RAPA: RECURSIVELY ALIGNED PATHWAY ADAPTATION OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-Efficient Fine-Tuning (PEFT) adapts large language models (LLMs) by training only a small fraction of parameters. Adapter-based approaches reduce compute per step but introduce practical overhead from the additional adapter path (e.g., extra kernel launches and activation storage). Adapter-free approaches avoid this structural overhead by directly updating pretrained weights; however, per-layer random index selection can fragment the trainable subspace, attenuating gradient flow and limiting accuracy. We propose **Recursively Aligned Pathway Adaptation (RAPA)**, an adapter-free PEFT method that forms index-consistent pathways through depth. RAPA follows two principles: (i) selecting balanced submatrices that maximize the number of weights alignable across layers, and (ii) recursively aligning these indices across layers and residual connections. In experiments, RAPA matches or surpasses strong PEFT baselines across most benchmarks while preserving adapter-free efficiency with minimal memory and compute overhead. Code is available at `https://anonymous.4open.science/r/rapa`.

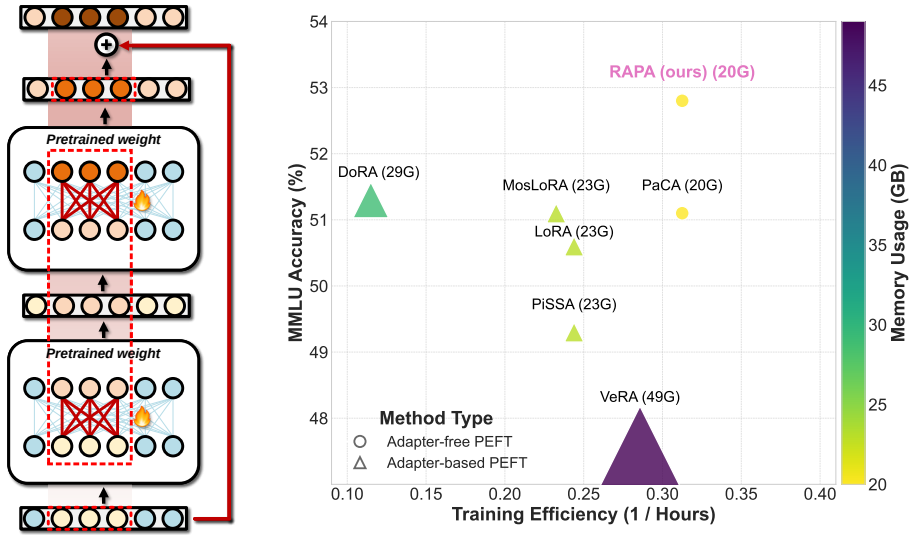
1 INTRODUCTION

Large Language Models (LLMs) now match or surpass expert-level performance in many natural language processing tasks (Zhao et al., 2024b; Team et al., 2024; Dubey et al., 2024; DeepSeek-AI et al., 2025; Jiang et al., 2024a; Chowdhery et al., 2023). In particular, to specialize these LLMs for downstream applications, fine-tuning is commonly employed to adjust their parameters using task-specific data (Wolpert & Macready, 1997). However, fine-tuning these large-scale models for specific tasks demands substantial computational resources and memory (Brown et al., 2020; Achiam et al., 2023; Kaplan et al., 2020), posing significant challenges to practical applications.

To address these challenges, a wide range of Parameter-Efficient Fine-Tuning (PEFT) schemes have been proposed, aiming at reducing computational costs and memory footprints required during fine-tuning. Representative techniques such as Adapter Tuning (Houlsby et al., 2019; Pfeiffer et al., 2020), Low-Rank Adaptation (LoRA) (Hu et al., 2022) and its variants (Liu et al., 2024; Wu et al., 2024a) significantly improve efficiency by freezing pre-trained weights and introducing a small number of trainable adapter parameters.

In theory, adapters introduce only a small amount of additional computation, but they can result in significant training-time overhead in practice; the base and adapter paths are executed sequentially, and computing gradients for adapter parameters typically requires retaining full input activations, which can increase kernel launches and activation memory. By contrast, adapter-free approaches do not require the adapter modules and hence may avoid these structural sources of overhead, at the expense of different trade-offs: BitFit (Zaken et al., 2021b) and LayerNorm tuning (Zhao et al., 2024a) prioritize simplicity and stability over capacity, and PaCA (Woo et al., 2025) selects trainable coordinates at the layer level without explicit cross-layer structure, limiting expressiveness and slowing down convergence in some settings. Our study focuses on characterizing these trade-offs and designing an improved PEFT approach.

In this paper, we first analyze the sources of training overhead in widely used adapter-based methods. We then introduce **Recursively Aligned Pathway Adaptation (RAPA)**, an adapter-free PEFT algorithm that selects and aligns indices across all layers so that the updated weights form a consistent gradient pathway (Figure 1(a)). By aligning the trainable coordinates through depth (and, when applicable, choosing shape-balanced submatrices), RAPA aims to enhance representational capacity



(a) RAPA **(Ours)**

(b) Comprehensive PEFT Comparison

Figure 1: Performance comparison of proposed RAPA method with other PEFTs. **(a)** 5-shot MMLU accuracy compared with other PEFT methods on different tuning parameter budgets. **(b)** Comprehensive comparison of PEFT methods across accuracy, efficiency, and memory usage, highlighting RAPA's overall superiority.

and stabilize gradient flow while avoiding both adapter-induced overhead and per-layer re-sampling costs.

We evaluate RAPA across diverse models and tasks. Specifically, we conduct 5-shot MMLU fine-tuning (Hendrycks et al., 2020) (Figure 1(b)), commonsense reasoning benchmarks, and instruction tuning followed by MT-Bench evaluation. All experiments are run on a single NVIDIA A100 80 GB GPU under matched trainable-parameter budgets and comparable optimization settings to ensure fair comparison with prior PEFT baselines. Full datasets, hyperparameters, and evaluation protocols are provided in the experimental setup for reproducibility.

2 Background and Motivation

2.1 Challenges in Adapter-based PEFT

Fine-tuning LLMs typically incurs significant memory and computation costs. This is mainly because back-propagation requires storing all intermediate activations and updating a massive set of parameters. These requirements hinder practical deployment of LLMs, especially on resource-constrained hardware.

To address these challenges, PEFT methods aim to reduce the number of trainable parameters while maintaining task accuracy. One widely used approach is LoRA (Hu et al., 2022), which introduces additional rank-constrained matrices to the existing pre-trained weights as below:

$$W = W_0 + \Delta W = W_0 + s \cdot BA$$

where $A \in \mathbb{R}^{r \times d}$, $B \in \mathbb{R}^{d \times r}$, and s is a scaling factor. While LoRA reduces the number of trainable parameters, it still requires storing all input activations in memory, as the gradients for the low-rank adapter matrices must be computed with respect to the original input activation. In addition, LoRA introduces sequential computation steps and hence suffers from latency overhead during training; the forward propagation of LoRA proceeds sequentially through the pre-trained weights and the low-rank adapter, which hinders hardware-level parallelism. It was reported that LoRA incurs approximately 33% training-time overhead compared to full fine-tuning, as the forward pass must compute both the pre-trained weight and the low-rank adapter in series rather than in parallel (Hu et al., 2022; Woo et al., 2025). In addition, its low-rank structure limits representational capacity on complex tasks.

