LoRA-drop: Efficient LoRA Parameter Pruning based on Output Evaluation

Anonymous ACL submission

Abstract

Low-Rank Adaptation (LoRA) introduces auxiliary parameters for each layer to fine-tune the pre-trained model under limited computing resources. But it still faces challenges of resource consumption when scaling up to larger models. Previous studies employ pruning techniques by evaluating the importance of LoRA parameters for different layers to address the problem. However, these efforts only analyzed parameter features to evaluate their importance. Indeed, the output of LoRA related to the parameters and data is the factor that directly impacts the frozen model. To this end, we propose LoRA-drop which evaluates the importance of the parameters by analyzing the LoRA output. We retain LoRA for important layers and the LoRA of the other layers share the same parameters. Abundant experiments on NLU and NLG tasks demonstrate the effectiveness of LoRA-drop.

1 Introduction

Parameter-efficient fine-tuning (PEFT) methods have attracted more and more attention with the development of large language models (LLM) (Li and Liang, 2021a; Lester et al., 2021a). Among various PEFT methods, LoRA (Hu et al., 2021) has been particularly prevalent in recent studies. LoRA freezes the pre-trained parameters and introduces auxiliary trainable parameters $\Delta \boldsymbol{W}$ for each layer as shown in Figure 1. LoRA significantly reduces the training cost while achieving impressive results.

To further improve the parameter efficiency of LoRA, previous studies employ pruning techniques by evaluating the importance of LoRA parameters for different layers. Sparse Adapter (He et al., 2022) evaluates the importance of LoRA based on different parameter features such as parameter count, parameter size, and parameter gradient. AdaLoRA (Zhang et al., 2022) designs importance criteria based on the singular value decomposition (SVD) of $\Delta \boldsymbol{W}$ to prune unimportant singular values. All of these efforts only focused on analyzing parameter features to evaluate their importance.

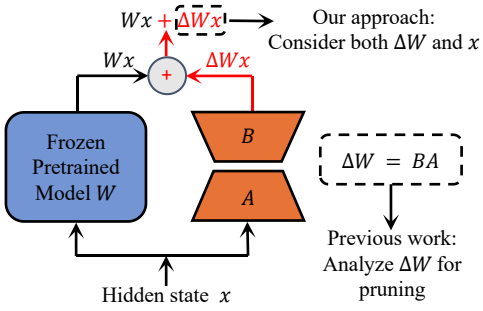

Figure 1: The diagram of LoRA

Indeed, the output of LoRA related to the parameters and data is the factor that directly impacts the frozen model. In each layer of the pre-trained model, the LoRA adds a bias term $\Delta \boldsymbol{W}\boldsymbol{x}$ to the frozen model. Intuitively, if the norm of $\Delta \boldsymbol{W}\boldsymbol{x}$ is large, the LoRA of this layer has an important impact on the frozen model.

Our preliminary experiment shows that the LoRA for some layers always has little impact on a specific task. Thus we could prune these LoRA parameters. To this end, we propose LoRA-drop which evaluates the importance of parameters by analyzing the LoRA output for each layer. First, we sample specific task datasets and then utilize the sampled data to perform a limited number of updates to LoRA. The importance of LoRA at various layers is determined based on $\Delta \boldsymbol{W}\boldsymbol{x}$. We retain the LoRA of layers with a large importance score, and the LoRA of the other layers share the same parameter. Finally, We fine-tune the model under the new LoRA setting.

We conducted comprehensive experiments on multiple NLU and NLG tasks with different sizes of the pre-trained model, showing that LoRA-drop achieves comparable results with origin LoRA with 50% of LoRA parameters. Analysis experiments demonstrate the effectiveness of LoRA-drop.

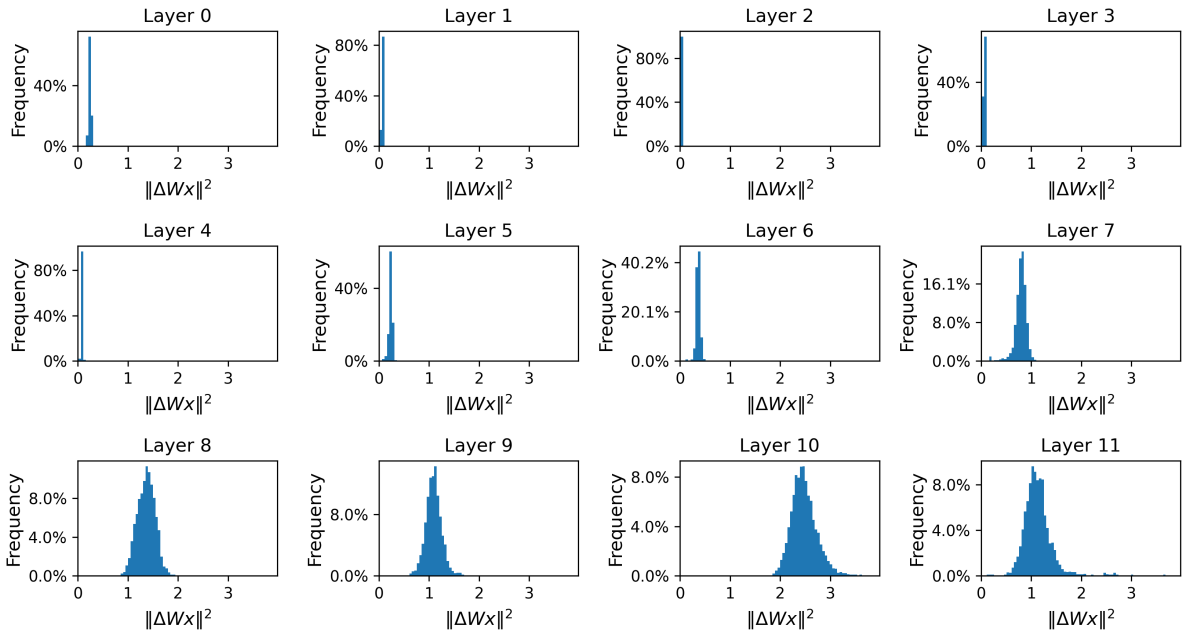

Figure 2: The frequency distribution of the squared norm of query LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ on the RTE task.

