.4	**90.9**	68.0	94.7	86.6	**92.4**	88.2
$\text{RoB}_{\text{large}}$($\text{Adpt}^{\text{P}}$)	3M	$96.1_{\pm 0.3}$	$90.2_{\pm 0.7}$	$68.3_{\pm 1.0}$	$\mathbf{94.8}_{\pm 0.2}$	$83.8_{\pm 2.9}$	$92.1_{\pm 0.7}$	87.6
$\text{RoB}_{\text{large}}$($\text{Adpt}^{\text{P}}$)	0.8M	$\mathbf{96.6}_{\pm 0.2}$	$89.7_{\pm 1.2}$	$67.8_{\pm 2.5}$	$\mathbf{94.8}_{\pm 0.3}$	$80.1_{\pm 2.9}$	$91.9_{\pm 0.4}$	86.8
$\text{RoB}_{\text{large}}$($\text{Adpt}^{\text{H}}$)	6M	$96.2_{\pm 0.3}$	$88.7_{\pm 2.9}$	$66.5_{\pm 4.4}$	$94.7_{\pm 0.2}$	$83.4_{\pm 1.1}$	$91.0_{\pm 1.7}$	86.8
$\text{RoB}_{\text{large}}$($\text{Adpt}^{\text{H}}$)	0.8M	$96.3_{\pm 0.5}$	$87.7_{\pm 1.7}$	$66.3_{\pm 2.0}$	$94.7_{\pm 0.2}$	$72.9_{\pm 2.9}$	$91.5_{\pm 0.5}$	84.9
$\text{RoB}_{\text{large}}$(LoRA)	0.8M	$96.2_{\pm 0.5}$	$90.2_{\pm 1.0}$	$68.2_{\pm 1.9}$	$\mathbf{94.8}_{\pm 0.3}$	$85.2_{\pm 1.1}$	$92.3_{\pm 0.5}$	87.8
$\text{RoB}_{\text{large}}$(**OURS**)	0.8M	$95.5_{\pm 0.2}$	$\mathbf{90.9}_{\pm 0.3}$	$\mathbf{68.5}_{\pm 0.5}$	$94.7_{\pm 0.4}$	$\mathbf{88.4}_{\pm 1.6}$	$\mathbf{92.4}_{\pm 0.4}$	**88.4**

Table 1. Performance on GLUE benchmark with RoBERTa Base and RoBERTa Large models. Best results are highlighted in **bold**.

1. More Experiment Results

1.1. Natural Language Understanding

We evaluate our method on the GLUE benchmark (General Language Understanding Evaluation), which consists of six datasets and can be grouped into three types: single-sentence classification tasks, similarity and paraphrase tasks and natural language inference tasks. Both RoBERTa Base and RoBERTa Large fine-tuned as backbone.

The results are reported in Table 1. We report the median of 5 random seed results, where the best epoch is selected for each run. We can see that, 1) our method obtains around $1\% \sim 3\%$ accuracy improvement of the mean accuracy compared with baseline methods with the similar or less trainable parameters on RoBERTa Base. 2) our method obtains around $0.2\% \sim 3\%$ accuracy improvement of the mean accuracy compared with baseline methods with the same or less trainable parameters on RoBERTa Large. 3) Notably, our method outperforms all baselines on RoBERTa Base of CoLA, SST-2 and RTE. And our method outperforms all baselines on RoBERTa Large of CoLA and RTE.

1.2. Storage Burden On GLUE Benchmark

In Table 2, To further demonstrate the computational cost advantage of our CDRA-SPTWe provide training time and GPU memory metrics with the RoBERTa Base on the GLUE benchmark where we select LORA, and SPT as our comparison baselines. From the results, we observe that both LORA and SPT take more training time and higher GPU memory, which is because only fine-tuning submatrix of CDRA-SPT introduces less parameters.

Table 2. Anaysis of computational cost on GLUE.

Method	Training time (hour)	Fine-tuning Memory (GB)
LoRA	9.41	5.77
SPT-LoRA	9.63	5.79
Ours	**9.17**	**4.70**

1.3. The way of updating sensitive parameters

In Table 3, we compare our method with mask-tuning and full fine-tuning on sub-parameter matrix. From results, we find that our method obtains around 2.7% and 3.8% accuracy improvement of the mean accuracy. Because correlation between sensitive row and column vectors can be fully characterized by CDRA-SPT.

Table 3. Different ways of updating compact matrix on VTAB.

Method	Full-finetuning	Mask-tuning	Ours
MeanAcc.	73.0	74.1	**76.8**

1.4. The rank of compact sub-parameter matrix

We report the performance in increased ranks case in Table 4. The result shows that the compact sub-parameter matrix is more likely low rank matrix. A similar experiment conducted by previous work [1].

Table 4. Analysis of different rank adaptation on VTAB.

Rank	32	64	128	Full
MeanAcc.	75.4	75.4	75.0	74.2

1.5. Experiment on different sub-matrix sizes

In Table 5, we provide experiment on different sub-matrix sizes. From results, we can find that the performance is decreasing when the size of sub-matrix sizes is smaller. This experiment further prove the advantage of our method.

Table 5. Different sub-parameter matrix's size on VTAB.

Size	Ours($\times 1.0$)	$\times 0.5$	$\times 0.3$	$\times 0.1$	$\times 0.01$
MeanAcc.	**76.8**	74.6	74.1	73.8	73.5

2. Baselines Description

Full Fine-tuning (FF): During fine-tuning, updating all the parameters on the base model which is initialized with pre-trained weights and biases.

Bitfit: Only updating the bias vectors during fine-tuning.
Adapter: **Adapter$^{\mathbf{H}}$** firstly adds trainable modules between the self-attention and the FNN modules, which consist of two-layers followed by a subsequent residual connection. Unlike the **Adapter$^{\mathbf{H}}$**, **Adapter$^{\mathbf{P}}$** inserts the adapter layers after the feed-forward layer. Motivated by some adapter layers that are not activated, **Adapter$^{\mathbf{D}}$** drop them and utilize the actual efficient parameters.
LoRA: Introduce two low-rank matrices to update incremental weight, which can be merged into original matrix.
DyLoRA:[2] For finding the best rank choice, this method trains a dynamic search-free LoRA models.
AdaLoRA:[3] Proposing dynamic rank adaption based on the singular value decomposition and use importance-aware rank allocation to prune redundant singular values.
Partialft-k: Only updating last k layers and the linear classification head while other parameters are freezing.
Mlp-k: Updating a trainable k-layer multi-layer perceptron as classification head while other parameters are freezing.
ShallowPrompt: Only introduce trainable prompts to the input space of the pretrained ViT.
DeepPrompt: Introduce additional trainable prompts to the sequence in the MHA of each ViT block.
Gradient Based Selection(GPS): Selecting top-k parameters of neuron based on gradient before mask tuning.

References

[1] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020. 1

[2] Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3274–3287, 2023. 2

[3] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview, 2023. 2