2.2 Approaches to Weight Selection in Adapter-Free PEFT

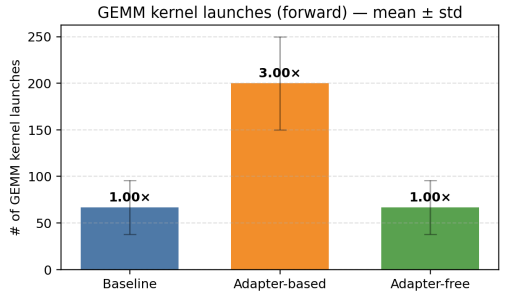
The central challenge in adapter-free PEFT is selecting which subset of weights to update. Existing strategies range from simple random selection, which prioritizes speed and simplicity (Woo et al., 2025), to importance-based approaches that score weights via gradient magnitudes or activation sensitivities. However, these importance-driven methods typically require a costly precalibration step to compute these scores, introducing additional overhead and potential sensitivity to the calibration data (He et al., 2025).

In this work, we propose a *calibration-free, structure-aware selection* strategy that aligns trainable coordinates across layers (Section 3.2). By leveraging the inherent model architecture, our method avoids precomputation overhead while yielding tight seed-to-seed variability (Appendix A) and achieving high performance across tasks.

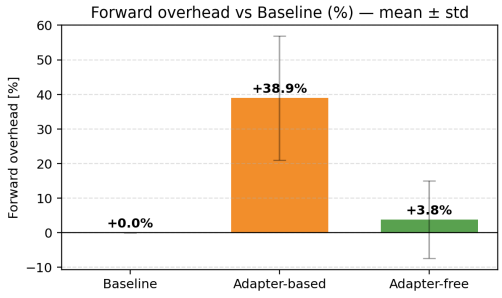
3 Our Work

In this section, We first characterize the training-time overhead of adapter-based PEFT. We then introduce *Recursively Aligned Pathway Adaptation (RAPA)*, a calibration-free, structure-aware weight selection method. The alignment is applied *recursively* across all layers—mirroring the top-down flow of backpropagation—to form coherent pathways. This is achieved by tuning balanced square submatrices of pretrained weights, enforcing cross-layer consistency in the chosen coordinates, and maximizing shared indices across layers under a fixed parameter budget. In experiments, RAPA achieves faster convergence and higher accuracy while preserving adapter-free training efficiency and avoiding calibration or per-layer resampling overhead.

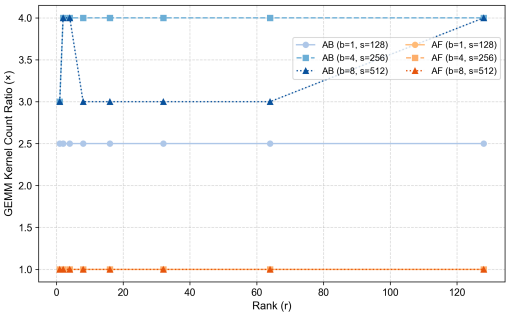
3.1 Revisiting the Overhead of Adapter-Based PEFT



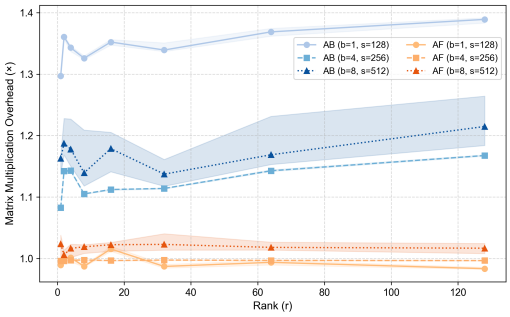
(a) Forward GEMM kernel launches.



(b) Forward-time overhead.



(c) GEMM kernel count ratio.



(d) GEMM-time overhead.

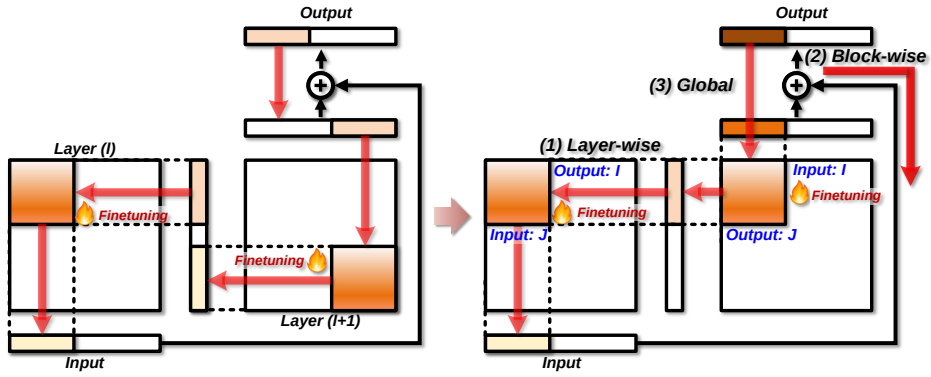
Figure 2: **(a)** Forward GEMM kernel launches. Adapter-based (AB) issues $\sim 3\times$ more kernels than the baseline. **(b)** Forward-time overhead. AB increases latency by $\sim 40\%$, whereas adapter-free (AF) closely matches baseline. **(c)** Kernel-count ratio (AB/baseline). The overhead pattern is largely insensitive to adapter rank r. **(d)** GEMM-time ratio (AB/baseline). Overhead depends more on batch size b and sequence length s.

To motivate our adapter-free (AF) approach, we conduct a controlled microbenchmark to isolate the sources of overhead in adapter-based (AB) methods. On a single linear layer (4096×4096, `bfloat16`) using an NVIDIA A100 (80 GB) GPU, we compare an AB setup (LoRA (Hu et al., 2022)) against an AF setup and a frozen baseline, keeping the trainable parameter budget identical for both AB and AF. We measure throughput with CUDA events and collect CUDA-kernel statistics using `torch.profiler`; results are aggregated over multiple trials (median, with 95% bootstrap confidence intervals).

As illustrated in Figure 2(a), the AB method substantially increases the number of General Matrix Multiplication (GEMM) kernel launches—by over $3\times$ on average, which corresponds to a forward-time overhead of $\sim 40\%$ (Figure 2(b)). In contrast, AF keeps the kernel count essentially unchanged and incurs only a small overhead ($\sim 4\%$).

These results indicate that the dominant inefficiency is not added FLOPs but the proliferation of small, sequential kernel launches that underutilize the GPU (Shi et al., 2016). In Figure 2(c), the kernel-count ratio is largely insensitive to adapter rank r. By contrast, Figure 2(d) shows that the GEMM-time ratio depends more on batch size b and sequence length s. Simply scaling b or s to improve utilization is often impractical due to activation-memory growth in deep models. This structural overhead of separate adapter modules motivates our method RAPA, which aims to retain high performance without incurring this computational burden.

3.2 RAPA: Recursively Aligned Pathway Adaptation



(a) Random index selection (b) Recursively Aligned Pathway **(Ours)**

Figure 3: Comparison between the naive random index selection and the proposed recursively aligned fine-tuning. **(a)** The naive method randomly selects indices for fine-tuning, resulting in partially frozen weights that disrupt gradient flows and lead to inefficient convergence. **(b)** The proposed approach selects square-shaped submatrices for fine-tuning, structurally aligning connections across layers to maintain continuous gradient propagation and accelerate loss convergence.

3.2.1 Balanced Submatrix Selection

Let $W \in \mathbb{R}^{d\times d}$ be a pretrained weight matrix. LoRA has rank r, updating $2 \times r \times d$ parameters. We keep the same parameter budget but reshape it into a $k \times k$ square block:

$$\#\,\text{Trainable Params: } k^2 = 2\,r\,d \quad \Longrightarrow \quad k = \sqrt{2\,r\,d}, \qquad \text{rank}_{\text{balance}} = k > r \;\; (\text{when } r < d). \tag{1}$$

We propose selecting a square-shaped block of weights of size $k \times k$ when tuning k^2 weights (Eq. 1). Thus the rank increases from r to $\sqrt{2\,r\,d}$ without affecting the rest of the matrix. This rank expansion typically endows the fine-tuned subspace with greater expressive capacity, enabling the adapter to capture more complex task-specific patterns (Jiang et al., 2024b).