2 Preliminary Experiment

LoRA utilizes the product of two low-rank matrices to simulate incremental updates to a full-rank matrix. During training, the pre-trained parameters are frozen and do not receive gradient updates, while the two low-rank matrices are trained. Let $\boldsymbol{W}_i$ denote the query/key/value matrix of ith transformer layer and $\boldsymbol{x}_i$ denote the input of the ith transformer. The two low-rank matrices are $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$. Thus, the query/key/value vector is as following:

$$\boldsymbol{h}_i = \boldsymbol{W}_i \boldsymbol{x}_i + \Delta \boldsymbol{W}_i \boldsymbol{x}_i = \boldsymbol{W}_i \boldsymbol{x}_i + \boldsymbol{B}_i \boldsymbol{A}_i \boldsymbol{x}_i \quad (1)$$

where $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ is the bias introduced by the LoRA paramerters.

Indeed, the $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ is the factor that directly influences the frozen pre-trained model. The larger $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$, the more important the LoRA is. However, the $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ is related to the LoRA parameter and the hidden state. Previous work prunes LoRA by only analyzing its parameters, ignoring the feature of the hidden state.

We fine-tune the RoBERTa-base model with LoRA on the RTE task and statistic the distribution of the squared norm of the LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$. Following the setting of (Hu et al., 2021), the LoRA is added to the query and value matrix. The distribution of query LoRA is shown in Figure 2. The distribution of value LoRA is shown in Figure 4.

As seen, the squared norm of $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ of some layers is always small, indicating that these layers have almost no impact on the frozen model. Thus we could prune the LoRA to improve the paramerter efficiency according to $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$.

3 Methodology

In this section, we introduce LoRA-drop, a novel parameter-efficient fine-tuning method by pruning based on LoRA output. Concretely, we perform stratified sampling on the downstream task dataset to obtain a subset of training data. The sampling ratio is set to α, where $0<\alpha<1$. Then the LoRA parameters are updated with a limited number of steps using this subset.

Then we compute the squared norm of LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ for each layer. The squared norms are averaged over training dataset to get the importance score of ith layer $\boldsymbol{g}_i$. To indicate the relative importance, we normalize the score $\boldsymbol{y}_i = \frac{\boldsymbol{g}_i}{\sum_i \boldsymbol{g}_i}$.

After obtaining the importance of each layer, we sort the layers according to $\boldsymbol{y}_i$. We select the layers from the most important to least important until the sum importance of the selected layer reaches a threshold T. The LoRA of these selected layers will be retained during training, while the LoRA of the other layers will be replaced by a shared parameter. The hyper-parameter T controls the number of the selected layers.

Finally, we fine-tune the model under the new LoRA setting using the training dataset. The overall workflow of LoRA-drop is shown in Figure 3.

Model RoBERTa-base	#Tr. Params	RTE (Acc)	MRPC (Acc)	STS-B (Spea.)	CoLA (Matt.)	SST-2 (Acc)	QNLI (Acc)	MNLI (Acc)	QQP (Acc)	Avg.
Full-FT*	125M	78.7	**90.2**	**91.2**	**63.6**	**94.8**	92.8	**87.6**	**91.8**	**86.3**
LoRA	0.29M	$\underline{80.8}_{\pm 1.5}$	$89.1_{\pm 0.6}$	$\mathbf{91.2}_{\pm 0.1}$	$62.4_{\pm 0.7}$	$94.3_{\pm 0.3}$	$\underline{93.0}_{\pm 0.2}$	$\underline{87.5}_{\pm 0.2}$	$\underline{90.3}_{\pm 0.1}$	86.1
Sparse Adapter	0.15M	$78.7_{\pm 1.1}$	$88.0_{\pm 0.5}$	$89.5_{\pm 0.4}$	$60.1_{\pm 0.7}$	$94.1_{\pm 0.1}$	$92.8_{\pm 0.1}$	$87.1_{\pm 0.2}$	$89.6_{\pm 0.1}$	85.0
VeRA	0.03M	$78.0_{\pm 1.1}$	$88.4_{\pm 0.1}$	$89.8_{\pm 0.2}$	$60.9_{\pm 0.5}$	$93.7_{\pm 0.1}$	$89.6_{\pm 0.1}$	$83.7_{\pm 0.1}$	$86.8_{\pm 0.1}$	83.9
Tied-LoRA	0.15M	$80.0_{\pm 0.9}$	$89.1_{\pm 0.6}$	$90.3_{\pm 0.1}$	$62.0_{\pm 0.8}$	$94.1_{\pm 0.3}$	$91.6_{\pm 0.4}$	$86.9_{\pm 0.1}$	$89.7_{\pm 0.1}$	85.5
LoRA-drop (ours)	0.15M	$\mathbf{81.4}_{\pm 0.5}$	$\underline{89.5}_{\pm 0.5}$	$\underline{91.0}_{\pm 0.1}$	$\underline{62.9}_{\pm 0.2}$	$\underline{94.5}_{\pm 0.2}$	$\mathbf{93.1}_{\pm 0.1}$	$87.3_{\pm 0.2}$	$90.1_{\pm 0.1}$	$\underline{86.2}$

Table 1: Results of the RoBERTa-base with different training strategies on the GLUE benchmark. The results are averaged from three seeds to produce solid results. The subscript is the standard deviation. Bold and underline indicate the first and second best results in the corresponding regime. #Tr. refers to trainable. * refers to the results directly from their original paper, in which Full-FT is derived from (Liu et al., 2019).

4 Experiments

4.1 Setup

Datasets. We evaluate our model on both Natural Language Understanding (NLU) and Natural Language Generation (NLG) tasks.

For NLU, we evaluate our method on the GLUE benchmark (Wang et al., 2018), which consists of eight datasets: CoLA, SST-2, MRPC, QQP, STS-B, MNLI, QNLI, and RTE. We use Matthew's correlation coefficient, Spearman's correlation coefficient, and overall accuracy (for both matched and mismatched sentences) to evaluate the CoLA, STS-B, and MNLI datasets. For the remaining datasets, we apply accuracy as the evaluation metric.

