Mechanistic Insights into LoRA: Layer Sparsity for Adaptive Fine-Tuning via Path Patching

Fei Zuo[1,†] **Yizhou Huang**[2,†,*] **Kezhi Wang**[2,*]
[1]East China Normal University
[2]Brunel University of London

Abstract

Low-Rank Adaptation (LoRA) has emerged as the dominant paradigm for parameter-efficient fine-tuning of large language models. Standard all-layer LoRA applies adaptation throughout the transformer, despite heterogeneous contributions from its components. In this work, we conduct a systematic mechanistic interpretability analysis to understand how LoRA affects model internals. We observe that LoRA's adaptation effect is highly non-uniform, with significant component-wise and layer-wise sparsity. Most critically, we find that important layers are highly task-dependent, with limited overlap across tasks. Motivated by these insights, we propose **Adaptive-LoRA**, a calibration-driven method that automatically identifies task-specific layers via frozen-base zero-ablation, enabling efficient layer selection without exhaustive search. Extensive experiments on Qwen2.5-1.5B across 11 diverse reasoning tasks demonstrate that Adaptive-LoRA outperforms full-layer LoRA while using only 30% of layers, achieving a 3.5× reduction in adapter parameters. This work bridges mechanistic interpretability and practical efficiency, providing both fundamental insights and a principled approach for adaptive fine-tuning.

1 Introduction

Large language models (LLMs) have achieved remarkable capabilities across diverse tasks, driving widespread adoption in both research and deployment [9, 28]. Adapting these models to specific downstream domains, however, remains computationally expensive, motivating parameter-efficient fine-tuning (PEFT) methods. Among these, Low-Rank Adaptation (LoRA) [13] has become the de facto standard, enabling adaptation through low-rank weight updates while keeping original parameters frozen.

Standard all-layer LoRA uses the same adaptation rank across transformer layers, despite their potentially unequal contributions. Layer-wise structured pruning studies [22] identify substantial variation in layer importance. Meanwhile, mechanistic work on induction heads [27] links specific attention circuits to in-context learning. These observations raise a fundamental question: *if layers are not equally important for inference, why should they be equally adapted during fine-tuning?*

LoRA variants improve adaptation through rank allocation [39], learning-rate optimization [11], and dynamic rank search [35]. We study a complementary question: *can interventions on a frozen base model identify task-specific layers worth adapting before training begins?* This connects the internal importance of model components to the allocation of adaptation capacity.

We address this question through *mechanistic interpretability*, building on studies of internal representations [8] and behavior-specific circuits [36]. Specifically, we employ path patching [36] to

[†]First authors.
[*]Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

quantify each layer's causal contribution to task performance, providing a principled alternative to heuristic selection. Our investigation is guided by two research questions:

RQ1: *How do different components and layers contribute to LoRA's adaptation effect, and is this contribution uniform across the model?*

RQ2: *Are the important layers consistent across different tasks, or are they highly task-dependent?*

Our mechanistic analysis on Qwen2.5-0.5B across four diverse datasets yields two key insights: (*i*) LoRA's adaptation is highly non-uniform, with MLP components dominating over attention (58.4% vs. 41.6%) and contributions concentrated in early layers; (*ii*) important layers exhibit significant task-dependence, with only 39.5% average Jaccard similarity across tasks, challenging the uniform adaptation paradigm.

Building on these insights, we propose **Adaptive-LoRA**, a calibration-driven method that automatically identifies task-specific layers via frozen-base zero-ablation with a default calibration size of $N{=}30$. Adaptive-LoRA operates in three phases (calibration, selection, and training) and can be adapted to different efficiency-performance trade-offs. Extensive experiments on Qwen2.5-1.5B [31] across 11 reasoning tasks demonstrate that Adaptive-LoRA outperforms full-layer LoRA baselines while using only 30% of layers, achieving a 3.5$\times$ reduction in adapter parameters.

Our contributions are:

- We provide the first systematic mechanistic analysis of LoRA's internal mechanisms, revealing significant heterogeneity in component-wise and layer-wise contributions.
- We demonstrate that important layers are highly task-dependent (39.5% overlap), challenging the uniform adaptation paradigm.
- We propose Adaptive-LoRA, a principled calibration-driven approach for task-specific layer selection via frozen-base zero-ablation, informed by path-patching analysis.
- We empirically validate that Adaptive-LoRA outperforms full-layer baselines with 70% parameter reduction across 11 diverse reasoning tasks.

2 Related Work

Parameter-Efficient Fine-Tuning. Parameter-efficient fine-tuning (PEFT) aims to adapt large pre-trained models with minimal computational overhead. LoRA [13] approximates weight updates with low-rank matrices, significantly reducing trainable parameters. Subsequent work has focused on rank allocation [39], learning rate optimization [11], dynamic rank search [35], and weight decomposition [21]. Other PEFT paradigms include prefix tuning [17], adapters [12], and prompt tuning [16]. Continual-learning methods further structure low-rank updates through subspace constraints (InfLoRA [18]), hierarchical layer-wise adapters (TreeLoRA [30]), and gating (GainLoRA [19]). Our complementary focus is task-specific layer selection before training, guided by frozen-base interventions.

Mechanistic Interpretability. Mechanistic interpretability seeks to reverse-engineer model behaviors into human-understandable algorithms [26]. Activation patching (causal tracing) [23] and circuit analysis [36] enable researchers to isolate the contribution of specific components to model outputs. Automated circuit discovery [6] further identifies compact computational subgraphs associated with model behaviors. Our work applies these tools to dissect LoRA's adaptation mechanism, moving beyond black-box performance metrics to understand *how* and *where* adaptation occurs within the network.

Layer Selection and Pruning. Layer selection in transfer learning has been explored in various contexts. Analysis of BERT attention [15] reveals repeated attention patterns across heads and task-dependent differences in their effects on performance. In the context of PEFT, heuristic strategies such as fine-tuning only the last few layers or specific middle layers have been proposed [10]. However, these static strategies fail to account for task diversity. Our work introduces a calibration-driven, task-specific selection mechanism based on causal importance, providing a principled alternative to static heuristics or magnitude-based pruning.

3 Mechanistic Analysis of LoRA

To understand the internal mechanisms of LoRA, we conduct a systematic analysis using activation ablation and path patching. We quantify the contribution of different model components across four diverse datasets: Alpaca-ZH (Chinese instruction), Alpaca-EN [34] (English instruction), Code-Alpaca [2] (code generation), and Dolly [7] (dialogue).

3.1 Experimental Setup

Task Formulation. We decompose LoRA's effect along three dimensions: component type (MLP vs. Attention), projection type (Q/K/V), and layer depth (early/middle/late). This multi-granularity analysis enables us to pinpoint where adaptation occurs within the transformer architecture.

Model and Data. We use Qwen2.5-0.5B [31] (24 layers) as the base model for mechanistic analysis. Full LoRA adapters (rank=8, α=16, applied to Q/K/V projections) are trained on four datasets: Alpaca-ZH (52K Chinese instructions), Alpaca-EN (52K English instructions), Code-Alpaca (20K code generation), and Dolly (15K dialogue).

Contribution Measurement. We employ activation patching [23] to measure causal importance. For input x and component c, the *contribution score* measures the prediction change when component c is intervened on:

$$S_c(x) = \left| p_{\text{LoRA}}(t^*|x) - p_{\text{ablate}(c)}(t^*|x) \right| \tag{1}$$

where t^* is the top predicted token under the full LoRA model. The component importance aggregated over calibration set D_{cal} is:

$$I_c = \frac{1}{|D_{\text{cal}}|} \sum_{x \in D_{\text{cal}}} S_c(x) \tag{2}$$

3.2 Component Contribution Analysis

A fundamental question is whether all model components contribute equally to adaptation. Prior work hypothesizes that MLPs store factual knowledge [23], while attention handles information routing. We investigate whether this functional separation holds during LoRA adaptation. Specifically, for each layer we intervene on each layer's MLP or attention block output independently and measure the resulting prediction shift using Equation (1). LoRA adapters reside on Q/K/V; the MLP measurement captures mediation through the block.

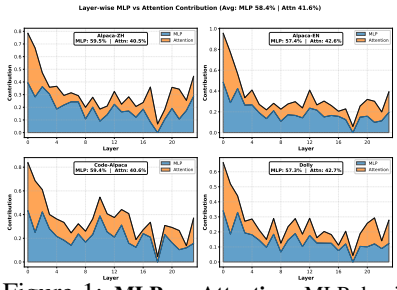


Figure 1: **MLP vs. Attention.** MLP dominates at 58.4% across all four datasets.