Since we still update exactly $2\,r\,d$ parameters, the per-step FLOPs and optimizer memory are unchanged; only their 2-D locations differ. Unlike column-only or row-only selections, this square-shaped selection balances input and output dimensions and leads to an increased rank of the fine-tuned subspace. As a result, it provides richer expressiveness.

Experimental results in Section 4.1 demonstrate a steady improvement in accuracy as the row-to-column ratio approaches a balanced state. Optimal scores are achieved within a range closely centered around, but not precisely at, 1:1 ratio. This trend confirms that increasing the rank of the fine-tuned subspace is a critical factor in improving fine-tuning accuracy. Furthermore, balanced-shaped selection maximizes the opportunity for index alignment with adjacent layers, thereby reinforcing coherent cross-layer learning pathways.

3.2.2 Structural Index Alignment: Optimizing Gradient Flow

RAPA enhances training dynamics by structurally aligning the indices—the selected row and column positions—of fine-tuned weights across layers. This strategy promotes coherent gradient propagation by reducing discontinuities, which accelerates convergence.

Throughout this paper, we use the term *block* to denote a Transformer sub-module, namely an attention block or a feed-forward network (FFN) block. For notational clarity, we model each block as **two consecutive linear layers** whose input and output are linked by a residual connection (He et al., 2016b;a).

In this section, we describe three types of index alignment strategies:

- **Layer-wise Alignment:** Align output indices of a layer with input indices of the next.
- **Block-wise Alignment:** Align indices across both linear layers and the residual connection inside each attention or FFN block.
- **Global Alignment:** Align indices consistently across multiple residual-connected blocks.

Method. An overview of the index alignment method is provided in Figure 3. Since we update only a $k \times k$ sub–matrix per layer, *which weights are selected* determines whether the back-propagated gradient can continue propagating through trainable paths or is blocked by frozen weights. The core idea is therefore simple:

(i) Choose a fixed index set for the trainable subspace.

(ii) Apply it consistently across all residual-compatible layers.

This single decision simultaneously enforces *layer*, *block*, and *global* alignment. To implement this strategy, we begin by selecting a fixed index set $I, J \subset \{1, \ldots, d\}$ of size k from the hidden dimension d. In each transformer layer l, we update only the square of the weight matrix $W^{(l)}_{I,J}$.

$$x^{(l+1)}_{I} = W^{(l)}_{I,J} x^{(l)}_{J}, \quad x^{(l+2)}_{J} = x^{(l)}_{J} + W^{(l+1)}_{J,I} x^{(l+1)}_{I} \tag{2}$$

As shown in Eq. 2, the layer-wise alignment is achieved by selecting the same index set across two consecutive layers, highlighted in blue. The red-colored indices represent block-wise alignment, which matches the index sets of the input at layer l and the output at layer $l+2$, taking into account the residual connection within the block.

We express global alignment across residual-connected blocks as

$$x^{(b+1)}_{J} = x^{(b)}_{J} + W^{(b+1)}_{J,I}\left(W^{(b)}_{I,J}, x^{(b)}_{J}\right) = x^{(b)}_{J} + \mathcal{F}\left(W^{(b)}_{J,J}, x^{(b)}_{J}\right) \tag{3}$$

$$\Longrightarrow x^{(b+m)}_{J} = x^{(b)}_{J} + \sum_{j=0}^{m-1} \mathcal{F}\Big(W^{(b+j)}_{J,J}, x^{(b+j)}_{J}\Big) \tag{4}$$

Eq. 3 describes the forward propagation of a block indexed by b, and its generalization to m consecutive blocks in Eq. 4, where $\mathcal{F}(\cdot)$ denotes the transformation through the two linear layers within each block. These alignment strategies enhance direct gradient propagation through residual shortcuts and inter-layer connections.

As a result, Figure 4 shows that the training loss converges noticeably faster when the trainable weight indices are *aligned*. As demonstrated in Table 1, alignment not only accelerates convergence but also improves final accuracy. To confirm the robustness of this trend, we present the detailed training loss curves across multiple random seeds in Appendix C. The full strategy is summarized in Appendix E.

Figure 4: Comparison of training loss curves for index alignment strategies.

Table 1: Effect of index alignment strategies on 5-shot MMLU accuracy.

Method	Hums.	STEM	Social.	Other	Avg.
Random	48.2	41.3	60.2	59.4	51.7
Layer Align	49.3	42.1	59.8	58.4	52.0
Block Align	49.4	41.5	59.9	57.9	51.8
Block+Global	49.3	42.1	60.3	59.1	**52.3**

Why alignment works. We provide a simple back-propagation analysis to illustrate how the index alignment method facilitates more effective loss convergence. The chain rule for any trainable element of $W^{(l)}_{I,J}$ yields

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}_{I,J}} = \frac{\partial \mathcal{L}}{\partial x^{(l+2)}_{J}} \, W^{(l+1)}_{J,I} \, x^{(l)}_{J}. \tag{5}$$

As shown in Eq. 5, when index alignment includes $W^{(l+1)}_{J,I}$ in the trainable subspace, its update at layer $l{+}1$ is immediately reflected in the gradient of layer l. This leads to more consistent gradient directions and faster convergence.

We further strengthen this connection by choosing the same set J for both the block input and the output connected via the residual path. This yields

$$\frac{\partial \mathcal{L}}{\partial x^{(l)}_{J}} = \frac{\partial \mathcal{L}}{\partial x^{(l+2)}_{J}} \left(1 + W^{(l+1)}_{J,I} W^{(l)}_{I,J}\right) \tag{6}$$

where the identity term in Eq. 6 arises from the residual shortcut. The loss gradient at the block output $x^{(l+2)}_{J}$ is directly reflected in the gradient of the block input $x^{(l)}_{J}$ through the residual connection. As a result, the model can respond to the loss signal more quickly, leading to faster convergence.

Finally, we propagate the alignment globally across blocks by fixing the index set J and reusing it as both the input and residual-output weights in every residual block.

$$\frac{\partial \mathcal{L}}{\partial x^{(b)}_{J}} = \underbrace{\frac{\partial \mathcal{L}}{\partial x^{(b+m)}_{J}}}_{\text{direct shortcut}} + \frac{\partial \mathcal{L}}{\partial x^{(b+m)}_{J}} \frac{\partial}{\partial x^{(b)}_{J}} \sum_{j=0}^{m-1} \mathcal{F}\left(W^{(b+j)}_{J,J}, \, x^{(b+j)}_{J}\right) \tag{7}$$

Back-propagating through m such blocks results in Eq. 7. The equation above suggests that the gradient at block $b+m$ reaches block b both directly through the residual shortcut and through the aligned subspace shared by all trained blocks (a detailed analysis is provided in Appendix B).

4 Experiments

We conducted comprehensive experiments to evaluate the effectiveness and generalization capability of RAPA, along with the impact of critical design decisions. All experiments were performed on a single NVIDIA A100 80GB GPU, using 16-bit mixed precision (Micikevicius et al., 2017) and the AdamW (Loshchilov & Hutter, 2017) optimizer unless otherwise specified.