The NLG tasks in our experiments include the table-to-text datasets E2E (Dušek et al., 2020) and DART (Nan et al., 2021), as well as the summarization dataset DialogSum (Chen et al., 2021). We evaluate both NLG tasks using BLEU (Papineni et al., 2002) and Rouge-L (Lin, 2004) scores.

Baselines. The following methods are chosen as baselines: **FULL-FT** updates all model parameters which need a lot of computing resources. **LoRA** (Hu et al., 2021) represents the original LoRA method. **Sparse Adapter** (He et al., 2022) apply typical pruning methods to LoRA and reduce the trainable parameters. **VeRA** (Kopiczko et al., 2023) shares and freezes randomly initialized LoRA and introduces trainable vectors for each layer to reduce the parameters of LoRA. **Tied-LoRA** (Renduchintala et al., 2023) makes the frozen LoRA in VeRA trainable.

More implementation details can be found in Section A.1.

4.2 Main Results

The main results of RoBERTa-base with different training strategies on the GLUE benchmark are shown in Table 1. With an approximately 50% reduction in standard LoRA parameters, our proposed LoRA-drop achieves an average score of 86.2, on par with Full-FT (86.3) and LoRA (86.1). This indicates the effectiveness of our proposed LoRA-drop.

Model Llama2 7b	#Tr. Params	Dialogsum BLEU	Dialogsum ROUGE-L
Full-FT	6.6B	**46.27**	**43.07**
LoRA	8.4M	44.60	41.12
LoRA-drop (ours)	5.2M	44.34	40.96

Table 2: Results of Llama2-7b with different training strategies on the summarization dataset Dialogsum.

Moreover, LoRA-drop achieves 1.2, 2.3, and 0.7 improvement in terms of average score compared to Sparse Adapter, VERA, and Tied-LoRA respectively. The results demonstrate that the output of LoRA is a superior strategy to evaluate the importance and reduce trainable parameters, thereby enhancing parameter efficiency.

The results of RoBERTa-large and Llama2-7b with different training strategies on the GLUE benchmark are presented in Table 3 and Table 4. It is noted that we use Llama2-7b to obtain the token representation, rather than generate the answer. On both models, our method utilizes about 52% of the standard LoRA parameters and achieves average scores of 89.1 and 88.6 for RoBERTa-large and Llama2-7b respectively, outperforming LoRA and Full-FT. This demonstrates the effectiveness of our method across models of different scales.

The NLG results including table2text and summarization are shown in Tables 5 and Table 2. On Llama2-7b, our method achieves results on par with the original LoRA while using approximately 50% and 62% of the original LoRA parameters for table2text and summarization respectively. This confirms the effectiveness of our approach across both NLU and NLG tasks.

4.3 Analysis

Adaptation to different downstream task data. The insight of our approach is to derive LoRA importance adapted to the distribution of different task data, enabling the simplification of LoRA parameters. To further validate the rationality of this insight, we plotted line charts depicting the distribution of LoRA importance $\boldsymbol{y}$ for four different datasets in GLUE on RoBERTa and Llama2.

The results are presented in Figure 9 and Figure 10. We observed that the importance distributions differed across datasets, indicating that the relative importance assigned by LoRA is data-dependent. This also demonstrates the rationale behind LoRA-drop, which calculates importance scores based on both the LoRA parameters and the LoRA output.

The influence of LoRA share. In our approach, the layers with low importance are shared with the same LoRA parameters. We investigate the influence after the LoRA parameters are shared. Following the LoRA share operation on the RoBERTa-base model trained on 20% of the RTE training set data for 4 epochs, we plotted the importance distribution for each layer of the model.

The results of query and value distribution are shown in Figure 7 and Figure 8. It shows that the importance distribution of LoRA at each layer remains almost consistent with the original LoRA after the LoRA parameters are shared. This suggests that the LoRA of layers with low importance could be shared. The importance distribution of other layers is not affected, thereby maintaining good performance in fine-tuning the model.

The influence of sample proportion. We investigate the influence of the sample proportion when calculating the importance of LoRA. We sample ten different-sized datasets from the RTE dataset. We train the RoBERTa-base model using LoRA for the same number of steps and obtain the LoRA importance scores for each sample proportion.

The results are shown in Figure 11. The relative importance order of each layer remains consistent across various sample proportions, indicating that this operation is insensitive to the size of the sampled data and exhibits robustness.

Selection of threshold T. LoRA-drop introduces a hyper-parameter T to control the number of selected layers. We selected four datasets from GLUE to analyze the impact of threshold T.

The results are shown in Table 6. When T is less than 0.9, the model performance increases with T, and when T equals 0.9, approximately half of the layers' LoRA are selected on average. If T continues to increase, the model performance no longer shows significant improvement. Hence in our experiments, we default to setting T as 0.9.

4.4 Ablation Study

In this subsection, we conduct ablation experiments to verify the following two questions: Q1: Is replacing LoRA at layers with small $\boldsymbol{y}$ with shared parameters better than directly removing them? Q2: Does retaining LoRA with large $\boldsymbol{y}$ is reasonable?

To demonstrate this, we compare LoRA-drop with the following variants on the RoBERTa-base model, where k refers to the number of LoRA retained by LoRA-drop.

LoRA-drop (w/o share) directly removes the low-importance layers of LoRA without using additional shared parameters. As opposed to LoRA-drop, **LoRA-drop ($\Delta \boldsymbol{W}\boldsymbol{x}$ inv)** replace high-importance layers of LoRA with shared LoRA and retain the other LoRA. **LoRA-drop (random)** randomly selects k layers that retain LoRA parameters. (Houlsby et al., 2019) found that lower layers often have small impact on performances, so **LoRA-drop (top k)** selects the top k layers of the 12 layer model. We experiment with these four settings on the validation set of the GLUE benchmark.

The results are shown in Table 7. Regarding Q1, we observed that sharing a LoRA among the layers with small importance is necessary, achieving better results compared to removing them directly.

