VECTOR SEGMENTED AND RECOMBINED ADAPTATION FOR SCALABLE AND EFFICIENT MODEL TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Among the most commonly utilized parameter-efficient fine-tuning (PEFT) methods, LoRA and its variations have achieved significant popularity. The Vector-based Random Matrix Adaptation (VeRA), one typical variant, utilizes random weights and projections to reduce the number of trainable parameters greatly. However, it requires additional GPU memory and computational resources, probably resulting in a lack of scalability that leads to performance bottlenecks in complex tasks. Besides, the inappropriate initialization of random matrices may affect model performance. To address these problems, we propose a new method called Vector **Se**gmented and **R**ecombined **A**daptation (SeRA). SeRA segments input vectors into sub-vectors for individual dimensionality reduction, then introduces a square matrix to combine the information from the reduced sub-vectors, and finally expands the dimensionality independently to adapt the size of pre-trained model. SeRA allows for flexible increase of trainable parameters to enhance performance in complex tasks, and avoids the problem caused by random matrices initialization. Through evaluations on the image classification, cross-modal image-text retrieval, instruction-tuning and GLUE benchmark, we demonstrate the scalability and efficiency of SeRA. Furthermore, we utilize Singular Value Decomposition on the adaptation matrices of SeRA, to analyze how the information characteristics of the matrices change in different ranks and tasks. The results can serve as the guide for selecting appropriate parameter amounts in different tasks.

1 INTRODUCTION

With the rapid development of intelligent models, the demands for them from society and commerce have become increasingly diverse. Although general models like GPT-4 (OpenAI, 2023) can meet most daily needs, there is still considerable room for improvement in specialized domains. For example, Codex (Chen et al., 2021), which is a code generator developed by OpenAI based on GPT, demonstrates superior performance to general models after being fine-tuned on open-source code data from GitHub.

Compared to full-parameter fine-tuning, LoRA (Hu et al., 2022) is a simple and efficient fine-tuning method. It introduces low-rank matrices, updating only a small number of parameters while keeping most pre-trained parameters fixed, to achieve good performance in specific tasks. Many variants of LoRA have been proposed to enhance parameter efficiency or performance. VeRA (Kopiczko et al., 2024), a recently proposed efficient LoRA variant, uses "scaling vectors" to adjust frozen random matrices shared across layers, achieving comparable performance with only one-tenth parameters compared to LoRA. Although models utilizing random weights and projections can significantly reduce parameter usage (Peng et al., 2021; Ramanujan et al., 2020), the fixed, randomly initialized matrices still occupy GPU memory, increasing resource consumption, which may limit its scalability. We report VeRA's GPU memory consumption in the Appendix B. Additionally, experiments show that VeRA may fail to converge under certain seed initializations.

In recent fine-tuning study, MELoRA (Ren et al., 2024) and MoSLoRA (Wu et al., 2024) have demonstrated remarkable advantages in terms of parameter efficiency and flexibility, respectively. MELoRA achieves efficient feature extraction by segmenting and reducing the dimensionality of the input vector, while MoSLoRA introduces a matrix adjustment mechanism to bolster the model's adaptability to intricate tasks. Drawing inspiration from the multi-head attention mechanism, this paper proposes a novel method Vector **Se**gmented and **R**ecombined **A**daptation (SeRA), which formally integrates these two techniques. SeRA features two key steps as follows. **(1) Vector segmentation**: the input vectors of the fine-tuning layers are split into several sub-vectors. SeRA independently reduces and expands their dimensionalities, reducing the number of SeRA's parameters. **(2) Sub-vector aggregation**: after dimensionality reduction, a square matrix is introduced to adjust the reduced sub-vectors. This process is equivalent to expanding their dimensionalities and adding them together, which fuses the information of the sub-vectors, ensuring performance while maintaining efficiency. This method aims to provide a flexible and efficient solution that can navigate the evolving complexity of tasks. Take the autonomous driving image classification task as an example, a fine-tuning method is required to quickly adapt to data changes in order to iterate the model in the early stages of training, which requires the method to be highly efficient; as the scale of the task increases, the lack of scalability may lead to performance bottlenecks. We design increasingly complex classification modes to simulate the real situation for experiment, and show that SeRA can adapt well to this increasingly complex task.

Hu et al. (2022) has shown that for tasks requiring only minor parameter adjustments to achieve good performance, increasing the rank of trainable adaptation matrix does not improve performance. However, many studies have argued that increasing the rank of the adaptation matrix leads to better model performance in memory-intensive tasks, so to analyze the reasons for this, we conducted experiments across tasks with different ranks with SeRA. By applying singular value decomposition (SVD) to the fine-tuned adaptation matrix, we aim to explain this phenomenon from the perspective of singular vector subspaces and singular values. ALL in all, the key innovations and contributions of this paper are as follows:

- We propose SeRA, a scalable and efficient method that avoids the problems caused by random weights and projections. We demonstrate its effectiveness through image classification, cross-modal image-text retrieval, instruction-tuning, and natural language understanding (GLUE), as well as its scalability in the image classification.
- Using SeRA, we conducted experiments across tasks with varying ranks, applying singular value decomposition (SVD) to analyze how the rank of adaptation matrix affects information and performance across different tasks. This analysis will aid in selecting the appropriate number of trainable parameters for different tasks.