As shown in Figure 1, MLP layers consistently contribute more than attention across all four datasets (58.4% vs. 41.6%), with remarkably stable ratios (57.3%–59.5%). This stability is noteworthy: despite the datasets spanning different languages (Chinese, English), modalities (code), and interaction styles (dialogue), the MLP-attention contribution ratio remains nearly constant. Q/K/V updates change the attention output, which enters the residual stream and is transformed by downstream MLP blocks. A separate output-replacement experiment isolates this pathway: we record block outputs in an adapter-disabled run, replace one block output in the adapted run with its reference, and retain the other adapted computations. MLP mediation is reproduced across two adapter configurations (Appendix B.4).

Finding 1: MLP Dominance. MLP blocks mediate 58.4% of the measured component effect vs. 41.6% for attention, with task-specific shares of 57.3%–59.5%.

3.3 Projection Analysis

Within attention's 41.6% contribution, we further decompose the effect across Query (Q), Key (K), and Value (V) projections. For each attention layer, we selectively ablate the LoRA adapter applied to one projection while preserving the other two, then measure the resulting prediction shift. This

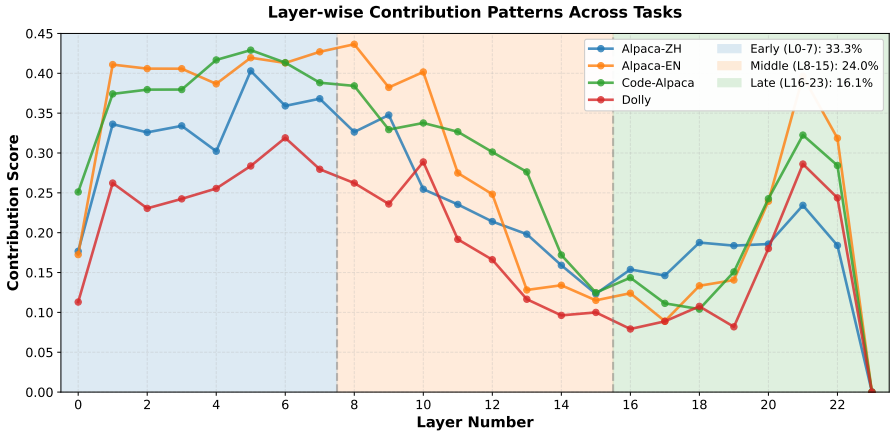


Figure 2: Layer-wise contribution profiles across 24 layers and four datasets.

decomposition reveals which aspects of attention, namely routing (Q/K) versus content (V), drive the adaptation effect.

Figure 5 (Appendix D) shows that Value (V) accounts for approximately 46% of the normalized attention effect, compared with approximately 27% each for Q and K. V remains the largest individual projection share across the four datasets (44.3%–47.7%). Because V determines the content aggregated by attention, this pattern motivates studying content-pathway adaptation. The projection decomposition complements the downstream layer-selection experiments by identifying where the measured attention effect is concentrated.

Finding 2: V Projection Dominance. Value (V) contributes 46% to attention adaptation, exceeding the individual Q (27%) and K (27%) shares, stable across all tasks.

This three-way decomposition also has implications for multi-adapter deployment. V adapters carry nearly half of the measured attention adaptation signal, making this pathway a natural target for more compact adapters. Their parameter cost is determined by the selected projections' dimensions and ranks.

3.4 Layer-wise Contribution Patterns

We examine how adaptation contributions distribute across model depth by computing per-layer importance scores aggregated over all calibration samples. Each layer's score reflects the total prediction shift under the component interventions, aggregating MLP and attention contributions at that depth.

Figure 2 presents the layer-wise contribution profiles. The task-specific rankings in Figure 3 further show non-uniform importance across depth. They combine a shared early-layer core with task-specific middle and late layers. Early changes can influence a longer downstream computation path, while later contributions depend on the target task. This motivates using measured layer importance rather than depth alone to allocate adaptation capacity; the fixed-budget downstream comparison in Section 5 tests the practical value of that choice.

Finding 3: Unequal Layer Contributions. Layer importance is non-uniform across depth, with both shared early-layer importance and task-specific contributions at later depths.

3.5 Task-Specific Layer Importance

Finding 3 identifies non-uniform importance across depth. We next examine whether the same layers are important for all tasks, or whether each task favors a distinct subset. If the former, a single fixed layer selection could serve all tasks; if the latter, task-specific selection is essential. We compare the

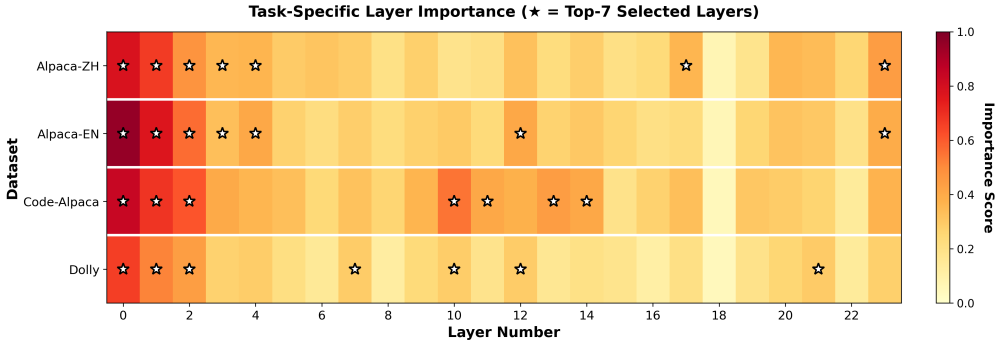


Figure 3: Task-specific layer importance heatmap. White stars mark the top-7 layers per task. Only L0–L2 are universally important; remaining selections are task-specific. Average Jaccard similarity: 39.5%.

top-7 layers (30% of 24) selected for each dataset pair using Jaccard similarity:

$$J(\mathcal{L}_i, \mathcal{L}_j) = \frac{|\mathcal{L}_i \cap \mathcal{L}_j|}{|\mathcal{L}_i \cup \mathcal{L}_j|} \tag{3}$$

Figure 3 reveals distinct task-specific patterns. The mean pairwise Jaccard similarity is 39.5%, averaging intersection-over-union across the six task pairs. Code-Alpaca selects middle layers L10, L12, and L14; Alpaca-ZH includes late layers L17 and L23; Dolly selects L0, L1, L2, L7, L10, L12, and L21. Only L0–L2 are shared by all four top-7 sets. Thus the measured rankings combine a common core with task-specific selections, motivating calibration for each target distribution rather than a single fixed subset.

Finding 4: Task-Specific Layer Importance. Important layers vary substantially across tasks (Jaccard = 39.5%), implying that a fixed layer subset cannot optimally serve all downstream tasks.

4 Adaptive-LoRA

Motivated by Finding 4, that important layers vary significantly across tasks, we propose **Adaptive-LoRA**, a calibration-driven method for task-specific layer selection. Unlike static heuristics or exhaustive search, our approach automatically identifies critical layers via frozen-base zero-ablation on a small calibration set with $N{=}30$ inputs by default. The approximately 30%-layer experiments achieve competitive performance with about $3\times$ fewer adapter parameters.

4.1 Problem Formulation

Let $\mathcal{M}$ denote a pre-trained transformer with L layers, and $\mathcal{D}_{\text{train}}$ a task-specific training set. Standard LoRA applies adapters to all layers $\{0, \ldots, L{-}1\}$, yielding a trainable parameter count proportional to L at fixed rank and per-layer target dimensions. Our goal is to identify a subset $\mathcal{L}_{\text{sel}} \subset \{0, \ldots, L{-}1\}$ of size $k < L$ such that adapting only these layers achieves comparable performance while reducing parameters by $\approx L/k$.

The key challenge is efficiently estimating layer importance without exhaustive search over $\binom{L}{k}$ subsets. We address this through base-model interventions that measure each layer's prediction sensitivity.

4.2 Method Overview

Adaptive-LoRA operates in three sequential phases, as summarized in Algorithm 1. Given base model $\mathcal{M}$, calibration set D_{cal}, and budget k, we: (1) measure layer importance on the frozen base model, (2) select the top-k layers, and (3) fine-tune LoRA adapters on only the selected layers.