Regarding Q2, LoRA-drop yields better performance compared to all the other three variants. It confirms the reasonableness for retaining the LoRA of layers with large importance.

5 Conclusion

In this paper, we propose a new parameter-efficient fine-tuning method LoRA-drop based on LoRA. We calculate the importance of LoRA for each layer by analyzing their output. The LoRA parameters of layers with large importance are retained and the other layers share the same parameter, resulting in a significant reduction in the number of parameters that need to be trained compared to the original LoRA. Abundant experiments on multiple NLU and NLG datasets show that LoRA-drop achieves comparable results with origin LoRA with 50% of LoRA parameters.

Limitations

The applicability of our method to different tasks and architectures is not extensively analyzed in this paper, it can be explored in future research by investigating the distribution patterns of LoRA importance when fine-tuning different pre-training models on various tasks and conducting in-depth analysis of the underlying mechanisms.

References

Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. 2021. DialogSum: A real-life scenario dialogue summarization dataset. In *Findings of the Association for Computational Linguistics*, pages 5062–5074.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Proceedings of the Conference on Neural Information Processing Systems*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, pages 123–156.

Anchun Gui and Han Xiao. 2023. HiFi: High-information attention heads hold for parameter-efficient model adaptation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 8521–8537.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. 2022. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of the Association for Computational Linguistics*, pages 2184–2190.

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

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

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *arXiv preprint arXiv:1911.03090*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S Torr. 2018. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*.

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

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

Xiang Lisa Li and Percy Liang. 2021a. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 4582–4597.

Xiang Lisa Li and Percy Liang. 2021b. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 4582–4597.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

Wei Liu, Ying Qin, Zhiyuan Peng, and Tan Lee. 2023. Sparsely shared lora on whisper for child speech recognition. *arXiv preprint arXiv:2309.11756*.

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

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher,

and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 432–447.

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

Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2023. Tied-lora: Enhacing parameter efficiency of lora with weight tying. *arXiv preprint arXiv:2311.09578*.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. 2021. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 7930–7946.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1–9.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2022. Adaptive budget allocation for parameter-efficient fine-tuning. In *Proceedings of the International Conference on Learning Representations*.

A Appendix

A.1 Implementation Details

Our LoRA configuration aligns with the experimental setup of (Hu et al., 2021), where LoRA is applied to the query and value matrices in each self-attention module. We each use a shared LoRA in place of the low importance query LoRA and value LoRA.

The low-rank matrix A of the LoRA architecture is initialized using Kaiming initialization (He et al., 2015), while matrix B is initialized with zeros. Unless specified otherwise, the default rank for LoRA is set to 8.

We conducted NLU experiments on the GLUE benchmark using RoBERTa-base (Liu et al., 2019). The data sampling ratio α is set to 0.1, the number of training epochs n is set to 3, and the threshold T for LoRA-drop is set to 0.9. To ensure consistency in the trainable parameter count between the baseline and our method, we set the sparsity rate of Sparse Adapter to 0.5. We set the pruning method of Sparse Adapter to the performance-optimal SNIP (Lee et al., 2018). The rank of Tied-LoRA is set to 56. The design characteristics of the VeRA method determine that its trainable parameter count cannot reach the same order of magnitude as LoRA; otherwise, VeRA would no longer be a low-rank matrix. Therefore, we set the rank of VeRA to 512 based on the best hyperparameters provided in the original paper.

To evaluate the effectiveness of our method on larger-scale models, we also conducted NLU experiments on the GLUE benchmark using the larger models RoBERTa-large (Liu et al., 2019) and Llama 7b (Touvron et al., 2023). We performed 3 runs with different random seeds and recorded the best results for different seeds. The averaged results and the standard deviation are calculated.

To evaluate the effectiveness of our method on generation tasks, we conducted NLG experiments using the Llama2 7b on the table2text datasets: E2E and DART, as well as the summarization dataset DialogSum. For the table2text tasks, we set the rank of LoRA to 8, while for the summarization task, we set the rank of LoRA to 16.

A.2 Related Work

Parameter-efficient fine-tuning (PEFT) is the mainstream method for the current fine-tuning of pre-trained models, which can be broadly categorized into addition-based methods and specification-based methods (Ding et al., 2022).

A.2.1 Addition-based Methods

Addition-based methods introduce additional trainable neural modules into pre-trained models. Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2021) are two of the most typical methods. Prefix-tuning (Li and Liang, 2021b) inserts trainable tokens into the input and hidden states of each Transformer layer. Prompt-tuning (Lester et al., 2021b) adds only a continuous learnable vector to the input layer.

To improve the parameter efficiency, many studies focus on reducing the trainable parameters of LoRA (Zhang et al., 2022; Dettmers et al., 2023). S2-LoRA (Liu et al., 2023) shares the LoRA parameters, and introduces trainable scaling vectors with inter-layer variations. VeRA (Kopiczko et al., 2023) and Tied-LoRA (Renduchintala et al., 2023), further reduce the parameter count by sharing parameters for all layers and modules of LoRA. Sparse adapter (He et al., 2022) enhances the parameter efficiency of LoRA and other Adapters using network pruning methods. Adapter Drop (Rücklé et al., 2021) empirically removes lower-layer Adapters that are generally considered to have a small impact on task performance.

A.2.2 Specification-based Methods

Specification-based methods make a small subset of parameters in the pre-trained model trainable while keeping the rest frozen. BitFit (Zaken et al., 2022)only fine-tunes the bias parameters of each FFN. (Lee et al., 2019) fine-tunes only the last quarter of BERT and RoBERTa's final layer, achieving 90% of the performance of full-parameter fine-tuning. HiFi (Gui and Xiao, 2023) fine-tunes attention heads that are highly informative and strongly correlated for a specific task.