2 Related Work

Multi-Head Attention (MHA) Multi-Head Attention (MHA) is a core component of the Transformer model, first introduced by Vaswani et al. (2017). It enhances the model's ability to capture complex patterns by computing correlations in different subspaces in parallel across multiple attention heads. Each head learns distinct attention patterns, and the results are concatenated and projected back to the original dimension. This mechanism significantly improves the model's capability to capture global information. Subsequent research has further expanded the theory and applications of MHA. For instance, Cordonnier et al. (2020) demonstrated that multi-head attention has the capability to approximate convolutional kernels, providing additional insights into its representational power. Various optimization strategies for MHA have been proposed, such as hierarchical attention mechanisms (Yang et al., 2020) and sparse attention mechanisms (Child et al., 2019), which primarily aim to improve computational efficiency and model adaptability. SeRA introduces a novel lightweight adaptation approach by segmenting input vectors, which is closely related to the grouping computation concept in multi-head attention (MHA). Additionally, the matrices in SeRA share the same optimization design objectives as the projection matrices in MHA, further enhancing the model's representational capacity across different tasks.

Vector-based Random Matrix Adaptation (VeRA) VeRA is a more efficient method than LoRA, which achieves efficient training by introducing trainable "scaling vectors" and freezing random matrices during training, achieving comparable performance to LoRA in GLUE with only one-tenth of LoRA's parameters. However, despite the low number of fine-tuned parameters in VeRA, the random matrix still needs to be involved in the computation and save the activations in the GPU. When the "scaling vectors" are relatively large, the random matrix takes up a lot of memory, limiting the scalability of VeRA.

Recent different variants of LoRA Since LoRA was introduced, numerous improved variants have been proposed, with previous work focusing on three main areas: Firstly, enhancing LoRA's parameter efficiency, which further reduce parameter usage without sacrificing performance (Zhou et al., 2024; Zhang et al., 2023a; Kopiczko et al., 2024; Ren et al., 2024); Secondly, addressing the limitations of LoRA's low-rank update in certain tasks by increasing the matrices's rank without adding extra trainable parameters (Jiang et al., 2024; Wu et al., 2024); Lastly, improving LoRA's performance through optimized training strategies (Hayou et al., 2024; Zhang et al., 2023b; Liu et al., 2024). While many optimization strategies can be applied simultaneously, balancing the goals of parameter efficiency and scalability remains a challenge. An scalable and efficient method capable of handling tasks of varying complexity forms the foundation of our proposed SeRA.

3 METHOD

3.1 METHOD STRUCTURE

In the specific implementation of the multi-head attention (Vaswani et al., 2017), the matrix is divided into several matrix blocks after isometric mapping. Each matrix block is regarded as a head, this method divides the original vector space into several vector subspaces for attention computation. No additional trainable parameters are introduced which means higher parameter efficiency. Each head represents information of different spaces after being computed by the attention function. The way to combine them is to expand their dimensionalities then add them directly by adjusting the square matrix W_o for both steps. If we want to increase the expressiveness of model, just increase the number of heads. Based on above two points, the structure of SeRA is shown in Figure 1, and the matrix expression is shown in the Equation 1 and Equation 2.

$$h = xW_0 + x\Delta W = xW_0 + xACB \tag{1}$$

$$A = \begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & A_{split} \end{pmatrix} \qquad B = \begin{pmatrix} B_1 & 0 & \cdots & 0 \\ 0 & B_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & B_{split} \end{pmatrix} \tag{2}$$

We define the input vector as $x \in \mathbb{R}^{d_{in}}$ and the output vector as $h \in \mathbb{R}^{d_{out}}$, Firstly, the input vector is divided into $split$ sub-vectors: $Split\,(x) = [x_1, \cdots x_{split}]$ where $x_i \in \mathbb{R}^{\frac{d_{in}}{split}}$ $i = 1, 2, 3 \cdots split$ and then calculated by A. This process involves reducing the dimensionality of each sub-vector separately: $x_i A_i = c_i^{in}$, where $c_i^{in} \in \mathbb{R}^{\frac{r}{split}}$ and $A_i \in \mathbb{R}^{\frac{d_{in}}{split} \times \frac{r}{split}}$. From a local perspective, all reduced sub-vectors form r-dimensional intermediate vector. From a global perspective, the whole input is reduced to r-dimensional intermediate vector through matrix A. The formula can be expressed as:

$$xA = c^{in} = Concat\left(c_1^{in}, \cdots, c_{split}^{in}\right) \tag{3}$$

where $c^{in} \in \mathbb{R}^r$, $A \in \mathbb{R}^{d_{in} \times r}$ and A is a sparse matrix with a block diagonal structure. After dimensionality reduction, the data will be calculated by matrix C. The role of the matrix C is to expand each of the reduced sub-vectors to r-dimensionality and