We measured accuracy on the MMLU benchmark and on eight commonsense reasoning tasks (Clark et al., 2018; 2019; Zellers et al., 2019; Mihaylov et al., 2018; Bisk et al., 2020; Sap et al., 2019; Sakaguchi et al., 2020). Additionally, we conducted instruction tuning and evaluated the conversational capabilities of the model using the MT-Bench benchmark (Zheng et al., 2023). We also analyzed the effects of weight shape selection and index alignment strategies introduced in Section 3.2. Detailed experimental settings and hyperparameter configurations are provided in Appendix F.

4.1 Effects of Balanced Weight Selection and Alignment

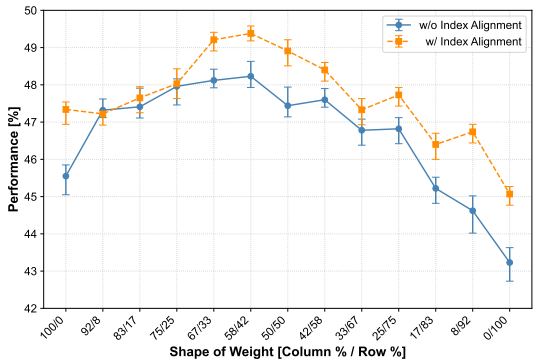


Figure 5: Effects of weight shape (row/column ratio) and index alignment on MMLU accuracy.

We investigated how the row-to-column ratio of trainable weights influences task accuracy and how this relationship changes after index alignment is applied. Figure 5 plots MMLU accuracy as a function of the selected row-to-column ratio on x-axis, comparing cases without index alignment (blue curve) and with index alignment (yellow curve). We fine-tuned the LLaMA2-7B model (Touvron et al., 2023) on the MMLU dataset for this analysis; the detailed experimental setup is provided in Appendix F (Table 7).

As shown in Figure 5, selecting a symmetric subset of weights yields higher task accuracy compared to extreme cases where only row or column weights are selected. Focusing on the random selection case (blue curve), a balanced (50/50) ratio achieves an accuracy of 47.4%, outperforming the extreme column-only case (100/0) and row-only (0/100) configurations by 1.8% and 4.2%, respectively.

This trend remains consistent when the **index alignment method** is applied: balanced selection again achieves the highest accuracy. Moreover, the aligned variant consistently outperforms the random baseline across all settings. In the balanced case, index alignment improves performance to 48.9%, 1.5% higher than the random selection method at 47.4%. Even in the column-only setting, alignment increases accuracy from 45.6% to 47.3%, and in the row-only setting, from 43.2% to 45.1%, confirming that alignment enhances performance even under asymmetric or suboptimal weight configurations.

4.2 Fine-tuning for Downstream Task

Table 2: Comparisons of memory usage (Mem), training time (Time), and 5-shot accuracy on MMLU dataset when fine-tuning the LLaMA2-7B and LLaMA2-13B models using various PEFT algorithms.

Model	Method	# Params	Mem	Time	Accuracy (%)				
					Hums.	STEM	Social.	Other	Avg.
LLaMA2-7B	Baseline	-	-	-	43.6	37.1	51.4	53.0	45.9
	LoRA	319.8M	53.4G	3.3h	50.6	43.3	61.9	59.5	53.4
	DoRA	321.1M	75.3G	6.2h	51.8	43.1	61.9	60.1	53.9
	MosLoRA	323.5M	53.4G	3.5h	50.7	42.4	60.5	58.2	52.6
	PaCA	348.1M	41.7G	2.8h	50.7	43.3	63.5	60.1	53.9
	RAPA (Ours)	319.6M	41.5G	2.9h	**51.4**	**44.4**	**64.3**	**60.2**	**54.6**
LLaMA2-13B	Baseline	-	-	-	53.2	44.2	62.8	60.8	55.0
	LoRA	500.7M	60.3G	5.9h	56.4	47.5	**67.9**	64.2	58.6
	MosLoRA	505.3M	60.6G	6.1h	55.5	46.0	67.4	64.2	57.9
	DoRA	502.8M	80.3G	12.0h	57.0	47.8	67.6	64.6	58.9
	PaCA	545.3M	52.7G	5.1h	57.2	47.5	67.5	65.1	59.0
	RAPA (Ours)	501.5M	52.9G	5.2h	**57.9**	**48.1**	67.6	**65.9**	**59.6**

We validate the task accuracy and efficiency of RAPA using the MMLU 5-shot benchmark, which evaluates complex reasoning across 57 academic subjects. RAPA was compared against several PEFT baselines, including LoRA (Hu et al., 2022), DoRA (Liu et al., 2024), MosLoRA (Wu et al., 2024a), as well as PaCA (Woo et al., 2025), using the MMLU dataset (Hendrycks et al., 2020). All

PEFT schemes were evaluated on both LLaMA2-7B and LLaMA2-13B. For fair comparisons, the LoRA baseline was configured with a rank of 128, and RAPA was adjusted to match the trainable parameter count. The results from additional experiments across various rank settings are reported in Appendix D, and the detailed configurations are provided in Appendix F.

On the MMLU benchmark, RAPA consistently outperforms other PEFT methods (Table 2). It achieves higher accuracy than PaCA (+0.7 % on LLaMA2-7B, +0.6 % on LLaMA2-13B) with similar efficiency, and surpasses LoRA and MosLoRA in both accuracy and training speed. While DoRA reaches competitive accuracy, it exhibits significantly longer training time, highlighting the computational efficiency of RAPA. Among all compared methods, RAPA achieves the highest average accuracy on both models, demonstrating the effectiveness of the structured weight selection and index alignment strategies proposed in this work.

4.3 Commonsense Reasoning

Table 3: Comparison of PEFT methods on commonsense reasoning benchmarks.

Method	Mem	Time	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	SIQA	WinoG.	Avg.
Baseline	-	-	43.3	74.6	77.7	76.0	44.2	79.1	46.1	69.1	63.8
LoRA	53.2G	4.8h	43.0	72.6	80.6	67.8	44.6	76.6	54.9	74.3	64.3
DoRA	75.3G	8.8h	52.1	80.0	73.7	76.4	49.0	80.4	58.4	82.0	69.0
MosLoRA	53.4G	5.0h	53.0	81.2	70.7	76.3	50.2	80.8	58.8	82.1	69.1
PaCA	41.5G	4.0h	51.0	78.5	73.7	75.4	48.8	80.2	57.5	80.3	68.2
RAPA (Ours)	42.1G	4.1h	**53.1**	**81.1**	**76.4**	**77.1**	**49.8**	**80.4**	**58.9**	**80.8**	**69.7**

We further evaluated RAPA on eight canonical commonsense reasoning tasks (ARC Challenge, ARC Easy, BoolQ, HellaSwag, OpenbookQA, PIQA, Social IQA, and Winogrande), comparing its accuracy against LoRA, DoRA, MosLoRA, and PaCA. All experiments were conducted on the LLaMA2-7B model and we fine-tuned it on a commonsense reasoning dataset consisting of 170k samples. Detailed hyperparameters are listed in Appendix F.

Table 3 reports that RAPA achieves the highest macro-average accuracy of 69.7%, outperforming the strongest adapter-based baseline, DoRA (69.0%), by 0.7% and the strongest adapter-free baseline, PaCA (68.2%), by 1.5%.

Despite higher accuracy, RAPA keeps the computational footprint essentially flat: 42.1 GB peak GPU memory usage and 4.1 hour of wall-clock time, nearly identical to PaCA (41.5GB and 4.0 hour) and approximately 20% lower than those of LoRA and MosLoRA (53 GB), while requiring less than half the GPU time compared to DoRA (8.8 hours). Since RAPA remains adapter-free, its per-step FLOPs and optimizer state size are unchanged relative to PaCA.