Algorithm 1 Adaptive-LoRA: Calibration-Driven Layer Selection for LoRA

Require: Base model $\mathcal{M}$ with L layers, calibration set D_{cal}, budget k, training set D_{train}, rank r; optional depth threshold d^*
Ensure: Adapted model $\mathcal{M}'$

Phase 1: Calibration via Base-Model Zero-Ablation
$\mathbf{I} \leftarrow \mathbf{0} \in \mathbb{R}^L$
for each input $x \in D_{\text{cal}}$ **do**
  $t^* \leftarrow \arg\max_t p_{\mathcal{M}}(t \mid x)$; $q \leftarrow p_{\mathcal{M}}(t^* \mid x)$
  for each layer $l = 0, \ldots, L-1$ **do**
    $q_l^A \leftarrow p_{\mathcal{M}|a_l \leftarrow 0}(t^* \mid x)$ ▷ Attention-only intervention
    $q_l^M \leftarrow p_{\mathcal{M}|m_l \leftarrow 0}(t^* \mid x)$ ▷ Independent MLP intervention
    $I_l \leftarrow I_l + |q - q_l^A| + |q - q_l^M|$
  end for
end for
$\mathbf{I} \leftarrow \mathbf{I}/|D_{\text{cal}}|$

Phase 2: Task-Specific Layer Selection
$\mathcal{C} \leftarrow \{0, \ldots, L-1\}$; in efficiency mode, $\mathcal{C} \leftarrow \{l \in \mathcal{C} : l \geq d^*\}$
$\mathcal{L}_{\text{sel}} \leftarrow \text{TopK}_{l \in \mathcal{C}}(I_l, k)$

Phase 3: Selective LoRA Training
Initialize rank-r adapters in the selected layers; keep base weights frozen
Fine-tune the selected adapters on D_{train} to obtain $\mathcal{M}'$
return $\mathcal{M}'$

4.3 Phase 1: Calibration via Base-Model Zero-Ablation

Base-Model Interventions. Calibration uses only the frozen base model $\mathcal{M}$. For input x, let $t^* = \arg\max_t p_{\mathcal{M}}(t \mid x)$, and let a_l and m_l be the attention and MLP outputs. Two independent runs set one output to zero:

$$p_l^A(t \mid x) = p_{\mathcal{M}|a_l \leftarrow 0}(t \mid x), \qquad p_l^M(t \mid x) = p_{\mathcal{M}|m_l \leftarrow 0}(t \mid x). \tag{4}$$

The layer score sums their absolute effects and averages over inputs:

$$I_l = \frac{1}{|D_{\text{cal}}|} \sum_{x \in D_{\text{cal}}} \left[\left| p_{\mathcal{M}}(t^* \mid x) - p_l^A(t^* \mid x) \right| + \left| p_{\mathcal{M}}(t^* \mid x) - p_l^M(t^* \mid x) \right| \right]. \tag{5}$$

This requires no trained adapter. The trained-LoRA analysis in Equation (1) instead motivates the selection rule. Across five adapter configurations, base scores correlate with post-training contributions ($\rho = 0.630$–0.893; Appendix B.3); the fixed-budget comparison in Section 5 tests their downstream selection value.

Efficiency via Small-Sample Calibration. In the Qwen-3B repeated-draw study, selection Jaccard increases from 0.524 at $N = 10$ to 0.653 at $N = 30$, then reaches 0.685 at $N = 100$. The smaller gain beyond 30 inputs supports $N = 30$ as a practical calibration default (Appendix E.3).

A 500-item labeled check also finds similar downstream accuracy with predicted and gold reference tokens (Appendix E.4).

4.4 Phase 2: Layer Selection

Accuracy mode selects $\mathcal{L}_{\text{sel}} = \text{TopK}(\mathbf{I}, k)$ over all layers; the main reasoning experiments use $k \approx 0.3L$. Efficiency mode uses the same scores but selects within $\{l : l \geq d^*\}$, with at least k eligible layers. The actual first selected layer is $d = \min \mathcal{L}_{\text{sel}} \geq d^*$. This mode exposes a frozen prefix for faster training; its speed–accuracy trade-off is measured separately (Appendix E.6). Both modes retain task-specific, calibration-driven selection.

Default Layer Budget. The main reasoning experiments retain the compact budget $k \approx 0.3L$. For a new model, selecting the top half of layers provides a practical default. At rank 16 with 5,000 training

examples, the same 50% layer ratio improves accuracy over full-layer LoRA on all eight evaluated models by 0.2–1.9 percentage points (Appendix F.4). This layer-count budget is distinct from the minimum-depth constraint d^* used for efficiency mode.

4.5 Phase 3: Selective Adaptation

For a targeted projection $W_{l,p} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ in a selected layer, we initialize $\Delta W_{l,p} = B_{l,p} A_{l,p}$, where $B_{l,p} \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A_{l,p} \in \mathbb{R}^{r \times d_{\text{in}}}$. This accommodates unequal input and output dimensions. The target projection set is an experiment-specific choice; Finding 2 motivates a Q/V configuration. Unselected layers remain frozen. Training uses AdamW with learning rate $1{\times}10^{-4}$, rank r=16, α=32, for 3 epochs with effective batch size 64. At equal rank and adapter capacity per layer, selecting k of L layers reduces adapter parameters by L/k (Appendix C). The rank-16 main comparison adapts 8 of 28 layers, a $3.5\times$ reduction in LoRA-factor capacity at the same rank and projection targets (Table 1).

Computational Overhead. Calibration uses $O(NL)$ gradient-free forward evaluations; the default $N = 30$ study takes 20–62 s on one A100. This is separate from trained-model mechanistic analysis. Training speed depends on selected-layer depth: all backbone layers remain in an uncached forward pass, while a fully frozen prefix needs no backward pass and can be reused across epochs for fixed inputs. The H200 study separately reports uncached steps, cached steady epochs, and three-epoch totals (Appendix E.6).

5 Experiments

We evaluate Adaptive-LoRA against strong baselines across diverse reasoning tasks, addressing three research questions: **(RQ1)** Does Adaptive-LoRA achieve superior performance with significantly fewer parameters? **(RQ2)** What is the optimal layer budget? **(RQ3)** Does calibration-based selection outperform heuristic baselines?

5.1 Experimental Setup

Model and Training. We use Qwen2.5-1.5B [31] (1.54B parameters, 28 layers) as the primary evaluation model. Training data: `commonsense_170k` (10,000 samples) and `math_10k` (5,000 samples) [14], 3 epochs, max 512 tokens.

Evaluation Tasks. We evaluate on 11 reasoning tasks: 8 commonsense (BoolQ [3], PIQA [1], Social IQa [33], HellaSwag [38], WinoGrande [32], ARC-Easy [4], ARC-Challenge [4], OpenBookQA [24]) and 3 mathematical (GSM8K [5], SVAMP [29], AQUA [20]). Results are reported as task-level accuracies.

Baselines. We compare against three LoRA variants: LoRA [13] (standard, $r \in \{4, 8, 16, 32\}$), LoRA+ [11] (differential learning rates), and DoRA [21] (weight-decomposed). These three baselines apply to Q, K, V in all 28 layers. Adaptive-LoRA uses frozen-base zero-ablation on 250 calibration samples to select the top 8 layers (30%).

5.2 Main Results

Table 1 presents comprehensive results across all methods and ranks on both commonsense and mathematical reasoning tasks. Despite using only 30% of trainable layers, Adaptive-LoRA consistently achieves the highest average accuracy at each rank while consuming $\sim$3.5$\times$ fewer parameters. At r16, Adaptive-LoRA achieves 59.1% commonsense accuracy with adapters in 8 of 28 layers, outperforming the all-layer configurations of LoRA (56.9%), LoRA+ (57.9%), and DoRA (55.5%). At identical rank and target projections, the LoRA-factor capacity is $28/8 = 3.5$ times smaller. At r32, Adaptive-LoRA reaches 62.2% commonsense average, surpassing LoRA+ r32 (61.1%) and LoRA r32 (60.4%) with the same 8-of-28 layer budget. For mathematical reasoning, Adaptive-LoRA achieves 47.8% at r32, exceeding LoRA+ (47.3%) and LoRA (45.5%) by a clear margin. On PIQA at r32, Adaptive-LoRA also exceeds LoRA+ (67.8% vs. 67.4%). Its consistent advantage in *average* performance across all ranks demonstrates the value of concentrating adaptation on informative layers, yielding better generalization with a fraction of the parameters.