A.3 More figures and tables

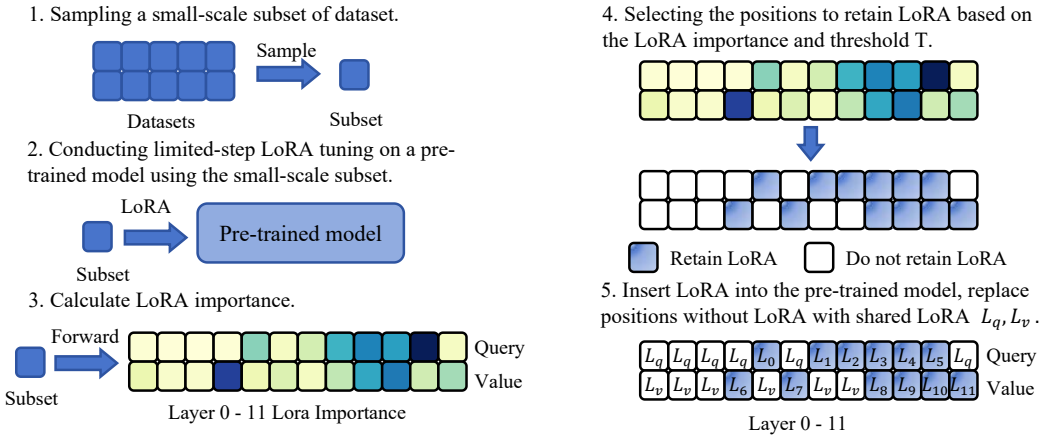

Figure 3: The overall workflow of LoRA-drop.

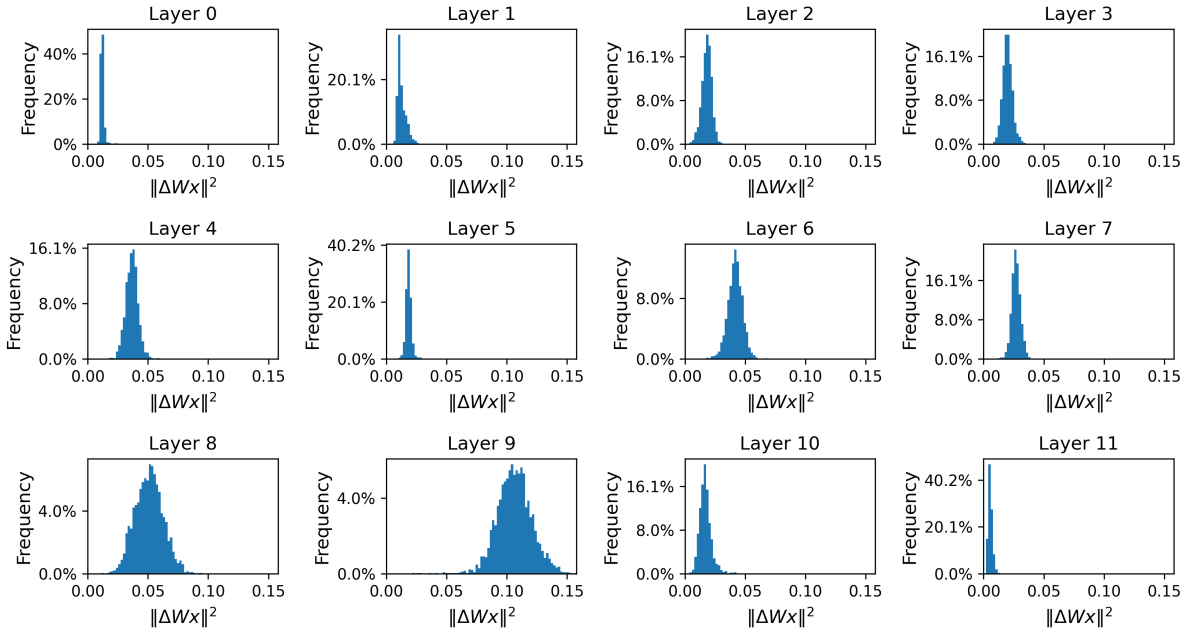

Figure 4: The frequency distribution of the squared norm of value LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ after fine-tuning on the RTE task.

Model RoB-large	#Tr. Params	RTE (Acc)	MRPC (Acc)	STS-B (Spea.)	CoLA (Matt.)	SST-2 (Acc)	QNLI (Acc)	MNLI (Acc)	QQP (Acc)	Avg.
Full-FT*	355M	86.6	**90.9**	**92.4**	68.0	**96.4**	94.7	90.2	**92.2**	88.9
LoRA	0.79M	$88.5_{\pm 0.7}$	$90.1_{\pm 0.8}$	$\mathbf{92.4}_{\pm 0.1}$	$67.8_{\pm 1.3}$	$96.0_{\pm 0.1}$	$94.8_{\pm 0.1}$	$90.6_{\pm 0.0}$	$91.4_{\pm 0.1}$	88.9
LoRA-drop (ours)	0.41M	$\mathbf{88.8}_{\pm 0.7}$	$89.9_{\pm 0.3}$	$92.2_{\pm 0.1}$	$\mathbf{68.5}_{\pm 1.7}$	$96.2_{\pm 0.1}$	$\mathbf{94.9}_{\pm 0.1}$	$\mathbf{90.7}_{\pm 0.1}$	$91.3_{\pm 0.5}$	**89.1**

Table 3: The performance of the RoBERTa-large on GLUE benchmark. * refers to the results directly from their original paper, in which Full-FT is derived from (Liu et al., 2019).

Model Llama2 7b	#Tr. Params	RTE (Acc)	MRPC (Acc)	STS-B (Spea.)	CoLA (Matt.)	SST-2 (Acc)	QNLI (Acc)	MNLI (Acc)	QQP (Acc)	Avg.
Full-FT	6.6B	83.4	88.7	89.8	67.9	92.3	93.6	86.3	**91.7**	86.7
LoRA	4.2M	$89.2_{\pm 0.5}$	$89.7_{\pm 0.5}$	$89.9_{\pm 0.1}$	$\mathbf{70.6}_{\pm 0.7}$	$\mathbf{96.8}_{\pm 0.2}$	$94.7_{\pm 0.2}$	$\mathbf{90.9}_{\pm 0.2}$	$91.6_{\pm 0.1}$	89.2
LoRA-drop (ours)	2.2M	$\mathbf{91.0}_{\pm 0.5}$	$\mathbf{90.2}_{\pm 0.3}$	$\mathbf{90.1}_{\pm 0.1}$	$69.0_{\pm 1.2}$	$\mathbf{96.8}_{\pm 0.2}$	$\mathbf{94.8}_{\pm 0.2}$	$90.6_{\pm 0.1}$	$91.6_{\pm 0.3}$	**89.3**

Table 4: The performance of the Llama2-7b on GLUE benchmark.