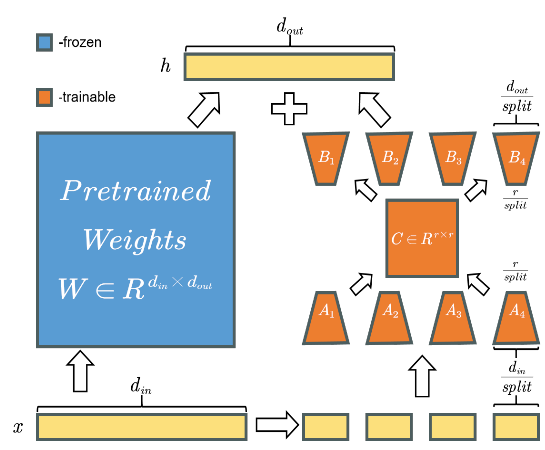

Figure 1: Structure of the SeRA. The input vector is split into several blocks (i.e. $split = 4$) and each block is reduced in dimensionality by the matrix A_i. The information fusion is performed by the matrix C, and then each block is expanded in dimensionality by the matrix B_i.

then accumulate them, which acts like the matrix W_o in the multi-head attention. Finally, expand the dimensionality of the data through matrix B to adapt size of pre-trained model. This calculation is symmetrical to the calculation of the matrix A. Consistent with LoRA, we scale the $x\Delta W$ by α which is a constant.

3.2 Parameter Count Analysis

The number of parameters fine-tuned by SeRA for a particular layer is $|\Theta| = \frac{d_{in}\times r_1}{split} + \frac{d_{out}\times r_1}{split} + r_1^2$ while the number of parameters fine-tuned by VeRA is $|\Theta| = d_{out} + r_2$. Due to $r_1 \ll d_{in}$ and $r_1 \ll d_{out}$ but r_2 is much bigger, the trainable parameters of both SeRA and VeRA can be compressed very little. However the number of parameters that VeRA needs to store in the gpu during training is $|\Theta| = d_{in}r_2 + d_{out}r_2 + r_2 + d_{out}$, which takes up a huge amount of gpu memory while SeRA is still $|\Theta| = \frac{d_{in}\times r_1}{split} + \frac{d_{out}\times r_1}{split} + r_1^2$. This confirms that SeRA uses less gpu memory than VeRA for the same number of fine-tuned parameters. In addition, we must clarify that although the drawbacks of VeRA in this regard may not be obvious in the context of small models with low parameter counts fine-tuning, for example, in the face of some progressively complex project contexts (e.g., self-driving road environment recognition) VeRA can not be scaled up from simple business scenarios to complex business scenarios, and when the size of the task increases, VeRA will cause performance bottlenecks due to the bottleneck of gpu memory. Then we must switch to other methods, resulting in switching costs. SeRA is able to adapt to tasks of arbitrary complexity and maintain method consistency throughout the project process.

Note that in SeRA, $split|r$ must be satisfied, which means $r = \alpha \times split$ and α is a positive integer. Then, the number of fine-tuned parameters of SeRA is $|\Theta| = (d_{in} + d_{out}) \times \alpha + r^2$. The two factors affecting the trainable parameters, the size of the pre-trained model and the rank of the adaptation matrix, are independent of each other, allowing SeRA to flexibly adjust the rank according to the size of pre-trained model.

4 Experiments

In this section, we present a series of experiments to evaluate SeRA. $split = r$ is used by default unless the settings of $split$ are mentioned. We begin by comparing our method to VeRA and other baselines on the RSCD (Zhao et al., 2023), which is the autonomous driving image classification task. We designed three gradually complex classification modes to simulate the gradually complicating situations in reality. Results demonstrate the scalability and efficiency of our method. Next, we turn our attention to cross-modal image-text retrieval using the CLIP model (Radford et al., 2021). We found that SeRA has excellent performance in this task. We then explore fine-tuning the LLaMA-3 model (AI@Meta, 2024) in the context of instruction tuning (Ouyang et al., 2022). Additionally, we evaluate SeRA in GLUE benchmark and perform singular value decomposition (SVD) analysis. Finally, an ablation study highlights the importance of each component in our method.

4.1 Image Classification

This analysis compares the performance of different fine-tuning methods on the image classification task, choosing the Road Surface Classification Dataset (RSCD) which is a valuable resource for research in the field of autonomous driving (Zhao et al., 2023). RSCD contains about one million image data, due to time and budget constraints, we performed a random selection of 10% of the training data from each category in the RSCD and used the full validation set to test. $\text{ViT}_{\text{Large}}$ (Dosovitskiy et al., 2021), which was pre-trained on 21K imagenet, was chosen as the pre-trained model.

The RSCD categorizes road surface conditions into three main attributes: friction level, road material, and road unevenness. The friction level attribute is subdivided into six categories based on varying weather conditions: dry, wet, water, fresh snow, melted snow, and ice. The road material attribute includes asphalt, concrete, mud, and gravel, while road unevenness is classified into smooth, slight unevenness, and severe unevenness, depending on the road's undulation amplitude. The dataset defines its classes by combining these three attributes. The road material and unevenness annotations are absent when the friction level is fresh snow, melted snow, or ice. Additionally, unevenness is not annotated for mud or gravel roads, resulting in a total of 27 combined classes. We designed three experimental modes to simulate increasingly complex scenarios for autonomous driving systems: “easy”, “medium”, “full”. The first mode, “easy”, fixes the friction level as dry and road material as asphalt, aiming to classify the road unevenness into three categories: severe, slight, or smooth. In the “medium” mode, only the friction level is fixed as dry, all road materials and road unevenness types are considered, resulting in a total of 8 classes. Finally, the “full” mode classifies across all categories, with a total of 27 classes.