These results reinforce the findings from the experiments on MMLU and underscore two key benefits of combining structured square selection with cross-layer index alignment: (i) a higher rank under a fixed parameter budget, which expands the hypothesis space sufficiently to model diverse commonsense reasoning tasks, and (ii) uninterrupted gradient flow across layers, which accelerates convergence without incurring additional memory overhead. Together, these results demonstrate that principled subspace design, rather than simply increasing parameter count, is a key to generalization performance in low-rank, adapter-free fine-tuning.

4.4 Instruction Tuning

We evaluated RAPA on the Mistral-7B model, which was instruction-tuned for one epoch on the OASST1 dataset using a single NVIDIA A100 GPU. Performance was subsequently measured using the MT-Bench benchmark, which averages scores over two conversational turns. We used GPT-4o-mini as the evaluator for this benchmark (see Table 10 in Appendix F for further details).

As shown in Table 4, RAPA achieved the highest average score of 4.84. Notably, compared to PaCA, another adapter-free method with nearly identical training efficiency, outperforms PaCA by almost 0.9 points (4.84 vs. 3.96). This demonstrates that RAPA overcomes the performance limitations of prior adapter-free approaches while also outperforming strong adapter-based methods like LoRA

(4.55) and DoRA (4.63). Furthermore, this high performance is achieved with remarkable efficiency, requiring less than half the training time of DoRA.

Table 4: Performance comparison of instruction tuning on Mistral-7B (Jiang et al., 2023).

Model	Method	Time	Mem	# Params	Human	STEM	Role	Extract	Writing	Reason	Coding	Math	Avg.
Mistral-7B	Baseline	-	-	-	1.85	3.20	2.95	3.15	2.50	1.70	2.30	1.40	2.38
	LoRA	47m	52G	168M	6.40	5.75	4.35	**5.20**	5.25	4.20	3.30	1.95	4.55
	DoRA	93m	69G	169M	**6.70**	5.60	4.30	4.40	5.75	**4.50**	**3.85**	1.95	4.63
	MosLoRA	48m	52G	169M	5.80	**6.05**	4.85	5.00	5.00	3.75	3.65	2.00	4.51
	PaCA	41m	47G	176M	5.35	5.45	4.15	4.50	4.55	2.55	3.20	1.95	3.96
	RAPA (Ours)	42m	48G	168M	**6.70**	5.82	**5.65**	**5.20**	**5.85**	4.20	3.22	**2.05**	**4.84**

5 Related work

Parameter-Efficient Fine-tuning (PEFT) A range of PEFT methods have been proposed to adapt large pre-trained models to downstream tasks while minimizing the number of trainable parameters (Han et al., 2024; Zaken et al., 2021a; Liu et al., 2022). Early approaches introduced adapter modules within each transformer layer (Houlsby et al., 2019; Pfeiffer et al., 2020), inserting small bottleneck networks that are trained while keeping the backbone frozen. Although being effective, these methods incur both training and inference overheads due to the additional network components. To address the inference inefficiency, LoRA (Hu et al., 2022) was proposed, which employs low-rank decomposed matrices during training and merges them back into the original weights during inference. This scheme enables LoRA and its variants (Liu et al., 2024; Wu et al., 2024b) to achieve zero inference-time overhead while maintaining high accuracy.

Parameter-efficient Column-wise Adaptation (PaCA) PaCA (Woo et al., 2025) is a lightweight fine-tuning method that randomly selects and updates a subset of column-wise weights from the pre-trained model without introducing additional adapter modules. Unlike LoRA-based schemes that utilize additional trainable low-rank adapters, PaCA performs adapter-free fine-tuning, thereby reducing both training time and memory consumption. However, the use of random index selection without structural alignment across layers lead to suboptimal gradient propagation and degraded convergence behavior in some cases.

Sparse Matrix Tuning (SMT) SMT (He et al., 2025) suggests gradient-based importance estimation to selectively fine-tune substructures within the pre-trained model, demonstrating improved accuracy and faster convergence compared to conventional PEFT schemes. However, SMT relies on performing full fine-tuning over the entire model for a number of iterations to obtain importance scores, incurring substantial computational and memory overhead. Furthermore, since the importance-based tuning modifies a subset of the full model parameters directly, it necessitates storing the entire model after adaptation, rather than storing only the lightweight adapter modules. This characteristic limits the scalability of SMT in multi-task or multi-domain scenarios (Sheng et al., 2023; Li et al., 2024; Zhao et al., 2024c; Zeng et al., 2025), where deploying only a specialized adapter for each task or domain is a key advantage of PEFT.

6 Conclusion

We presented RAPA, an adapter-free PEFT method that expands capacity and stabilizes optimization without added computational or memory cost. RAPA reallocates a fixed parameter budget to a balanced square submatrix and aligns the corresponding indices across residual-connected layers, yielding a higher rank of the weight update and coherent gradient pathways. The design is motivated by a simple backpropagation analysis and incurs no extra modules or calibration passes. Across MMLU, commonsense reasoning, and MT-Bench, RAPA improves accuracy and convergence under matched budgets while keeping time and memory near the adapter-free baseline. We summarize limitations and broader impacts in Appendices G and H.

ETHICS STATEMENT

Our research aims to make the fine-tuning of large language models more computationally efficient and accessible. By reducing the resources required for parameter-efficient fine-tuning, our method, RAPA, can help democratize access to advanced AI technologies, enabling a wider range of researchers, startups, and organizations to develop specialized models for beneficial applications. We acknowledge that any technology that lowers the barrier to entry for powerful models also carries the risk of misuse. More efficient fine-tuning techniques could potentially be leveraged by malicious actors to generate harmful content, such as high-quality disinformation or spam, at a larger scale. Furthermore, while RAPA is a general optimization method, the models fine-tuned with our technique will inherit any social biases present in the base model and the fine-tuning data. We believe the benefits of enabling broader access for legitimate research and innovation are substantial. We encourage the community to continue developing robust safeguards and responsible deployment practices for fine-tuned models. We have conducted our work in full adherence to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We are committed to the full reproducibility of our research. All experiments were conducted with a rigorous methodology, ensuring fair comparisons across methods under identical trainable-parameter budgets. We report our findings transparently, detailing performance across all benchmarks. To enable other researchers to verify and build upon our work, we provide our source code in the supplementary material. Furthermore, all datasets used are publicly available, and the complete experimental configurations and hyperparameters for each experiment are detailed in Appendix F.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 7432–7439, New York, NY, USA, 2020. AAAI Press. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet

Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

Haoze He, Juncheng B Li, Xuan Jiang, and Heather Miller. SMT: Fine-tuning large language models with sparse matrices. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=GbgCRJedQ7.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645. Springer, 2016b.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024a. doi: 10.48550/ARXIV.2401.04088. URL https://doi.org/10.48550/arXiv.2401.04088.

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. *CoRR*, abs/2405.12130, 2024b. doi: 10.48550/ARXIV.2405.12130. URL https://doi.org/10.48550/arXiv.2405.12130.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, et al. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

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. In *Forty-first International Conference on Machine Learning*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779*, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 8732–8740, New York, NY, USA, 2020. AAAI Press. doi: 10.1609/aaai.v34i05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, et al. S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285*, 2023.

Yang Shi, Uma Naresh Niranjan, Animashree Anandkumar, and Cris Cecka. Tensor contractions with extended blas kernels on cpu and gpu. In *2016 IEEE 23rd International Conference on High Performance Computing (HiPC)*, pp. 193–202. IEEE, 2016.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

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

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Sunghyeon Woo, Sol Namkung, Sunwoo Lee, Inho Jeong, Beomseok Kim, and Dongsuk Jeon. PaCA: Partial connection adaptation for efficient fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=iYkhxre0In.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 7880–7899. Association for Computational Linguistics, 2024a. URL https://aclanthology.org/2024.emnlp-main.450.