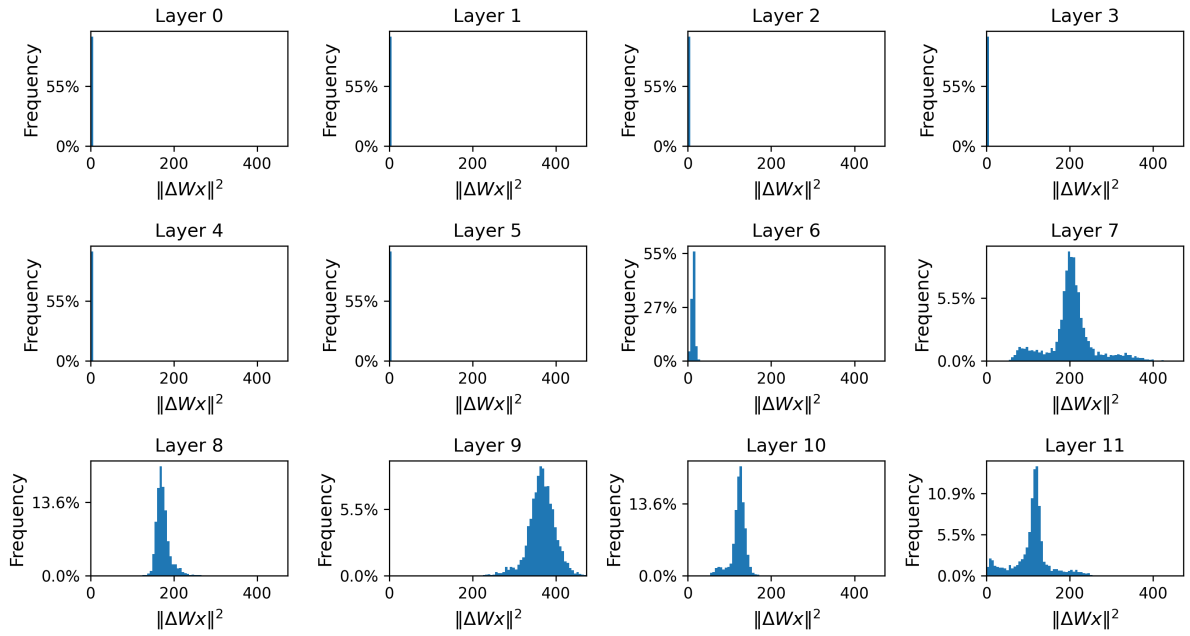

Figure 5: The frequency distribution of the squared norm of query LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ after fine-tuning on the MRPC task.

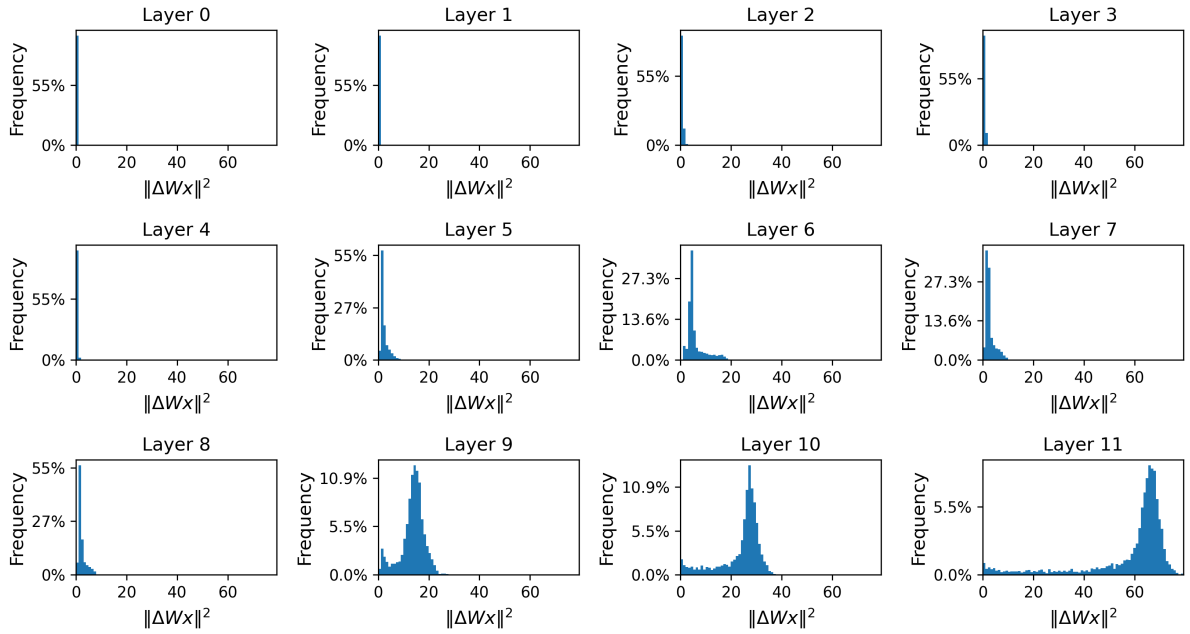

Figure 6: The frequency distribution of the squared norm of value LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ after fine-tuning on the MRPC task.

Model	#Tr.	E2E		DART	
Llama2 7b	Params	BLEU	ROUGE-L	BLEU	ROUGE-L
Full-FT	6.6B	**55.65**	**39.19**	**59.68**	**47.18**
LoRA	4.2M	53.70	35.94	57.42	42.92
LoRA-drop	2.1M	53.49	35.92	57.17	42.21

Table 5: The performance of Llama2-7b on two table2text datasets including E2E and DART. For both metrics, BLEU and ROUGE-L, higher is better.