In all modes, the rank of VeRA is 1024. In the “easy” mode, the rank for LoRA is 8, and for SeRA is 16. In the “medium” mode, LoRA’s rank remains the same, while SeRA’s rank is increased to 64. In the “full” mode, the rank of LoRA is set to [10, 40, 64], while for SeRA, the *split* and r are configured as [8, 8], [128, 128], [32, 256], and [8, 256]. For comparison with MELoRA, we performed a MELoRA test in “full” mode with the rank set to 1024 and split to 16, which means that each minilora is a 64-ranked. For all tuning methods, the classification head was fully adjusted and excluded when calculating the number of parameters, only the query and value layers are fine-tuned. Full parameter fine-tuning and training only the classification head were chosen as additional baseline.

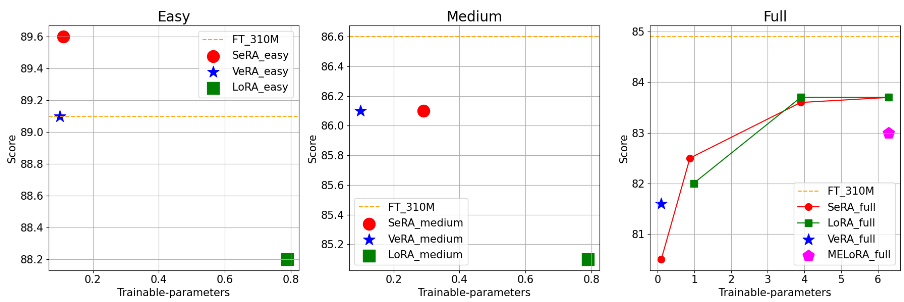

Figure 2: From left to right: Easy, Medium, Full. SeRA demonstrated efficiency across all modes, and due to its scalability, it achieved better performance than VeRA in the “full” mode by adjusting additional parameters. Note that due to space constraints, we did not include the results of training only the head in this figure. The results are **[75.6, 73.4, 67.7]** from “easy” to “full” mode.

From the results in Figure 2, we observe that VeRA maintains high efficiency in the “easy” and “medium” modes. However, in the more complex “full” mode, its performance lags behind other methods due to the limited number of adjustable parameters. LoRA shows low efficiency in relatively simple tasks. For MELoRA we used a relatively high rank for the experiment, but the result is not as good as LoRA and SeRA, which shows that rank is not the only factor affecting the model performance, but the way of dimensionality reduction and expansion also affect the model performance. Notably, SeRA shows outstanding results across all modes, demonstrating high efficiency in the “easy” and “medium” modes. SeRA performs well in “full” mode, which demonstrates its scalability. Constrained by space limitation, we reported additional experimental results with differnet datasets in Appendix D, which were able to reach the same conclusion.

4.2 Cross-Modal Image-Text Retrieval

The cross-modal image-text retrieval task is to search for relevant samples on another modality (e.g., text) based on query samples in one modality (e.g., image), which mainly involves modal representation, similarity representation and retrieval algorithms. We use $\text{CLIP}_{\text{base}}$ and $\text{CLIP}_{\text{large}}$ as pre-trained models, and fine-tune them using SeRA, VeRA, LoRA and MELoRA on the MSCOCO dataset (Lin et al., 2015). Triplet Loss function will be used to optimize these models (Schroff et al., 2015), which has the advantage that when two inputs are similar, Triplet loss enables better modeling of details.

We applied rank 8 to LoRA, MELoRA and SeRA on the visual and textual sides of CLIP with any size, in addition, we performed MELoRA experiments with rank 16. For VeRA, we applied rank 1024 on the textual side and the visual side for $\text{CLIP}_{\text{base}}$, and applied rank 2048 for $\text{CLIP}_{\text{large}}$. All methods fine-tuned only the query and value layers. In addition, we choosed full-parameter fine-tuning and zero-shot as baseline for comparisons. Each image in this dataset is accompanied by at least 5 textual descriptions, we used full data from the training set (containing 118,287 images), and test them on full validation set (containing 5,000 images). We use the recall[1] metric to compare different methods.

Based on results in Table 1, for $\text{CLIP}_{\text{base}}$, SeRA even exceeds the performance of full-parameter fine-tuning. For $\text{CLIP}_{\text{large}}$, SeRA exerts the highest performance of all PEFT methods.

Cross-modal image-text retrieval is a key challenge in modal alignment within the multimodal domain (Cao et al., 2022). Both text encoders and visual encoders have already learned their respective modal representations and only need to align them. Generally, various relationships exist between different modalities. An image can be described in multiple ways of text. It is difficult to determine which description is most appropriate (Chun et al., 2021). SeRA is capable of capturing and synthesizing information across multiple vector spaces, enabling it to identify the diverse connections between images and text in different dimensions. This ability contributes to SeRA's superior performance in this task.