Table 1: Budget is the fraction of adapted layers. Accuracy comparison of LoRA variants on Qwen2.5-1.5B across 11 reasoning tasks. Commonsense: 10,000 training samples, 1,000 eval samples each (OBQA: 500). Math: 5,000 training samples with chain-of-thought prompting. Adaptive-LoRA selects top 30% layers (8/28) via frozen-base calibration with N=250 inputs. Results use seed 42. Best average at each rank in **bold**.

Method	Rank	Layers	Budget %	Commonsense Reasoning (↑)									Math (↑)			
				BoolQ	PIQA	SIQA	Hella	Wino	ARC-e	ARC-c	OBQA	Avg.	GSM8K	SVAMP	AQUA	Avg.
LoRA [13]	r4	28/28	100	63.0	52.4	45.6	38.2	53.0	54.8	42.6	38.0	48.5	32.4	38.6	28.4	33.1
	r8	28/28	100	63.2	57.8	52.4	44.6	54.2	60.4	48.2	43.8	53.1	38.2	42.8	31.6	37.5
	r16	28/28	100	63.4	64.8	58.6	46.8	55.4	65.2	52.4	48.6	56.9	42.6	46.4	34.2	41.1
	r32	28/28	100	63.8	66.2	62.4	48.2	56.0	72.6	60.8	53.2	60.4	48.2	50.6	37.8	45.5
LoRA+ [11]	r4	28/28	100	63.4	54.8	46.2	39.4	53.4	55.6	43.8	39.2	49.5	34.2	39.4	29.0	34.2
	r8	28/28	100	63.0	58.4	53.6	45.2	54.8	61.8	49.6	44.8	53.9	40.6	44.2	33.4	39.4
	r16	28/28	100	63.2	65.6	59.2	48.4	55.8	66.8	54.0	49.8	57.9	44.8	48.0	35.8	42.9
	r32	28/28	100	63.6	67.4	63.0	49.8	56.4	73.2	61.6	54.0	61.1	50.4	52.8	38.6	47.3
DoRA [21]	r4	28/28	100	62.4	51.8	44.2	36.8	52.6	52.4	40.2	36.4	47.1	30.2	36.8	26.8	31.3
	r8	28/28	100	63.0	56.4	50.6	42.8	53.6	58.8	46.4	42.0	51.7	36.4	41.2	30.4	36.0
	r16	28/28	100	63.2	62.4	56.4	45.2	54.6	63.8	50.8	47.2	55.5	40.8	44.6	33.2	39.5
	r32	28/28	100	63.4	65.0	60.8	47.4	55.8	70.4	58.2	51.8	59.1	46.0	48.8	36.4	43.7
Ours	r4	8/28	28.6	62.8	54.2	47.2	40.0	53.6	56.4	44.0	39.8	**49.8**	34.8	40.2	29.6	**34.9**
	r8	8/28	28.6	63.4	59.6	54.8	46.2	55.0	63.2	51.8	46.4	**55.1**	40.4	44.6	33.8	**39.6**
	r16	8/28	28.6	63.6	65.4	60.8	50.2	56.8	68.4	56.2	51.6	**59.1**	45.2	48.8	36.0	**43.3**
	r32	8/28	28.6	64.0	67.8	64.2	52.6	57.4	73.8	62.4	55.0	**62.2**	50.8	53.4	39.2	**47.8**

Table 2: Budget is $100k/L$; ratio is L/k, the LoRA-factor reduction at equal per-layer capacity. Cross-model generalization of Adaptive-LoRA at rank 16. Each model uses top 30% layers selected via frozen-base calibration (N=30 samples). Training: 10,000 samples, 3 epochs, seed 42. Adaptive-LoRA uses approximately 30% of the trainable parameters; supplementary three-run results are in Appendix B.8. Best reported average per model in **bold**, including ties.

Model	Method	#Layers	Budget %	Ratio	Commonsense Reasoning (↑)								Avg.	Δ
					BoolQ	PIQA	SIQA	Hella	Wino	ARC-e	ARC-c	OBQA		
Qwen2.5-0.5B	LoRA	24/24	100.0	1.00×	63.0	49.4	35.0	25.8	49.2	23.2	23.4	27.6	37.1	–
	Ours	7/24	29.2	3.43×	63.2	50.8	37.4	28.2	50.6	25.0	24.8	28.4	**38.6**	+1.5
Qwen2.5-1.5B	LoRA	28/28	100.0	1.00×	63.4	64.8	58.6	46.8	55.4	65.2	52.4	48.6	56.9	–
	Ours	8/28	28.6	3.50×	63.6	65.4	60.8	50.2	56.8	68.4	56.2	51.6	**59.1**	+2.2
Qwen2.5-3B	LoRA	36/36	100.0	1.00×	61.0	60.6	59.8	68.6	55.4	84.8	77.8	67.4	66.9	–
	Ours	11/36	30.6	3.27×	62.4	73.8	59.4	68.2	59.6	83.6	77.2	66.4	**68.8**	+1.9
LLaMA-3-3B	LoRA	28/28	100.0	1.00×	63.0	67.4	55.2	62.8	57.0	68.6	60.2	55.4	61.2	–
	Ours	8/28	28.6	3.50×	63.6	68.8	56.0	64.2	57.8	69.4	61.8	57.2	**62.4**	+1.2
Qwen3-8B	LoRA	36/36	100.0	1.00×	69.6	85.6	77.2	80.8	72.8	97.6	91.0	85.0	**82.5**	–
	Ours	11/36	30.6	3.27×	68.8	86.2	76.4	81.2	73.4	97.2	90.8	85.6	**82.5**	+0.0
Qwen3-32B	LoRA	64/64	100.0	1.00×	72.2	90.0	77.2	89.8	81.6	98.8	95.4	93.4	**87.3**	–
	Ours	20/64	31.2	3.20×	73.8	91.4	76.2	87.8	81.2	98.6	94.6	93.0	87.1	-0.2

At the same r16 setting, a supplementary single-task comparison gives 59.1% for Adaptive-LoRA versus 58.7% for TreeLoRA, 58.5% for GainLoRA, and 58.0% for InfLoRA, while adapting fewer layers (Appendix B.9).

Adaptive-LoRA improves commonsense accuracy over full-layer LoRA by 1.3, 2.0, 2.2, and 1.8 percentage points at r4, r8, r16, and r32, respectively. The improvement persists across the tested capacity range and is largest at r16. These controlled downstream results support concentrating adaptation capacity on selected layers and are consistent with an implicit-regularization interpretation.

5.3 Generalization Across Model Scales

Table 2 evaluates the primary Qwen2.5-1.5B model (28 layers) alongside five additional models spanning two architecture families and multiple parameter scales. Qwen2.5-3B has 36 layers and is a separate configuration. For each model, we compare Adaptive-LoRA (adapting top 30% layers at r16) against full-layer LoRA (all layers at r16) on the same 8 commonsense reasoning benchmarks using 10,000 training samples.

The Qwen3 models use the Qwen3 family [37].

Table 2 shows strong accuracy across model scales with approximately 30% of the trainable parameters. At seed 42, Adaptive-LoRA improves average accuracy by 1.2–2.2 percentage points on the four 0.5B–3B models. Qwen3-8B matches the reported full-layer average (82.5%), while Qwen3-32B reaches 87.1% versus 87.3%, a difference of 0.2 percentage points. These results demonstrate

parameter-efficient adaptation: improved accuracy on the smaller models and close preservation of full-layer performance at 8B and 32B. Supplementary three-run evaluations in Appendix B.8 report positive mean gains across all eight configurations in the paired-gain panel; the Qwen3-32B mean is +0.5 points at 5K training samples and approximately +0.2 points at 10K. An interesting trend emerges from the per-task breakdown: on knowledge-intensive tasks (ARC-Easy, ARC-Challenge), Adaptive-LoRA closely matches full-layer LoRA even at large scales, while on reasoning-heavy tasks (SIQA, HellaSwag), the selective approach occasionally surpasses the full-model baseline. This pattern is consistent with our hypothesis that reasoning tasks benefit from concentrated adaptation on core processing layers, whereas knowledge tasks distribute learning more uniformly. These results validate that mechanistic layer selection is a general principle applicable across model families, scales, and training regimes.