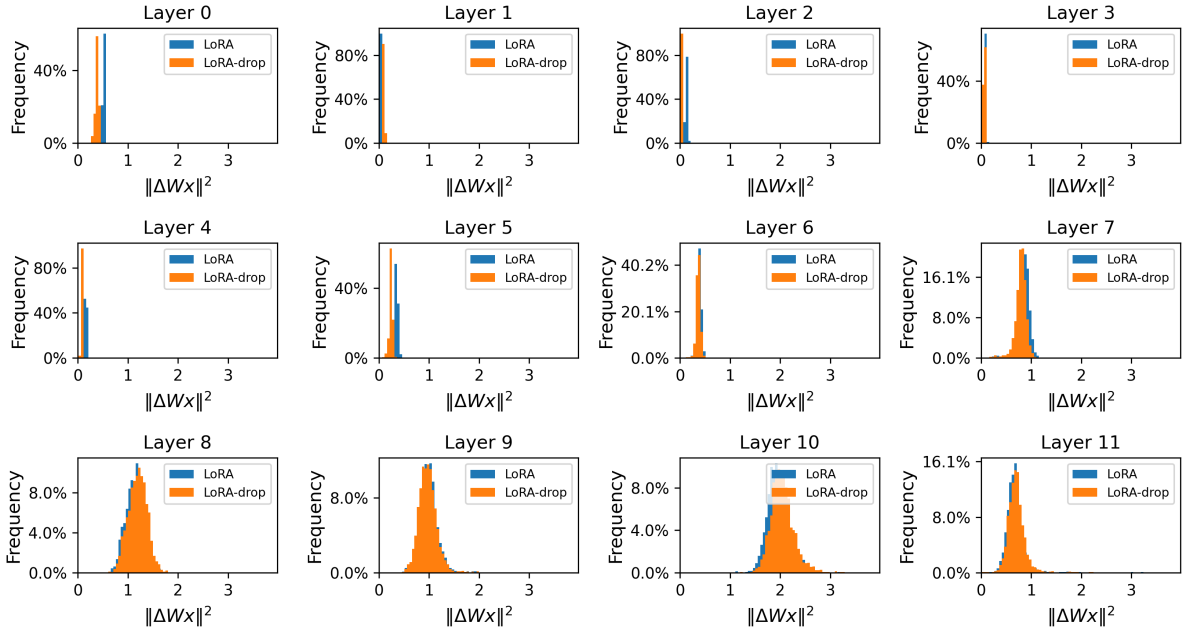

Figure 7: The query LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ squared norm frequency distribution of LoRA and LoRA-drop.

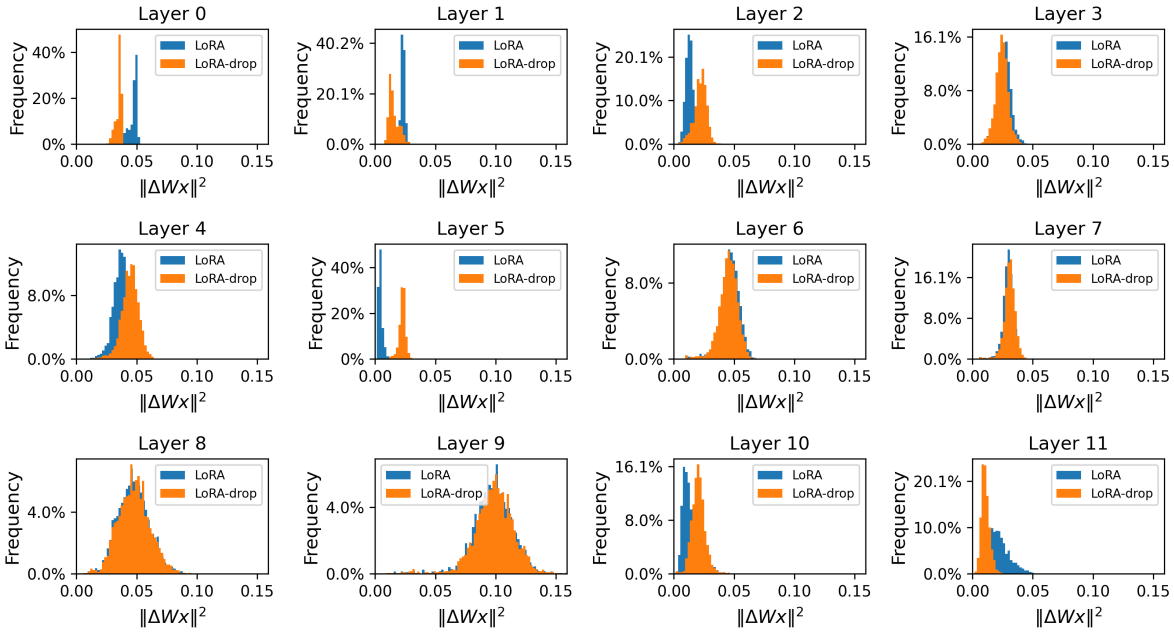

Figure 8: The value LoRA output $\Delta \boldsymbol{W}_i \boldsymbol{x}_i$ squared norm frequency distribution of LoRA and LoRA-drop.

Threshold	Avg. layer num		RTE	CoLA	QNLI	QQP	Avg. score
	W_query	W_value	(ACC)	(Matt.)	(ACC)	(ACC)	
1(LoRA)	12	12	82.3	61.9	93.1	90.4	82.0
0.95	6	9	83.0	62.6	93.1	90.2	82.2
0.9	5	7	81.9	63.1	93.2	90.2	82.1
0.8	5	5	80.9	63.1	93.2	89.6	81.7
0.7	4	4	78.3	62.1	92.5	89.3	80.6

Table 6: The influence of the threshold T and its equivalent average number of layers.