Table 1: CLIP base (B) and large (L) with different fine-tuning methods on MSCOCO. SeRA outperforms several baselines with comparable or fewer trainable parameters.

	Method	#Trainable Parameters	R1_i2t	R1_t2i	R5_i2t	R5_t2i	R10_i2t	R10_t2i	Sum
Base	Zero-Shot	0M	49.92	30.392	74.6	54.684	83.1	66.144	358.84
	FT	151.8M	**54.76**	37.204	**78.54**	63.636	86.12	74.004	394.264
	LoRA	0.492M	49.92	31.456	75.34	56.204	83.96	67.04	363.92
	VeRA	0.080M	53.9	35.416	77.5	60.472	85.86	71.392	384.54
	MELoRA	0.061M	52.88	37.796	77.12	64.16	84.92	74.6	391.476
	MELoRA	0.123M	52.64	38.524	77.58	**64.972**	85.48	**75.224**	394.42
	SeRA	0.064M	53.62	**38.56**	77.56	64.676	**86.16**	75.124	**395.7**
Large	Zero-Shot	0M	56.1	35.528	79.56	59.836	86.82	70.192	388.036
	FT	428.7M	**58.22**	41.732	**81.3**	67.04	**88.06**	76.704	**413.056**
	LoRA	1.081M	56.98	36.024	80.24	60.324	87.42	70.688	391.676
	VeRA	0.141M	56.1	35.528	79.5	59.88	86.88	70.208	388.096
	MELoRA	0.135M	52.88	41.78	77.18	67.028	85.18	76.752	400.8
	MELoRA	0.27M	50.78	41.564	75.0	66.968	83.86	76.54	394.712
	SeRA	0.14M	53.38	**42.108**	77.82	**67.28**	85.3	**76.916**	402.804

4.3 Instruction Tuning

Instruction tuning is a process that involves fine-tuning a pre-trained large language model using natural language instruction data in a supervised manner. This method aims to enhance the model's ability to comprehend and follow specific instructions, thereby improving its performance on designated tasks (Ouyang et al., 2022). We trained the LLaMA3-8B model using LoRA, SeRA, MELoRA, MoSLoRA on a cleaned version of the Alpaca dataset (Taori et al., 2023), employing quantization techniques (Dettmers et al., 2023) to enable operation on a single GPU.

We evaluated models on MT-Bench (Zheng et al., 2023) by generating outputs for a set of 80 predefined multi-round questions and scoring the responses using GPT-4 (OpenAI, 2023). GPT-4 assigns a score from 1 to 10 for each answer. Then we evaluated models on MMLU (Massive Multitask Language Understanding) (Hendrycks et al., 2021): A benchmark for multitask language understanding covering 57 disciplines, designed to test the model's generalization across a wide range of tasks. BBH (Big-Bench Hard) (Srivastava et al., 2023): A challenging subset of the Big-Bench dataset, used to evaluate the model's reasoning capabilities in complex tasks. DROP (Discrete Rea-

[1]Taking R10_i2t as an example, for a given image, calculate its cosine similarity with all text samples and sort them in descending order. If any of the five texts associated with the image is ranked in the top ten, the image is considered to have successfully retrieved the target text. Result of this metric is the percentage of successful retrievals relative to the total number.

Table 2: Instruction tuning experiment of various methods

Method	#Trainable Parameters	MMLU	BBH	DROP	Human eval	AVG	MT-Bench
SeRA$^{r=64,split=64}$	3.5M	62.58	**43.53**	49.21	40.85	**49.04**	6.50
SeRA$^{r=32,split=16}$	5.5M	63.12	42.76	**49.39**	**40.85**	49.03	**6.63**
MeLoRA$^{r=64,split=64}$	2.6M	63.17	42.25	48.83	40.85	48.775	6.50
MeLoRA$^{r=128,split=64}$	5.2M	**63.48**	43.01	49.32	37.8	48.40	6.28
MoSLoRA$^{r=16}$	42.0M	62.46	42.87	48.18	39.63	48.285	6.15
LoRA$^{r=64}$	167.8M	62.07	42.9	47.35	35.98	47.08	5.68

soning Over Paragraphs) (Dua et al., 2019): A dataset for assessing the model's ability to handle discrete reasoning problems in reading comprehension tasks. HUMANEVAL (Chen et al., 2021): A benchmark for evaluating the model's performance in code generation, focusing on functionality and correctness. For more information on the experimental setup, please refer to Appendix E. We computed their averages except for MT-Bench in the AVG column, results are shown in Table 2.

The experimental results demonstrate the superiority of SeRA in terms of performance of instruction tuning.

4.4 Natural Language Understanding

We evaluated our method on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) using RoBERTa$_{large}$ model (Liu et al., 2019). In addition to LoRA and VeRA, we also compare SeRA to the following baselines. **Adapter tuning**: Initially introduced by Houlsby et al. (2019), it involves integrating adapter layers between the self-attention and MLP modules followed by a residual connection. This configuration, denoted as **AdapterH**, includes two fully connected layers and a nonlinearity. A variation, **AdapterP** by Pfeiffer et al. (2021), employs the adapter layer only after the MLP module and subsequent to a LayerNorm. **LoRA-FA**: This baseline freezes the A matrix during training and is able to reduce the number of parameters by about half and maintain performance.