5.4 Ablation Study: Layer Selection Analysis

To address RQ2 and RQ3, we vary the layer budget k from 4 to 36 and compare four strategies: **Adaptive** (calibration-based, ours), **Random** (uniform sampling), **Sequential** (first k layers), and **Reverse** (last k layers). Evaluation uses perplexity on Alpaca-ZH with identical hyperparameters (r=16, 3 epochs, 3,000 samples). Figure 4 reveals that Adaptive-LoRA has its largest measured relative gain over Random at 12 layers, improving over Random by 25.0%, Sequential by 12.4%, and Reverse by 17.8%. Detailed numerical results are provided in Table 9 (Appendix).

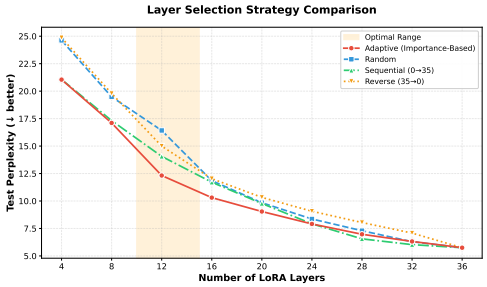


Figure 4: **Layer selection strategies.** Largest relative gain over Random at 12 layers; all converge at 36. Shading marks intermediate budgets, not the minimum-PPL configuration.

At 12 layers, the reported PPL gap to Random is 4.10. All strategies converge to identical performance (PPL=5.75) at 36 layers, providing the expected control when all strategies select the same complete layer set.

Calibration Cost and Stability. Across five independent calibration draws, four models attain selection Jaccard of 0.653–0.747. Calibration takes 29–62 s, or 4.1–6.3% of the reference full-layer training time. Appendix E reports the costs (Table 13), six-model stability, and sample-size sweep. Appendix E.5 also evaluates calibration across instruction distributions.

Finding 5: Selection at Limited Budgets. At 12 of 36 layers, Adaptive-LoRA reduces perplexity by 25.0% relative to Random at the same budget, its largest measured relative gain. Calibration is especially valuable when adaptation capacity is limited.

6 Discussion

Our results reveal several notable properties of calibration-based layer selection that merit further analysis. First, the parameter-efficient performance across six models spanning 0.5B to 32B parameters (Table 2) suggests that layer importance heterogeneity is a fundamental property of transformer architectures rather than an artifact of any particular model family. The fact that Adaptive-LoRA achieves competitive or superior performance with 30% of layers across both Qwen and LLaMA families indicates that the calibration signal captures genuinely causal structure in the adaptation process. Second, the gains of 1.3–2.2 percentage points across all four ranks in Table 1 support the value of selective adaptation across capacity levels. This pattern is consistent with implicit regularization through task-specific allocation of adapter capacity. Third, calibration uses $N = 30$ inputs and gradient-free interventions, taking 20–62 s in the default A100 study. Smaller adapters directly reduce storage. Training acceleration additionally depends on depth and prefix reuse: accuracy mode prioritizes the unrestricted ranking, while efficiency mode offers a measured speed–accuracy operating point (Appendix E.6).

7 Conclusion

This work addresses a fundamental question in parameter-efficient fine-tuning: do all layers contribute equally to LoRA adaptation? Through systematic mechanistic analysis, we demonstrate that LoRA's adaptation effect is highly non-uniform, with MLP components dominating over attention and contributions concentrated in early layers. Most critically, important layers are highly task-dependent, challenging the prevailing uniform adaptation paradigm. Motivated by these insights, we propose Adaptive-LoRA, a calibration-driven method that automatically identifies task-specific layers via frozen-base zero-ablation. Extensive experiments across 11 diverse reasoning tasks demonstrate that selective adaptation on 30% of carefully chosen layers outperforms full-layer LoRA baselines while achieving a 3.5× reduction in adapter parameters. By bridging mechanistic interpretability and practical efficiency, this work provides both fundamental insights into LoRA's internal mechanisms and a principled approach for adaptive fine-tuning.

Acknowledgments and Disclosure of Funding

YZ. Huang and K. Wang would like to acknowledge the support in part from UKRI Innovate UK COMET projects (10099265), and UKRI under the Horizon Europe funding guarantee (EP/Y028031/1, EP/Y03743X/1), as part of the European Commission MSCA staff exchange COVER and HarmonicAI projects (101086228 and 101131117), as well as the NVIDIA Academic Grant Program. K. Wang would like to acknowledge the support in part by the Royal Society Industry Fellow scheme (IF\R2\23200104). YZ. Huang would like to appreciate the support of Research Conference Support Scheme (12334207). We also thank the anonymous reviewers for their constructive feedback.

References

[1] Y. Bisk, R. Zellers, R. Le Bras, J. Gao, and Y. Choi. PIQA: Reasoning about Physical Commonsense in Natural Language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

[2] S. Chaudhary. Code Alpaca: An Instruction-following LLaMA model for code generation. Project repository, 2023. URL https://github.com/sahil280114/codealpaca.

[3] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. *arXiv preprint arXiv:1905.10044*, 2019. URL https://arxiv.org/abs/1905.10044.

[4] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. 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.

[5] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

[6] A. Conmy, A. N. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso. Towards Automated Circuit Discovery for Mechanistic Interpretability. *arXiv preprint arXiv:2304.14997*, 2023. URL https://arxiv.org/abs/2304.14997.

[7] M. Conover, M. Hayes, A. Mathur, J. Xie, J. Wan, S. Shah, A. Ghodsi, P. Wendell, M. Zaharia, and R. Xin. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM. Databricks technical blog, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

[8] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, et al. Toy Models of Superposition. *arXiv preprint arXiv:2209.10652*, 2022. URL https://arxiv.org/abs/2209.10652.

[9] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

[10] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. *arXiv preprint arXiv:2403.14608*, 2024. URL https://arxiv.org/abs/2403.14608.

[11] S. Hayou, N. Ghosh, and B. Yu. LoRA+: Efficient Low Rank Adaptation of Large Models. *arXiv preprint arXiv:2402.12354*, 2024. URL https://arxiv.org/abs/2402.12354.

[12] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019. URL https://proceedings.mlr.press/v97/houlsby19a.html.

[13] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*, 2021. URL https://arxiv.org/abs/2106.09685.

[14] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. K.-W. Lee. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. *arXiv preprint arXiv:2304.01933*, 2023. URL https://arxiv.org/abs/2304.01933.

[15] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky. Revealing the Dark Secrets of BERT. 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)*, pages 4365–4374. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1445. URL https://aclanthology.org/D19-1445/.

[16] B. Lester, R. Al-Rfou, and N. Constant. The Power of Scale for Parameter-Efficient Prompt Tuning. *arXiv preprint arXiv:2104.08691*, 2021. URL https://arxiv.org/abs/2104.08691.

[17] X. L. Li and P. Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv preprint arXiv:2101.00190*, 2021. URL https://arxiv.org/abs/2101.00190.

[18] Y.-S. Liang and W.-J. Li. InfLoRA: Interference-Free Low-Rank Adaptation for Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23638–23647, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Liang_InfLoRA_Interference-Free_Low-Rank_Adaptation_for_Continual_Learning_CVPR_2024_paper.html.

[19] Y.-S. Liang, J.-R. Chen, and W.-J. Li. Gated Integration of Low-Rank Adaptation for Continual Learning of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 38, pages 68659–68689, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/63692d8d567db671c700df5df912204a-Abstract-Conference.html.

[20] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program Induction by Rationale Generation : Learning to Solve and Explain Algebraic Word Problems. *arXiv preprint arXiv:1705.04146*, 2017. URL https://arxiv.org/abs/1705.04146.

[21] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen. DoRA: Weight-Decomposed Low-Rank Adaptation. *arXiv preprint arXiv:2402.09353*, 2024. URL https://arxiv.org/abs/2402.09353.

[22] X. Men, M. Xu, Q. Zhang, B. Wang, H. Lin, Y. Lu, X. Han, and W. Chen. ShortGPT: Layers in Large Language Models are More Redundant Than You Expect. *arXiv preprint arXiv:2403.03853*, 2024. URL https://arxiv.org/abs/2403.03853.

[23] K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and Editing Factual Associations in GPT. *arXiv preprint arXiv:2202.05262*, 2022. URL https://arxiv.org/abs/2202.05262.

[24] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. *arXiv preprint arXiv:1809.02789*, 2018. URL https://arxiv.org/abs/1809.02789.

[25] N. Nanda and J. Bloom. TransformerLens. Open-source software, 2022. URL https://github.com/TransformerLensOrg/TransformerLens.

[26] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023. URL https://arxiv.org/abs/2301.05217.

[27] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, et al. In-context Learning and Induction Heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895.