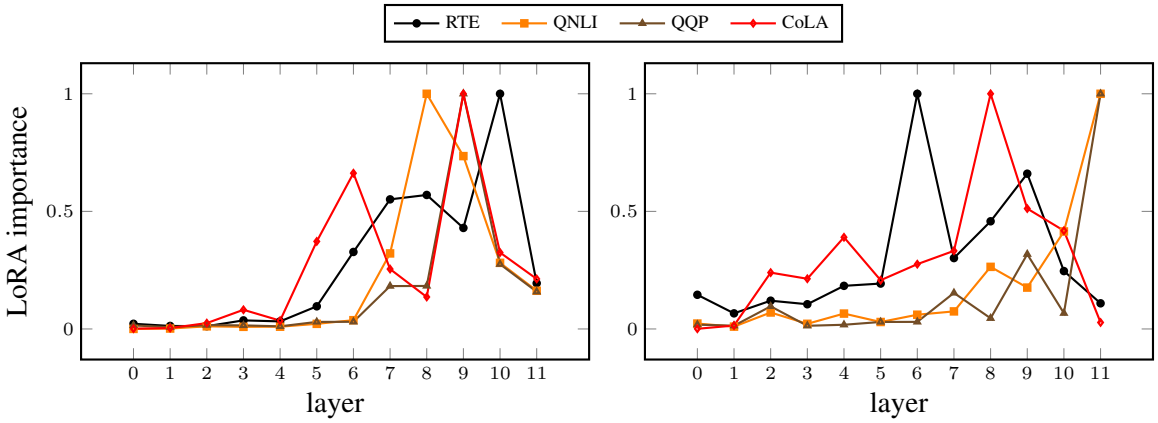

Figure 9: The relative magnitudes of LoRA outputs across different layers of RoBERTa-base on various datasets, for display, the value of the largest layer's LoRA output is normalized to 1 for each dataset.

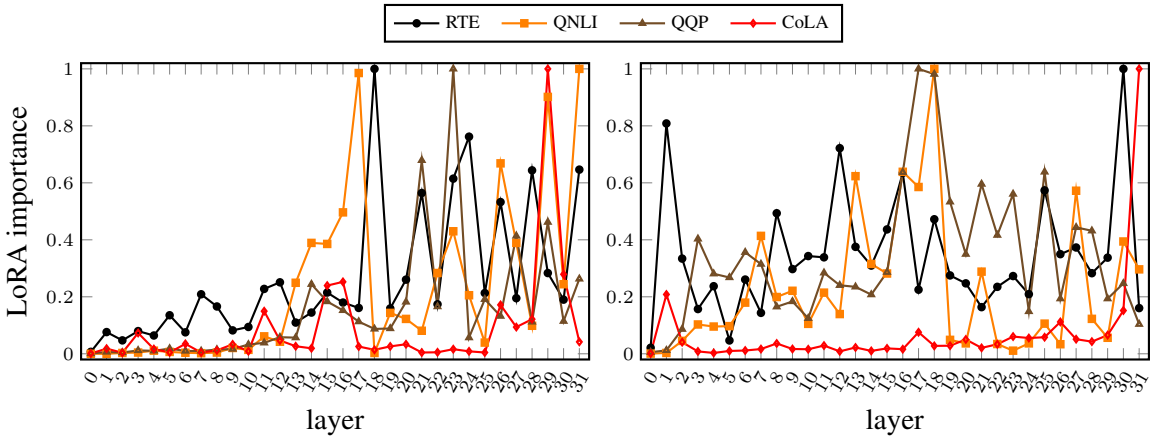

Figure 10: The relative magnitudes of LoRA outputs across different layers of Llama2-7b on various datasets, for display, the value of the largest layer's LoRA output is normalized to 1 for each dataset.

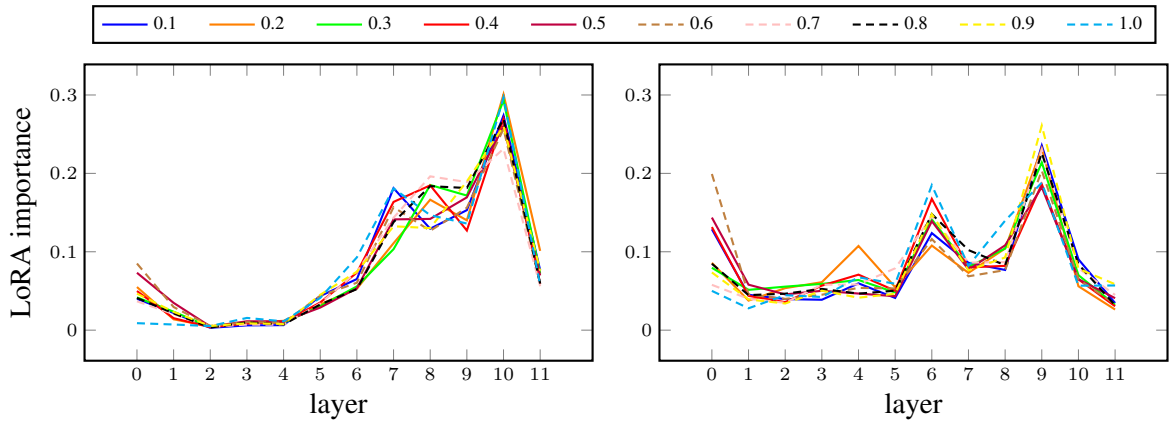

Figure 11: The effect of the sample proportion on the importance scores during the pre-processing stage. The lines of different colors represent sampling from varying proportions of training data.

Model (RoB-base)	RTE (Acc)	MRPC (Acc)	STS-B (Spea.)	CoLA (Matt.)	SST-2 (Acc)	QNLI (Acc)	MNLI (Acc)	QQP (Acc)	Avg.
LoRA-drop*	81.9	90.0	91.1	63.1	94.7	93.2	87.5	90.2	86.5
LoRA-drop(w/o share)	80.4	88.9	90.7	62.8	94.1	92.9	86.9	89.7	85.8
LoRA-drop(ΔWx inv)	79.1	89.7	90.4	60.5	94.3	92.9	87.3	89.9	85.5
LoRA-drop(random)	79.1	89.2	90.2	62.0	94.6	92.7	86.9	89.8	85.6
LoRA-drop(top k)	81.9	89.2	90.7	62.3	94.5	93.0	86.8	89.8	86.0

Table 7: Ablation experiments