For RoBERTa$_{large}$, we applied SeRA with ranks of 8 and 16. A and C are initialized using Kaiming initialization (He et al., 2015), B is initialized with zero. For VeRA, we used the HuggingFace PEFT implementation (Mangrulkar et al., 2022). To reproduce the VeRA method, we ran experiments with seeds [42, 64, 128, 256, 512], following the hyperparameter settings in the original paper (Kopiczko et al., 2024).

Our experimental setup is consistent with Hu et al. (2022). Only the query and value layers are fine-tuned. Classification head has the same learning rate as the fine-tuning layer, and its trainable parameters are excluded in the calculation. See Appendix A for specific settings. Due to time and budget constraints, we omitted the time-consuming MNLI and QQP tasks, and consequently did not apply the MNLI trick[2] to the MRPC, RTE, and STS-B tasks. We conducted five runs with different random seeds.

Table 3 shows the results. We found that the performance achieved by SeRA on each task is comparable to other methods, but with a smaller number of parameters. For VeRA, we observed training instability, where the model fails to converge or shows erratic for the MRPC, CoLA, and RTE tasks on certain seeds. The random seeds primarily influenced the initialization of the frozen matrices, and suboptimal initialization may lead to difficulties in model training.

4.5 Analysis Of The Effect Of Trainable Parameters On Performance

The experiments above demonstrate that for tasks requiring only minor parameter adjustment to achieve good performance, increasing the number of trained parameters does not lead to perfor-

[2]For RoBERTa model and MRPC, RTE and STS-B tasks, Hu et al. (2022) initialized the model with the best weights finetuned on the MNLI task.

Table 3: Results for different adaptation methods on the GLUE benchmark. We report Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for the remaining tasks. In all cases, higher values indicate better performance. Results of all methods except SeRA and VeRA* are sourced from prior work (Hu et al., 2022; Zhang et al., 2023a; Kopiczko et al., 2024). VeRA* indicates running under five random seeds. *Italics* indicate problems with training under certain seeds.

Method	#Trainable Parameters	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
Adpt^{P}	3M	96.1$_{\pm 0.3}$	90.2$_{\pm 0.7}$	**68.3$_{\pm 1.0}$**	**94.8$_{\pm 0.2}$**	83.8$_{\pm 2.9}$	92.1$_{\pm 0.7}$	87.6
Adpt^{P}	0.8M	**96.6$_{\pm 0.2}$**	89.7$_{\pm 1.2}$	67.8$_{\pm 2.5}$	94.8$_{\pm 0.3}$	80.1$_{\pm 2.9}$	91.9$_{\pm 0.4}$	86.8
Adpt^{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
Adpt^{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
LoRA-FA	3.7M	96	90	68	94.4	86.1	92	87.7
LoRA	0.8M	96.2$_{\pm 0.5}$	90.2$_{\pm 1.0}$	68.2$_{\pm 1.9}$	94.8$_{\pm 0.3}$	85.2$_{\pm 1.1}$	**92.3$_{\pm 0.5}$**	**87.8**
VeRA	0.061M	96.1$_{\pm 0.1}$	**90.9$_{\pm 0.7}$**	68.0$_{\pm 0.8}$	94.4$_{\pm 0.2}$	85.9$_{\pm 0.7}$	91.7$_{\pm 0.8}$	87.8
VeRA*	0.061M	95.4$_{\pm 0.6}$	*68.9*	*67.5*	94.3$_{\pm 0.2}$	*62.1*	91.7$_{\pm 0.4}$	80.0
SeRA$^{(r=8)}$	0.1M	95.9$_{\pm 0.3}$	90.2$_{\pm 0.2}$	66.6$_{\pm 0.4}$	94.6$_{\pm 0.1}$	85.6$_{\pm 0.3}$	91.8$_{\pm 0.1}$	87.5
SeRA$^{(r=16)}$	0.11M	96.0$_{\pm 0.}$	90.2$_{\pm 0.2}$	66.3$_{\pm 0.2}$	94.7$_{\pm 0.1}$	**86.6$_{\pm 0.8}$**	91.8$_{\pm 0.1}$	87.6

mance gains. However, for more complex tasks, increasing the number of parameters can progressively improve performance. In this section, we applied singular value decomposition (SVD) to the various rank adaptation matrices fine-tuned with SeRA across different tasks. We analyzed how the information characteristics represented by different ranks of the adaptation matrix vary in the context of these tasks.

Firstly, we fine-tuned RoBERTa$_{\text{large}}$ on the RTE task using SeRA with ranks of [8, 16, 32, 64, 128, 256]. The hyperparameter settings remain the same as above except rank.

Table 4: RoBERTa$_{\text{large}}$ fine-tuned with SeRA with different rank, the results show that the performance of this task is independent of the number of trainable parameters.