Taiqiang Wu, Jiahao Wang, Zhe Zhao, and Ngai Wong. Mixture-of-subspaces in low-rank adaptation. *arXiv preprint arXiv:2406.11909*, 2024b.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021a.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Hanqing Zeng, Yinglong Xia, Zhuokai Zhao, Gilbert Jiang, Qiang Zhang, Jiayi Liu, Lizhu Zhang, Xiangjun Fan, and Benyu Zhang. S'more: Structural mixture of residual experts for llm fine-tuning. *arXiv preprint arXiv:2504.06426*, 2025.

Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. Tuning layernorm in attention: Towards efficient multi-modal llm finetuning. In *ICLR*, 2024a.

Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. Lora land: 310 fine-tuned llms that rival gpt-4, A technical report. *CoRR*, abs/2405.00732, 2024b. doi: 10.48550/ARXIV.2405.00732. URL https://doi.org/10.48550/arXiv.2405.00732.

Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. Lora land: 310 fine-tuned llms that rival gpt-4, a technical report. *arXiv preprint arXiv:2405.00732*, 2024c.

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.

A Robustness of Fine-tuning vs. Initial Weight Sensitivity

Table 5: MMLU 5-shot accuracy comparison. The rows with the lowest (blue) and highest (red) average scores for each method are highlighted.

Method	Setting	Accuracy (%)				
		Hums.	STEM	Social.	Other	Avg.
Baseline	-	43.6	37.1	51.4	53.0	45.9
Pruning to Zero	Seed #1	34.0	31.8	38.5	42.8	36.4
	Seed #2	39.9	35.9	48.3	49.9	43.0
	Seed #3	39.5	38.3	47.4	49.1	43.1
	Seed #4	40.4	37.5	49.6	50.6	44.0
Fine-tuning	Seed #1	50.2	44.1	63.2	60.4	54.0
	Seed #2	51.3	43.2	62.6	60.9	54.1
	Seed #3	51.2	42.6	62.1	60.5	53.7
	Seed #4	51.2	44.3	62.8	60.6	54.3

To determine whether the initial weight sensitivity of a model to a downstream task could predict its final fine-tuning performance, we used the performance degradation from our "Pruning to Zero" experiment as a proxy for sensitivity (Table 5). The results revealed a high variance in sensitivity depending on the random seed; for instance, the accuracy drop from the baseline ranged from a minimal 1.9 % (Seed #4 from the previous table) to a substantial 9.5 % (Seed #1), indicating that different sets of weights with varying importance were pruned.

Notably, we observed no correlation between this initial weight sensitivity and the final fine-tuning accuracy. Despite the wide disparity in the pruning outcomes, all fine-tuning runs converged to a narrow and high-performance bracket (e.g., 53.7 % to 54.3 %). This suggests that the fine-tuning process is highly robust and capable of overriding the initial weight state of the model to find a consistent solution.

B Gradient Analysis of Index Alignment

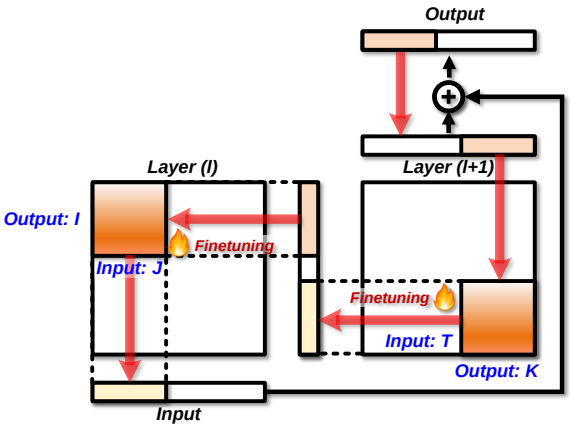


Figure 6: Example of disrupted gradient flow in the absence of index alignment

In this section, we analytically demonstrate the benefit of index alignment in RAPA through a formal mathematical formulation. As defined in Section 3.2.2, we consider a simplified structure in which a single block consists of two consecutive linear layers connected via a residual connection. Let

$I, J, K, T \subset \{1, \ldots, d\}$ denote the selected index subsets over the hidden dimension d, where each subset corresponds to the input/output subspaces involved in fine-tuning.

Mathematically, the forward propagation through a block without applying any alignment can be defined as:

$$x_I^{(l+1)} = W_{I,J}^{(l)} x_J^{(l)} \tag{8}$$

$$x_K^{(l+2)} = W_{K,T}^{(l+1)} x_T^{(l+1)} + x_K^{(l)} \tag{9}$$

B.1 Layer-wise Alignment: Aligning Output-Input Indices across Layers

In layer-wise alignment, the output indices of a layer are matched with the input indices of the next layer to enhance gradient propagation. Based on the forward propagation defined in Eqs. 8 and 9, the loss gradient with respect to the trainable weight $W_{K,T}^{(l+1)}$ at layer $l+1$ is given by

$$\frac{\partial \mathcal{L}}{\partial W_{K,T}^{(l+1)}} = \frac{\partial \mathcal{L}}{\partial x_K^{(l+2)}} \frac{\partial x_K^{(l+2)}}{\partial W_{K,T}^{(l+1)}} \tag{10}$$

The weights corresponding to the selected subspace of the layer $l+1$ are updated during back-propagation as follows:

$$W_{K,T}^{(l+1)} \leftarrow W_{K,T}^{(l+1)} - \eta \frac{\partial \mathcal{L}}{\partial W_{K,T}^{(l+1)}} \tag{11}$$

where η is the learning rate. By applying the chain rule, we can similarly compute the gradient of the selected subspace weights at layer l, $W_{I,J}^{(l)}$ as

$$\frac{\partial \mathcal{L}}{\partial W_{I,J}^{(l)}} = \frac{\partial \mathcal{L}}{\partial x_K^{(l+2)}} \frac{\partial x_K^{(l+2)}}{\partial x_I^{(l+1)}} \frac{\partial x_I^{(l+1)}}{\partial W_{I,J}^{(l)}} \tag{12}$$

where $\frac{\partial x_K^{(l+2)}}{\partial x_I^{(l+1)}} = W_{K,I}^{(l+1)}$ (note that $W_{K,I}^{(l+1)}$ is not included in the selected trainable subspace), and $\frac{\partial x_I^{(l+1)}}{\partial W_{I,J}^{(l)}} = x_J^{(l)}$. Accordingly, the gradient becomes:

$$\frac{\partial \mathcal{L}}{\partial W_{I,J}^{(l)}} = \frac{\partial \mathcal{L}}{\partial x_K^{(l+2)}} \boldsymbol{W_{K,I}^{(l+1)}} \boldsymbol{x_J^{(l)}} \tag{13}$$

The gradient in Eq. 13 reveals that the update of $W_{I,J}^{(l)}$ is modulated by the factor $W_{K,I}^{(l+1)}$ from the subsequent layer. Since Eq. 9 assumes that only the weights indexed by (K, T) are included in the *trainable subspace* of layer $l+1$, the element $W_{K,I}^{(l+1)}$ remains *frozen* during the update step of layer l when $T \neq I$. Consequently, the back-propagated gradient that reaches $W_{I,J}^{(l)}$ is scaled by this fixed weight.

When the index sets are **aligned** ($T = I$), the factor $W_{K,I}^{(l+1)}$ is updated at every iteration, allowing the gradient to flow more directly into $W_{I,J}^{(l)}$ and accelerating convergence, as discussed in Section 3.2.2.