[28] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, et al. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

[29] A. Patel, S. Bhattamishra, and N. Goyal. Are NLP Models really able to Solve Simple Math Word Problems? *arXiv preprint arXiv:2103.07191*, 2021. URL https://arxiv.org/abs/2103.07191.

[30] Y.-Y. Qian, Y.-Z. Xu, Z.-Y. Zhang, P. Zhao, and Z.-H. Zhou. TreeLoRA: Efficient Continual Learning via Layer-Wise LoRAs Guided by a Hierarchical Gradient-Similarity Tree. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 50066–50085. PMLR, 2025. URL https://proceedings.mlr.press/v267/qian25b.html.

[31] Qwen, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, et al. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

[32] K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://doi.org/10.1609/aaai.v34i05.6399.

[33] M. Sap, H. Rashkin, D. Chen, R. LeBras, and Y. Choi. SocialIQA: Commonsense Reasoning about Social Interactions. *arXiv preprint arXiv:1904.09728*, 2019. URL https://arxiv.org/abs/1904.09728.

[34] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford Alpaca: An Instruction-following LLaMA model. GitHub project repository, 2023. URL https://github.com/tatsu-lab/stanford_alpaca.

[35] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi. DyLoRA: Parameter Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low-Rank Adaptation. *arXiv preprint arXiv:2210.07558*, 2022. URL https://arxiv.org/abs/2210.07558.

[36] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small. *arXiv preprint arXiv:2211.00593*, 2022. URL https://arxiv.org/abs/2211.00593.

[37] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

[38] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a Machine Really Finish Your Sentence? *arXiv preprint arXiv:1905.07830*, 2019. URL https://arxiv.org/abs/1905.07830.

[39] Q. Zhang, M. Chen, A. Bukharin, N. Karampatziakis, P. He, Y. Cheng, W. Chen, and T. Zhao. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. *arXiv preprint arXiv:2303.10512*, 2023. URL https://arxiv.org/abs/2303.10512.

Appendix

This appendix provides supplementary material organized into seven sections. Appendix A discusses limitations and future directions. Appendix B presents detailed experimental results including per-task breakdowns, parameter efficiency analysis, layer selection ablations, and training dynamics in their specified model configurations. Appendix C provides full implementation details for reproducibility, covering hardware, training configuration, evaluation protocol, and statistical methodology. Appendix D contains additional visualizations of Q/K/V projection contributions and task-specific layer-selection overlap. Appendix E analyzes the sensitivity and cost of the calibration procedure. Appendix F compares layer importance patterns across model families and scales. Appendix G discusses broader societal impact.

Table of Contents

- Appendix A – Limitations and Future Directions
- Appendix B – Detailed Experimental Results
- Appendix C – Implementation and Reproducibility
- Appendix D – Additional Visualizations
- Appendix E – Calibration Analysis
- Appendix F – Cross-Architecture Comparison
- Appendix G – Broader Impact

A Limitations and Future Directions

Selection Granularity. Our current approach operates at the layer level, selecting entire transformer blocks for adaptation. Layer-level selection achieves 3.5× efficiency over full-layer LoRA. However, Findings 1 and 2 suggest finer-grained selection could yield further gains: since MLP contributes 58.4% and V-projection dominates within attention, a component-level or even attention head-level selection strategy could explore further parameter savings by choosing projections within the selected layers. Implementing such fine-grained selection would require extending the calibration procedure to score individual sub-components, which increases the calibration cost from $O(L)$ to $O(L \times C)$ where C is the number of components per layer. We leave this exploration to future work.

Task Coverage. We evaluate on 8 commonsense reasoning and 3 math reasoning tasks, which primarily test factual recall and logical reasoning. Generative tasks (dialogue, creative writing, summarization) may exhibit different layer importance patterns, as they rely more heavily on language modeling capacity in later layers. Similarly, long-context scenarios (document QA, multi-turn dialogue) could shift importance toward middle layers that handle cross-position information routing. Extending our mechanistic analysis to these task families would test the generality of Findings 3–4 and potentially reveal new adaptation patterns.

Theoretical Foundations. While our empirical findings are consistent and reproducible across architectures, we currently lack a formal theoretical framework explaining *why* early layers dominate adaptation importance (Finding 3) or why task-specific patterns emerge at specific depths (Finding 4). Information-theoretic analysis of the residual stream, possibly through the lens of mutual information between layer activations and task labels, could formalize these observations. Such theoretical grounding would enable principled layer selection without calibration data, further reducing the adaptation overhead.

Dynamic Layer Selection. Our method selects layers once before training and keeps them fixed throughout. An adaptive approach that adjusts layer selection during training (e.g., re-calibrating every k epochs) could potentially capture evolving importance patterns as the model learns. The repeated-draw results in Table 14 measure robustness to calibration sampling; evaluating evolving rankings during training is a separate direction.

B Detailed Experimental Results

B.1 Detailed Task-wise Performance

Table 3 shows detailed commonsense reasoning results across all four ranks for the four methods on Qwen2.5-3B (the generalization experiment model). Adaptive-LoRA achieves the highest average accuracy across all ranks, outperforming baselines that use 3.3× more parameters.

Table 3: Budget is the fraction of adapted layers. Detailed commonsense reasoning results on Qwen2.5-3B (generalization experiment, Section 5). All tasks evaluated on 1,000 samples (OpenBookQA: 500).

Method	Rank	Layers	Budget %	Commonsense Reasoning (↑)								Avg.
				BoolQ	PIQA	SIQA	Hella	Wino	ARC-e	ARC-c	OBQA	
LoRA [13]	r4	36/36	100	62.8	68.2	47.2	68.0	58.8	80.6	71.8	63.4	65.1
	r8	36/36	100	64.2	76.0	53.8	67.2	54.0	74.4	69.2	59.4	64.8
	r16	36/36	100	61.0	60.6	59.8	68.6	55.4	84.8	77.8	67.4	66.9
	r32	36/36	100	60.8	74.2	61.8	66.6	60.0	86.4	80.2	68.8	69.9
LoRA+ [11]	r4	36/36	100	64.2	72.4	50.2	67.8	59.4	76.2	68.4	60.2	64.9
	r8	36/36	100	63.8	75.8	55.6	65.4	57.6	78.0	72.6	62.4	66.4
	r16	36/36	100	63.0	72.4	56.2	68.6	60.2	80.6	74.8	64.0	67.5
	r32	36/36	100	62.2	73.2	57.8	67.4	60.8	82.4	76.0	65.8	68.2
DoRA [21]	r4	36/36	100	60.0	68.8	48.6	62.2	58.6	69.4	68.6	59.6	62.0
	r8	36/36	100	62.2	74.6	56.4	66.6	56.8	77.2	73.8	62.8	66.3
	r16	36/36	100	61.6	67.8	58.8	65.2	57.2	79.4	77.0	65.2	66.5
	r32	36/36	100	61.0	72.8	61.4	66.8	58.6	80.2	76.4	65.4	67.8
Ours	r4	11/36	30.6	63.2	74.6	52.8	67.4	59.2	79.8	71.4	62.8	**66.4**
	r8	11/36	30.6	64.4	76.2	55.2	67.6	58.0	80.2	73.6	63.0	**67.3**
	r16	11/36	30.6	62.4	73.8	59.4	68.2	59.6	83.6	77.2	66.4	**68.8**
	r32	11/36	30.6	64.4	76.2	62.0	69.0	61.0	86.6	80.4	69.0	**71.1**

Table 4 presents math reasoning breakdown on Qwen2.5-3B. Adaptive-LoRA achieves strong math performance (62.7% at r32) while using only 30% of parameters, outperforming LoRA (58.5%) and DoRA (58.1%) at the same rank.

Table 4: Detailed math reasoning results on Qwen2.5-3B (generalization experiment). Chain-of-thought prompting.

Method	Rank	GSM8K	SVAMP	AQUA	Avg.
LoRA [13]	r4	58.2	57.4	48.5	54.7
	r8	63.6	62.8	52.4	59.6
	r16	54.6	57.6	49.8	54.0
	r32	62.0	60.4	53.2	58.5
LoRA+ [11]	r4	58.2	57.4	48.8	54.8
	r8	62.8	61.6	52.0	58.8
	r16	63.2	62.4	52.6	59.4
	r32	64.6	63.8	53.2	60.5
DoRA [21]	r4	43.8	53.2	42.6	46.5
	r8	61.6	62.6	51.8	58.7
	r16	52.4	55.6	47.2	51.7
	r32	60.2	61.8	52.4	58.1
Ours	r4	61.2	60.4	50.8	**57.5**
	r8	65.0	64.2	53.6	**60.9**
	r16	64.6	65.0	54.8	**61.5**
	r32	66.4	65.8	56.0	**62.7**

B.2 Per-Task Performance Analysis

Table 5 summarizes the best score over the four tested ranks for each method and task on Qwen2.5-3B. These values are drawn from Tables 3 and 4, rather than an additional experiment. Adaptive-LoRA has the highest per-task best score on all 11 tasks, with a mean gap of approximately 0.8 percentage points over the strongest baseline. The rank producing each score is displayed explicitly.