# Parameters	0.1M	0.11M	0.15M	0.29M	0.88M	3.24M
Score	85.5	86.14	86.04	86.64	85.92	86.14

Table 4 shows that the performance does not get better with increasing number of parameters. In general, there is a large amount of noise and redundant information in the text data, and too many parameters lead to the model overfitting the training data, which leads to poor performance on the validation set. The text understanding task tends to focus on the model's ability to understand the linguistic structure and inference, rather than simple parameter fitting. The design of the architecture and the choice of fine-tuning strategy have a more significant impact on the effectiveness of the model.

For image classification task, "full" mode, which is shown in Figure 2, it is required that the model can accurately detect the subtle differences in the image. Increasing the number of fine-tuning parameters usually means that the model has more degrees of freedom to fit the details and features in the training data, which can improve the performance of the model on the validation set.

To analyze the results of singular value decomposition, we referred to the method which has been used in the Hu et al. (2022). Given $\Delta W_{r=8}$ and $\Delta W_{r=128}$, which are adaptation matrices of rank 8 and 128 from SeRA learned using the same pre-trained model. A singular value decomposition of them yields the left singular matrices $U_{r=8}$ and $U_{r=128}$. Calculate how much of the first i singular vector subspaces of $U_{r=8}$ are contained in the first j singular vector subspaces of $U_{r=128}$ by Equation 4.

$$\varphi(\Delta W_{r=8}, \Delta W_{r=128}, i, j) = \frac{||U_{r=8}^{iT} U_{r=128}^{j}||_F^2}{min(i, j)} \in [0, 1] \tag{4}$$

Value 0 means that the subspaces are completely separated and 1 means that the subspaces are completely overlapped, we give the results for the average of all the query layers and point out in the Appendix C that the conclusions for the value layer are consistent. We also report the singular values of the adaptation matrix for analysis.

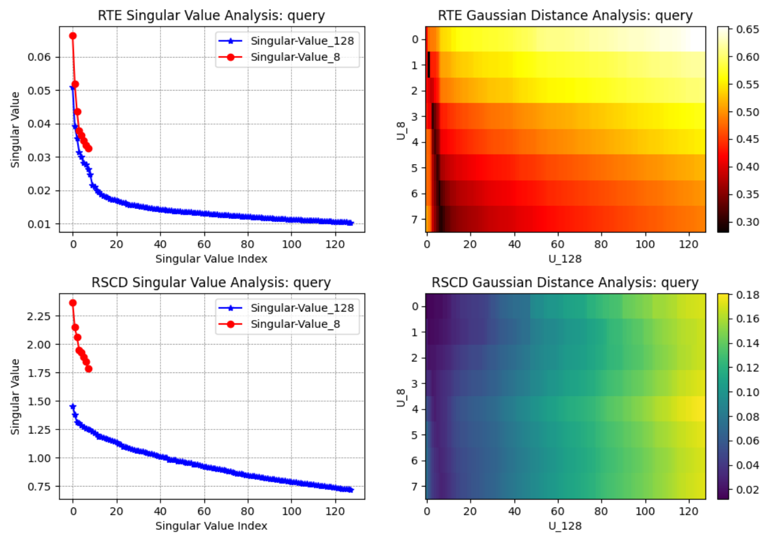

Figure 3: The left side of the image shows the singular values in descending order of size, and the right one shows the singular vector subspace similarity between $U_{r=8}$ and $U_{r=128}$.

Figure 3 shows the results of matrices with rank 8 and 128, which were fine-tuned on the RTE or RSCD tasks under "full" mode. For the RTE task, the singular values of the rank 8 matrix are similar in magnitude to the first few singular values of the rank 128 matrix, and both exhibit a relatively dispersed distribution. In contrast, for the RSCD, the singular values of the rank 128 matrix are more concentrated, with a narrower range of magnitudes. Analyzing from the perspective of singular vector subspace, the similarity of the first major column singular direction of the rank 8 adaptation matrix for the RTE task with all column singular directions of the rank 128 adaptation matrix is greater than 0.5, indicating a high overlap in their subspaces. In contrast, for the RSCD task, the similarity among all column singular directions is low.

The perspective of singular value and singular vector subspace jointly illustrate that for text comprehension task, the information overlap between low-rank and high-rank matrices is substantial. The high-rank matrix does not provide more effective information. In contrast, for the image classification task in "full" mode, the high-rank matrices yield better performance. The high-rank matrices contain more effective information so that cannot be equivalently replaced by low-rank matrices.

Overall, for tasks such as RTE that emphasize text understanding, performance primarily hinges on the inherent capabilities of the pre-trained model. Fine-tuning mainly serves to adapt the model's output to align with specific tasks, which is a relatively straightforward process. However, for more complex tasks, particularly when the pre-trained model lacks exposure to similar data, efficient fine-tuning methods often encounter performance bottlenecks due to the limited number of trainable parameters compared to full-parameter fine-tuning. Our supplementary experiments in Appendix D further validate the conclusions of our analysis.

4.6 Ablation Study

In this section, we conducted ablation study to examine the impact of individual component of our method. All experiments were performed on the STSB and RTE tasks. We maintained the

Table 5: Freeze A or B on RTE task and the STSB task respectively.

RANK	Parameters	RTE	STSB
16(standard)	110K	84.2	92.0
16(froze A)	61K	78.2	91.0
16(froze B)	61K	*58.7*	88.8
64(froze A)	246K	79.9	91.4
64(froze B)	246K	*60.3*	90.3

Table 6: Train only C matrix with different rank on RTE task and STSB task.