B.2 Block-wise Alignment: Aligning within Residual-connected Layers

The residual connection within the block transmits both activations and gradients directly across layers. Such residual paths preserve gradient magnitude and stabilize the training of deep neural networks (He et al., 2016b).

Gradient decomposition For clarity, we focus on a single residual block, referred to as the *target block*, whose final layer index is $l+2$. The gradient arriving at the output of the target block during back-propagation can be decomposed into two additive parts:

$$\frac{\partial \mathcal{L}}{\partial x_K^{(l+2)}} = {g_K}^{linear} + {g_K}^{residual} \tag{14}$$

where ${g_K}^{linear}$ is the gradient that has passed through all layers after the target block, and ${g_K}^{residual}$ flows along the residual connection.

Why align indices inside the block? As defined in Appendix B.1, the layer-wise alignment method aligns index sets between successive layers ($T = I$).

In this section, to motivate the need for block-wise alignment, we consider a block with a residual connection from input to output. We denote the input index set as J and the output index set as K. When the output index subspace K and input index subspace J of the block are not aligned, the back-propagation gradient expands as follows:

$$\frac{\partial \mathcal{L}}{\partial x_J^{(l)}} = \frac{\partial \mathcal{L}}{\partial x_K^{(l+2)}} \frac{\partial x_K^{(l+2)}}{\partial x_I^{(l+1)}} \frac{\partial x_I^{(l+1)}}{\partial x_J^{(l)}} \tag{15}$$

$$= \frac{\partial \mathcal{L}}{\partial x_K^{(l+2)}} \left(W_{K,I}^{(l+1)} W_{I,J}^{(l)} + \delta_r; J, K \right), \quad \delta_{r;J,K} = \begin{cases} 1 & \text{if } r \in J \cap K \\ 0 & \text{if } r \notin J \cap K \end{cases} \tag{16}$$

where $\delta_{r;J,K}$ is the indicator function. If we align the output and shortcut index sets such that $K = J$, the expression simplifies to:

$$\frac{\partial \mathcal{L}}{\partial x_J^{(l)}} = \frac{\partial \mathcal{L}}{\partial x_J^{(l+2)}} \left(W_{J,I}^{(l+1)} W_{I,J}^{(l)} + \mathbf{1} \right) \tag{17}$$

The additive identity term arises only when the same index set is used on both sides of the residual connection ($K = J$). This identity term creates a direct gradient path from the output of the block back to its input layer (l), as formalized in Eq. 14, effectively shortening the gradient route.

As a result, it helps preserve gradient magnitude and reduces training loss by approximately 3% (see Figure 4 in Section 3.2.2).

B.3 Global Alignment: Index Alignment across Residual Blocks

The benefits of residual connections extend not only within a block but also across multiple sequential blocks. To fully exploit this, we align indices between adjacent blocks. Global alignment extends the index-selection rule of RAPA so that the same index set T is used for fine-tuning in all consecutive blocks.

For forward propagation of block b (refer to Eq. 3 in Section 3.2.2 for notation), we write

$$x_J^{(b+1)} = x_J^{(b)} + \mathcal{F}\left(W_{J,J}^{(b)}, x_J^{(b)} \right) \tag{18}$$

where $\mathcal{F}(\cdot)$ is the two-layer transformation inside the block. Repeating this for m successive blocks yields

$$x_J^{(b+m)} = x_J^{(b)} + \sum_{i=0}^{m-1} \mathcal{F}\left(W_{J,J}^{(b+i)}, x_J^{(b+i)} \right). \tag{19}$$

Applying the chain rule, the gradient back-propagated to the initial input is represented by

$$\frac{\partial \mathcal{L}}{\partial x_J^{(b)}} = \underbrace{\frac{\partial \mathcal{L}}{\partial x_J^{(b+m)}}}_{\text{direct shortcut}} \left[1 + \underbrace{\frac{\partial}{\partial x_J^{(b)}} \sum_{i=0}^{m-1} \mathcal{F}\left(W_{J,J}^{(b+i)}, x_J^{(b+i)} \right)}_{\text{via learned weights}} \right]. \tag{20}$$

As in block-wise alignment, the term "1" directly links the final loss back to the shallow block without traversing intermediate weights. This direct gradient path:

- Shortens the effective back-propagation path length.
- Strengthens early-layer updates.
- Mitigates vanishing gradients even across many blocks.

C Detailed Training Loss Curves

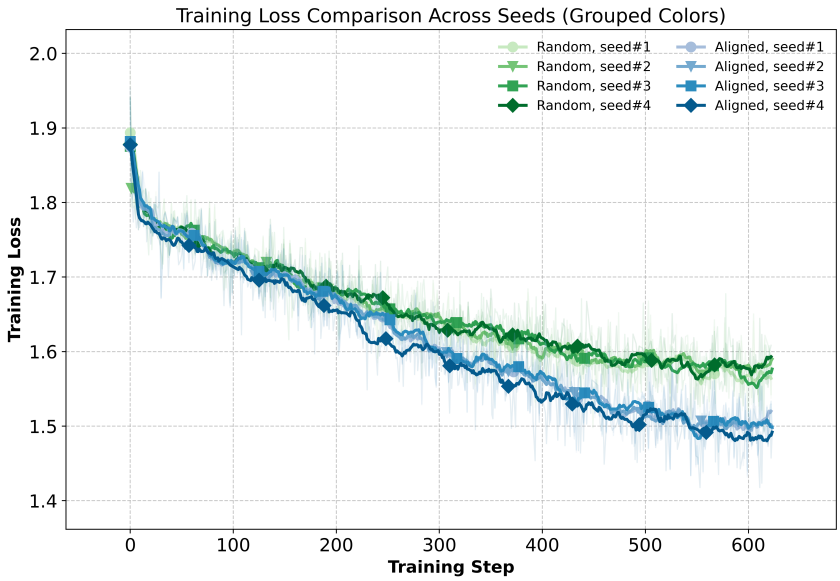


Figure 7: Detailed training loss curves for aligned vs. random index selection across four different random seeds. The grouped colors (blue hues for aligned, green hues for random) and unique markers identify each individual run. This demonstrates that the faster convergence of the aligned method is consistent and robust across various initializations.

D 5-SHOT MMLU ACCURACY COMPARISONS UNDER VARIOUS RANKS

Table 6: 5-shot accuracy (%) on MMLU dataset by rank and method.

Method	Rank	# Params(M)	Accuracy (%)				
			Hums.	STEM	Social.	Other	Avg.
Baseline	-	-	43.6	37.1	51.4	53.0	45.9
LoRA	N/A	-	-	-	-	-	-
DoRA	N/A	-	-	-	-	-	-
MosLoRA	N/A	-	-	-	-	-	-
PaCA	1	1.4M	44.6	38.1	54.0	53.8	47.2
RAPA (Ours)	60	1.4M	**46.7**	**40.6**	**56.3**	**56.4**	**49.6**
LoRA	1	2.5M	46.7	**42.6**	57.9	57.6	50.6
DoRA	1	3.9M	47.5	41.8	58.8	58.0	51.0
MosLoRA	1	2.5M	**48.2**	40.9	**59.8**	58.6	**51.4**
PaCA	2	2.7M	46.5	39.8	55.7	54.7	48.8
RAPA (Ours)	80	2.5M	47.6	41.8	58.8	**58.4**	51.1
LoRA	8	20.0M	48.5	41.2	57.3	56.5	50.6
DoRA	8	21.4M	48.7	42.3	58.3	57.6	51.3
MosLoRA	8	20.0M	46.6	42.2	**60.8**	57.4	51.1
PaCA	16	21.8M	48.7	41.7	58.7	57.6	51.2
RAPA (Ours)	228	20.1M	**50.1**	**42.6**	**60.8**	**59.1**	**52.8**
LoRA	64	159.9M	**51.2**	43.1	61.7	60.1	53.6
DoRA	64	161.3M	51.0	43.6	61.8	60.1	53.7
MosLoRA	64	160.8M	50.4	42.8	61.5	59.5	53.1
PaCA	128	174.1M	50.7	43.3	**63.5**	60.1	53.9
RAPA (Ours)	644	160.1M	**51.2**	**44.3**	62.8	**60.6**	**54.3**

E IMPLEMENTATION DETAILS

Algorithm 1 instantiates *all three* alignment rules in a single pass over the residual blocks.