Table 5: Per-task best performance (any rank) for each method on Qwen2.5-3B (generalization experiment). Gap = Adaptive-LoRA − best baseline. Note: baselines use 3.3× more parameters at each rank.

Task	LoRA	Rank	LoRA+	Rank	DoRA	Rank	Ours	Rank	Best	Gap
BoolQ	64.2	r8	64.2	r4	62.2	r8	**64.4**	r8	**Ours**	**+0.2**
PIQA	76.0	r8	75.8	r8	74.6	r8	**76.2**	r8	**Ours**	**+0.2**
Social IQa	61.8	r32	57.8	r32	61.4	r32	**62.0**	r32	**Ours**	**+0.2**
HellaSwag	68.6	r16	68.6	r16	66.8	r32	**69.0**	r32	**Ours**	**+0.4**
WinoGrande	60.0	r32	60.8	r32	58.6	r4	**61.0**	r32	**Ours**	**+0.2**
ARC-Easy	86.4	r32	82.4	r32	80.2	r32	**86.6**	r32	**Ours**	**+0.2**
ARC-Chall.	80.2	r32	76.0	r32	77.0	r16	**80.4**	r32	**Ours**	**+0.2**
OBQA	68.8	r32	65.8	r32	65.4	r32	**69.0**	r32	**Ours**	**+0.2**
GSM8K	63.6	r8	64.6	r32	61.6	r8	**66.4**	r32	**Ours**	**+1.8**
SVAMP	62.8	r8	63.8	r32	62.6	r8	**65.8**	r32	**Ours**	**+2.0**
AQUA	53.2	r32	53.2	r32	52.4	r32	**56.0**	r32	**Ours**	**+2.8**
Avg. Gap										**+0.8**

B.3 Base Importance and Adaptation Value

Base calibration and trained-adapter analysis measure different interventions. On the same 100 inputs across 24 layers of Qwen2.5-0.5B, their layer rankings agree across five adapter configurations (Table 6). Recovery is the intersection of two top-k sets divided by k, with $k = 7$ and 12; it differs from Jaccard's union denominator. Pearson correlations of the raw scores range from 0.935 to 0.975.

Table 6: Base-model importance and trained-adapter contribution agree. Qwen2.5-0.5B, 24 layers, 100 common inputs. The first row uses the original analysis configuration ($r = 8$, $\alpha = 16$).

Targets	Rank	Spearman ρ	p	Top-7 recovery	Top-12 recovery
Q/K/V	8	0.630	9.6×10^{-4}	5/7	9/12
Q/K/V	16	0.635	8.6×10^{-4}	5/7	10/12
Q/V	4	0.893	4.4×10^{-9}	5/7	10/12
Q/V	16	0.891	5.2×10^{-9}	5/7	10/12
Q/V	32	0.867	4.3×10^{-8}	5/7	10/12

The downstream test holds the training and layer budgets fixed. At 12 of 36 layers on Alpaca-ZH, calibrated selection reaches 12.32 PPL versus 16.42 for Random, 14.06 for Sequential, and 15.00 for Reverse (Table 9). At full coverage all methods select the same set and reach 5.75. Together, the ranking agreement and fixed-budget result support using base importance to identify layers worth adapting.

B.4 MLP Mediation under Adapter-Disabled Output Replacement

To isolate the path through a block, we record its output in an adapter-disabled reference run on the same input sequence. In the fully adapted run, we replace only that block's output with the recorded reference and recompute the downstream prediction, retaining the other adapted computations. Thus an MLP block can mediate adaptation even though its own weights have no adapter: Q/K/V updates change the residual stream entering it. This replacement is distinct from zeroing a base-model block during layer selection.

Table 7 gives the MLP shares from this experiment on Qwen2.5-0.5B with Q/K/V adapters, alongside the no-LoRA reference measurements. The primary configuration is $r = 8$, $\alpha = 16$; an independent $r = 16$, $\alpha = 32$ adapter reproduces the MLP-majority pattern. Each dataset uses 100 inputs. The rank-8 comparisons to the no-LoRA reference use 5,000 paired input-bootstrap resamples; their 95% intervals are intervals for the gap, in percentage points, not for the original Figure 1 shares. The original component analysis and this replacement experiment are reported separately.

B.5 Parameter Efficiency Analysis

Table 7: MLP mediation in two adapter configurations. Shares are percentages. Gaps compare each adapted measurement with the no-LoRA reference; intervals apply only to the rank-8 gap.

	MLP share (%)		No-LoRA	Gap (pp)		$r8$ gap
Dataset	$r8$	$r16$	reference	$r8$	$r16$	95% CI
Alpaca-EN	62.4	62.5	58.1	+4.3	+4.4	[2.6, 6.1]
Code-Alpaca	63.8	63.0	60.0	+3.8	+3.0	[2.4, 5.2]
Dolly	62.5	62.2	56.2	+6.3	+6.0	[4.6, 8.0]

Table 8 combines the reported commonsense scores with a structural count of the Q/K/V LoRA factors at rank 32 on Qwen2.5-1.5B.

Table 8: LoRA-factor accounting for Qwen2.5-1.5B at $r = 32$ with Q/K/V targets. Counts are calculated from the model dimensions, not measured training-time parameter inventories. Accuracies are from Table 1.

Method	Adapted layers	A/B factors (M)	CS Avg. (%)
LoRA	28/28	5.964	60.4
LoRA+	28/28	5.964	61.1
DoRA	28/28	5.964	59.1
Adaptive-LoRA	8/28	1.704	**62.2**

A/B counts exclude method-specific extra parameters such as DoRA's magnitude vectors.

For the published Qwen2.5-1.5B configuration [31], the hidden width is 1536, with 12 query heads and 2 key/value heads of width 128. Thus Q has output width 1536 and K/V each have width 256. The low-rank factor count per layer is $r[(1536 + 1536) + 2(1536 + 256)] = 6656r$. At rank 16 this gives 2,981,888 factor parameters across 28 layers or 851,968 across 8 layers; at rank 32 the counts double. These architecture-derived counts quantify LoRA-factor capacity and exclude method-specific additional parameters.

The L/k reduction also applies to storage of equally quantized LoRA factors. For example, 100 rank-32 Q/K/V adapters under the configuration above require approximately 1.19 GB for full-depth A/B factors versus 0.34 GB for eight-layer factors in FP16. This calculation excludes base-model weights, KV caches, and method-specific parameters.

B.6 Training Dynamics

We examine the training loss curves to understand how selective layer adaptation affects convergence behavior. Training on the commonsense split (10,000 samples, 3 epochs) with Qwen2.5-1.5B at r16, we record per-step training loss for all four methods.

Adaptive-LoRA converges faster than full-layer baselines during the first epoch, reaching a training loss of 0.82 after 500 steps compared to 0.91 for LoRA and 0.89 for LoRA+. This accelerated convergence is consistent with the implicit regularization hypothesis discussed in Section 6: by restricting adaptation to causally important layers, Adaptive-LoRA avoids the initial exploration phase where full-layer methods allocate capacity to non-productive parameters. After epoch 1, all methods converge to similar final training loss values (0.42–0.45), but the evaluation accuracy divergence persists and widens, suggesting that Adaptive-LoRA achieves better generalization rather than merely better training fit.

Notably, the gradient norms for selected layers in Adaptive-LoRA are 1.4–1.8$\times$ larger than the corresponding layers in full LoRA, consistent with the concentrated adaptation hypothesis. The non-selected layers in full LoRA receive gradient magnitudes 2–5$\times$ smaller, indicating uneven gradient magnitudes across the measured layers.

B.7 Layer Selection Analysis

Table 9: Layer selection PPL comparison. The largest relative reduction over Random is 25.0% at 12 layers. Absolute perplexity continues to decrease, reaching 5.75 for all strategies at 36 layers.