RANK	Parameters	RTE	STSB
16(standard)	110K	84.2	92.0
8	3K	*54.4*	77.7
16	12K	*55.8*	82.5
64	196K	*62.6*	89.8
128	786K	*62.7*	88.3

hyperparameters used in previous experiments and only modified the component under investigation for each test. Each experiment was conducted with 5 random seeds, and we reported the mean results.

Firstly, we freezed A to train the B and C matrices, and then we freezed B to train the A and C matrices to investigate the role of A and B matrices in the model. Based on the experimental results in Table 5, we find that both freezing A or B matrices cause different degrees of performance degradation. Raising the rank still cannot make up for the degradation of performance. Freezing the A matrix has less impact compared to freezing the B matrix, which indicates that the B matrix plays a more important role than the A matrix in SeRA. Freezing the A matrix still allows parameter adjustments through the C and B matrices, whereas freezing the B matrix leads to huge loss of information during the upscaling of vectors. This observation is consistent with the findings of Zhang et al. (2023a).

Secondly, we investigated the role of C matrix in SeRA. Specifically, we frozed the A and B matrices, trained only the C matrix and experimented with several different rank. This process essentially utilizes random weights and projections like VeRA (Kopiczko et al., 2024). The B matrix is initialized using the same strategy as the A matrix. Table 6 shows the results. Freezing A and B matrices at the same time causes a severe performance degradation. Increasing the rank of the matrices still cannot reach the performance of the standard method. This indicates that matrix C needs to be trained in conjunction with matrices A and B to exert its function. In the RTE task, we observe that certain random seeds lead to unstable training, with problematic results highlighted in *Italics*. This suggests that improper initialization of the random matrix can cause difficulties in model training.

5 Conclusion

In this paper, we propose a scalable and efficient PEFT method, called SeRA. It saves a large number of trainable parameters by segmenting the input vectors into multiple sub-vectors then calculating them separately, and recombines the information of sub-vectors by introducing a square matrix to improve the expressive power of the method. The performance of SeRA is demonstrated in a wide range of experiments. We show that different tasks have different requirements on the number of trainable parameters, and analyze the reasons from the perspective of singular value and singular vector subspace. Our method can be combined with a variety of advanced training strategies, such as AdaLoRA (Zhang et al., 2023b) and LoRA+ (Hayou et al., 2024). The variants of LoRA are rich and diverse. Investigating how to combine multiple advanced methods to ultimately come up with a widely applicable and efficient method is a promising research direction in the field of PEFT. In addition to the visual and textual domains, in this paper, we have also conducted experiments with SeRA in the multimodal domain with excellent performance. We haven't completely analyzed the reasons. It is worthwhile to continue exploring and investigating the superiority of this method in the multimodal domain.

References

AI@Meta. Llama 3 model card, 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

Min Cao, Shiping Li, Juntao Li, Liqiang Nie, and Min Zhang. Image-text retrieval: A survey on recent research and development, 2022.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Sanghyuk Chun, Seong Joon Oh, Rafael Sampaio de Rezende, Yannis Kalantidis, and Diane Larlus. Probabilistic embeddings for cross-modal retrieval, 2021. URL https://arxiv.org/abs/2101.05068.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers, 2020. URL https://arxiv.org/abs/1911.03584.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019. URL https://arxiv.org/abs/1903.00161.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models, 2024. URL https://arxiv.org/abs/2402.12354.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. URL https://arxiv.org/abs/1502.01852.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019. URL https://arxiv.org/abs/1902.00751.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. Mora: High-rank updating for parameter-efficient fine-tuning, 2024. URL https://arxiv.org/abs/2405.12130.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation, 2024. URL https://arxiv.org/abs/2310.11454.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024. URL https://arxiv.org/abs/2402.09353.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft, 2022.

OpenAI. Gpt-4 technical report, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari. What's hidden in a randomly weighted neural network? In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11890–11899, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society. doi: 10.1109/CVPR42600.2020.01191. URL https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01191.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. Melora: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning, 2024. URL https://arxiv.org/abs/2402.17263.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. doi: 10.1109/cvpr.2015.7298682. URL http://dx.doi.org/10.1109/CVPR.2015.7298682.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, and etc. Adam Fisch. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023. URL https://arxiv.org/abs/2206.04615.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019. URL https://arxiv.org/abs/1804.07461.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation, 2024. URL https://arxiv.org/abs/2406.11909.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020. URL https://arxiv.org/abs/1906.08237.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning, 2023a. URL https://arxiv.org/abs/2308.03303.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023b. URL https://arxiv.org/abs/2303.10512.

Tong Zhao, Junxiang He, Jingcheng Lv, Delei Min, and Yintao Wei. A comprehensive implementation of road surface classification for vehicle driving assistance: Dataset, models, and deployment. *IEEE Transactions on Intelligent Transportation Systems*, 24(8):8361–8370, 2023. doi: 10.1109/TITS.2023.3264588.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. Lora-drop: Efficient lora parameter pruning based on output evaluation, 2024. URL https://arxiv.org/abs/2402.07721.