Algorithm 1 Index Alignment of Trainable Weights

Input: residual blocks $(\mathcal{B}_1, \ldots, \mathcal{B}_L)$, hidden size d, square side k, seed s
$I \leftarrow \textsc{RandomPick}(d, k, s)$
for $b = 1$ **to** L **do**
  $\mathcal{B}_b.\texttt{input} \leftarrow I$ // block-wise and global alignment
  $\texttt{prevOut} \leftarrow I$
  for all linear layer ℓ **in** $\mathcal{B}_b$ **do**
    $\ell.\texttt{input} \leftarrow \texttt{prevOut}$ // layer-wise alignment
    $\ell.\texttt{output} \leftarrow \textsc{RandomPick}(d, k)$
    $\texttt{prevOut} \leftarrow \ell.\texttt{output}$
  end for
  $\mathcal{B}_b.\texttt{output} \leftarrow I$ // last layer output of a block
end for
return mapping $\{\ell.\texttt{input}, \ell.\texttt{output}\}$

F EXPERIMENTAL DETAILS

Table 7: Hyperparameters for the row/column weight ratio sweep experiment on LLaMA2-7B.

Fixed Hyperparameters	
Method	RAPA
Training Precision	16-bit mixed precision
Total Trainable Weights	4,096 per layer
Optimizer	AdamW
LR Scheduler	Cosine
Batch Size	8
Sequence Length	512
Epochs	1
Target Modules	Q, K, V, O layers in Attention

Sweep Configuration

Rank (column) is swept through powers of two from 1 to 4096. Rank (row) is set such that Rank (row) $\times$ Rank (column) = 4096. Below are representative examples.

Rank (column)	**Rank (row)**	**Ratio (column:row)**	**alpha**
1	4096	100/0	4
2	2048	92/8	8
		$\vdots$	
64	64	50/50	64
		$\vdots$	
2048	2	8/92	8192
4096	1	0/100	16384

Table 8: Hyperparameters for fine-tuning LLaMA2-7B and LLaMA2-13B on MMLU using PEFT (parentheses indicate settings for LLaMA2-13B).

Hyperparameters	LoRA	DoRA	MosLoRA	PaCA	RAPA
Training Precision	16-bit mixed precision (Micikevicius et al., 2017)				
Rank	128	128	128	256	910 (1018)
α	64	64	64	128	455 (509)
DropOut	0.1	0.1	0.1	-	-
Optimizer	AdamW (Loshchilov & Hutter, 2017)				
LR	1e-4, 2e-4, 3e-4				
LR Scheduler	cosine				
Batch Size	8 (4)				
Gradient Accumulation Steps	1 (2)				
Sequence Length	512				
Warmup Steps	100				
Epochs	1				
Target Modules	Q, K, V, O, Up, Down, Gate				

Table 9: Hyperparameters for fine-tuning LLaMA2-7B on Commonsense Reasoning using PEFT.

Hyperparameters	LoRA	DoRA	MosLoRA	PaCA	RAPA
Training Precision	16-bit mixed precision (Micikevicius et al., 2017)				
Rank	128	128	128	256	910
α	64	64	64	128	455
DropOut	0.1	0.1	0.1	-	-
Optimizer	AdamW (Loshchilov & Hutter, 2017)				
LR	2e-4, 3e-4				
LR Scheduler	cosine				
Batch Size	8				
Gradient Accumulation Steps	1				
Sequence Length	512				
Warmup Steps	100				
Epochs	1				
Target Modules	Q, K, V, O, Up, Down, Gate				

Table 10: Hyperparameters for instruction tuning Mistral-7B-v0.1 on OASST1 dataset using PEFT.

Hyperparameters	LoRA	DoRA	MosLoRA	PaCA	RAPA
Training Precision	16-bit mixed precision (Micikevicius et al., 2017)				
Rank	64	64	64	128	602
α	1				
DropOut	0.1	0.1	0.1	-	-
Optimizer	AdamW (Loshchilov & Hutter, 2017)				
LR	1e-3, 5e-4				
LR Scheduler	linear				
Batch Size	4				
Gradient Accumulation Steps	4				
Sequence Length	1024				
Warmup Steps	61				
Epochs	1				
Target Modules	Q, K, V, O, Up, Down, Gate				

G LIMITATION

While RAPA demonstrates strong empirical accuracy and training efficiency across a variety of fine-tuning tasks, several limitations remain. First, our method is evaluated solely in the context of fine-tuning pretrained language models; its applicability to other adaptation paradigms, such as continual learning or multi-stage pretraining, remains an open question. Second, although RAPA improves gradient propagation via structured index alignment, its effectiveness may be sensitive to model architecture and depth, particularly in transformer variants that diverge from standard residual block structures. Finally, RAPA employs a fixed global index set shared across layers. While this promotes implementation simplicity and structural coherence, it may limit the potential for layer-specific specialization. Exploring more flexible alignment strategies that adaptively tailor index sets per layer is a promising direction for future work.

H BROADER IMPACTS

H.1 MULTI-RAPA: EFFICIENT MULTI-ADAPTER SERVING WITHOUT MERGING

Recent works such as S-LoRA (Sheng et al., 2023) has highlighted the importance of supporting multiple fine-tuned adapters simultaneously in deployment environments. These systems aim to improve inference scalability by avoiding the need to merge adapter weights back into the base model, dynamically switching between LoRA modules to support various tasks.

Our proposed method, RAPA, complements this trend by further reducing the memory and computational demands of each adapter. Unlike LoRA-based methods that apply low-rank updates across all input dimensions, RAPA fine-tunes only a small square-aligned subset of weights. This makes each adapter lighter in both parameter count and run-time footprint, which in turn enables more adapters to be hosted concurrently in constrained environments. During inference, RAPA requires fewer compute and memory resources per step, an advantage particularly relevant to multi-tenant inference servers, on-device deployment, and cost-sensitive applications.

In summary, RAPA not only improves parameter-efficient fine-tuning but also enables more scalable, sustainable, and inclusive model serving, advancing the field of adaptive language modeling.

H.2 MOE-COMPATIBLE ADAPTATION WITH RAPA

RAPA supports Mixture-of-Experts (MoE) architectures by allowing each expert to update only a small square-aligned subspace of weights, requiring only 0.1 to 0.5% additional parameters per expert. This results in significantly reduced memory and computation overhead during both training and inference. When combined with sparse activation of MoE, the active parameter count per forward pass drops further, enabling low-latency and memory-efficient adaptation. Previous work such as MixLoRA (Li et al., 2024) and Mixtral (Jiang et al., 2024a) shows that lightweight adapters can improve multitask accuracy without compromising efficiency; RAPA achieves similar gains with even smaller experts and no merging overhead. Since each expert is only a few megabytes, hundreds of domain-specific RAPA experts can be held in memory simultaneously, supporting scalable and sustainable deployment. This design reduces entry barriers for low-resource settings, facilitates localization, and mitigates interference between tasks, offering a modular and future-proof approach to continuous customization of large models.