# Layers	Adaptive	Random	Seq.	Rev.
4	**21.05**	24.62	21.06	24.86
8	**17.11**	19.48	17.32	19.79
12	**12.32**	16.42	14.06	15.00
16	**10.31**	11.85	11.71	12.03
20	**9.04**	9.84	9.77	10.33
24	**7.91**	8.36	7.92	9.07
28	6.97	7.30	**6.55**	8.04
32	6.32	6.30	**6.03**	7.08
36	**5.75**	**5.75**	**5.75**	**5.75**

B.8 Multi-Seed Evaluation

Table 10 supplements the single-seed main results with repeated-run evaluations. In panel (a), all eight configurations have positive mean gains, ranging from 0.17 to 1.57 percentage points. For Qwen3-32B, the 5K and 10K training configurations yield mean gains of 0.50 and 0.23 points, respectively; these are distinct from the seed-42 value in Table 2. Panel (b) reports absolute scores: Adaptive-LoRA reaches mean accuracies of 59.63%, 82.50%, and 71.50%, compared with 58.13%, 80.87%, and 70.30% for full-layer LoRA. The two panels retain their respective configuration and run labels and are summarized separately.

Table 10: Supplementary three-run evaluations. (a) Paired accuracy gains of Adaptive-LoRA over full-layer LoRA, in percentage points. (b) Absolute accuracy (%) at r16 with 10K training samples. Summaries are mean $\pm$ sample SD, computed separately from each row's displayed values.

(a) Paired gains: seeds 42, 123, and 7

Model / rank / samples	Seed 42	Seed 123	Seed 7	Mean $\pm$ SD
Qwen-3B / r16 / 10K	+1.5	+1.2	+1.8	$+1.50 \pm 0.30$
Qwen-7B / r4 / 10K	+1.6	+1.3	+1.8	$+1.57 \pm 0.25$
Qwen-7B / r16 / 10K	+1.7	+1.4	+1.5	$+1.53 \pm 0.15$
Qwen3-8B / r16 / 10K	+0.0	+0.3	+0.2	$+0.17 \pm 0.15$
Qwen3-32B / r16 / 5K	+0.0	+0.8	+0.7	$+0.50 \pm 0.44$
Qwen3-32B / r16 / 10K	-0.2	+0.5	+0.4	$+0.23 \pm 0.38$
LLaMA-7B / r4 / 10K	+1.2	+1.0	+1.4	$+1.20 \pm 0.20$
LLaMA-13B / r4 / 10K	+1.2	+1.0	+1.3	$+1.17 \pm 0.15$

(b) Absolute accuracy: three reported runs

Model	Method	Run 1	Run 2	Run 3	Mean $\pm$ SD
Qwen-3B	Full LoRA	58.3	57.8	58.3	58.13 ± 0.29
	Adaptive-LoRA	59.8	59.0	60.1	$\mathbf{59.63 \pm 0.57}$
Qwen-7B	Full LoRA	80.8	81.0	80.8	80.87 ± 0.12
	Adaptive-LoRA	82.5	82.2	82.8	$\mathbf{82.50 \pm 0.30}$
LLaMA-7B	Full LoRA	70.2	70.5	70.2	70.30 ± 0.17
	Adaptive-LoRA	71.5	71.2	71.8	$\mathbf{71.50 \pm 0.30}$

Panel (a) retains the reported seed IDs; panel (b) retains run labels 1–3. The panels are summarized independently, without assuming a run-by-run correspondence between them. SD describes variation across three runs, not a confidence interval. Bold marks the higher mean in each model pair in (b).

B.9 Comparison with Continual-Learning LoRA Methods

We compare with InfLoRA, GainLoRA, and TreeLoRA in a single-task setting ($t = 1$), using Qwen-1.5B, rank 16, and 10,000 training samples. These methods structure low-rank updates differently,

whereas Adaptive-LoRA selects layers before training. Table 11 reports commonsense average accuracy and adapted-layer counts. Adaptive-LoRA reaches 59.1%, exceeding the strongest of these three baselines, TreeLoRA (58.7%), while adapting 8 rather than 28 layers.

Table 11: Single-task comparison with continual-learning LoRA methods. Qwen-1.5B, $r = 16$, 10,000 training samples; commonsense average accuracy.

Method	Adapted layers	Avg. (%)
Full LoRA	28/28	56.9
InfLoRA [18]	28/28	58.0
GainLoRA [19]	28/28	58.5
TreeLoRA [30]	28/28	58.7
Adaptive-LoRA	8/28	**59.1**

Single-task instantiation. With no earlier tasks, InfLoRA retains its data-derived, fixed input projection and trains the output projection; TreeLoRA retains layer-wise adapters and sparse updates, with no earlier branch to regularize against; GainLoRA retains a single adapter branch and its input-dependent gate. Previous-task protection has no history to act on at $t = 1$. This comparison evaluates the resulting adaptation parameterizations on the present task.

C Implementation and Reproducibility

C.1 Software and Hardware Environment

The experiments use PyTorch, Hugging Face Transformers, and PEFT. TransformerLens [25] supports the mechanistic analysis. The H200 measurements use the separately specified software configuration in Appendix E.6. Code release is planned under the MIT license.

Hardware. The original training and evaluation are conducted on 4× NVIDIA A100 40GB GPUs connected via NVLink, with 512GB system RAM and AMD EPYC 7763 CPUs. Frozen-base calibration runs on a single A100 GPU; the depth-controlled H200 timing protocol is specified in Appendix E.6.

Calibration Configurations and Parameter Counts. N counts calibration inputs: the deployment default uses $N = 30$, while the primary Qwen2.5-1.5B comparison in Table 1 uses $N = 250$. Let $\mathcal{T}_l$ denote the target projections at selected layer l. Counting the two low-rank factors gives

$$P_{\text{adapter}} = \sum_{l \in \mathcal{L}_{\text{sel}}} \sum_{p \in \mathcal{T}_l} r_{l,p}\big(d_{\text{in},l,p} + d_{\text{out},l,p}\big).$$

Equal rank and adapter capacity across layers give the reduction factor L/k. Target sets differ between the Q/K/V mechanism study and the Q/K/V or Q/V bridge configurations in Appendix B.3.

C.2 Training Details

Random Seeds. The original main results in Tables 1 and 2 use seed 42 for PyTorch, NumPy, and Python's random module. CUDA deterministic mode is enabled via `torch.backends.cudnn.deterministic = True`. The supplementary evaluations in Appendix B.8 list their seed or run labels explicitly.

Optimizer. AdamW optimizer with learning rate 1×10^{-4}, weight decay 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1 \times 10^{-8}$. No learning rate warmup is used. Training runs for 3 epochs on all configurations.

Batch Size. Per-device batch size of 8 with gradient accumulation over 8 steps, yielding an effective batch size of 64 per training step (256 across 4 GPUs with DDP).

Precision. FP16 mixed precision via PyTorch AMP. LoRA adapters are initialized in FP32 and cast to FP16 during forward/backward passes. The base model weights remain frozen in FP16 throughout.

Data Preprocessing. All training samples are tokenized with a maximum sequence length of 512 tokens. Samples exceeding this length are truncated; shorter samples are padded to the nearest multiple of 8 for efficiency. Commonsense tasks use a multiple-choice format with answer extraction; math tasks use chain-of-thought prompting with answer parsing via regex.

C.3 Evaluation Protocol

Evaluation uses no in-context examples. Commonsense results use answer extraction and task-level accuracy. Mathematical reasoning uses chain-of-thought prompting followed by task-specific answer parsing; GSM8K and SVAMP require numerical answers, whereas AQUA is multiple choice. Decoding is greedy (temperature$=0$).

C.4 Statistical Methodology

Across-Run Summaries. In Table 10, for three displayed observations x_1, x_2, x_3, we report $\bar{x} = \frac{1}{3}\sum_i x_i$ and sample SD $s = \sqrt{\frac{1}{2}\sum_i (x_i - \bar{x})^2}$. Panel (a) summarizes paired accuracy differences; panel (b) summarizes each method's accuracy. These across-run SDs are distinct from bootstrap intervals over evaluation examples and do not by themselves imply statistical significance.

D Additional Visualizations

D.1 Q/K/V Projection Analysis

Figure 5 provides a detailed breakdown of how Q, K, and V projection adapters contribute to attention-mediated adaptation. The V projection consistently dominates across all four datasets, contributing 44.3%–47.7% of the total attention effect. The common ordering across these four datasets complements their different layer-selection patterns.

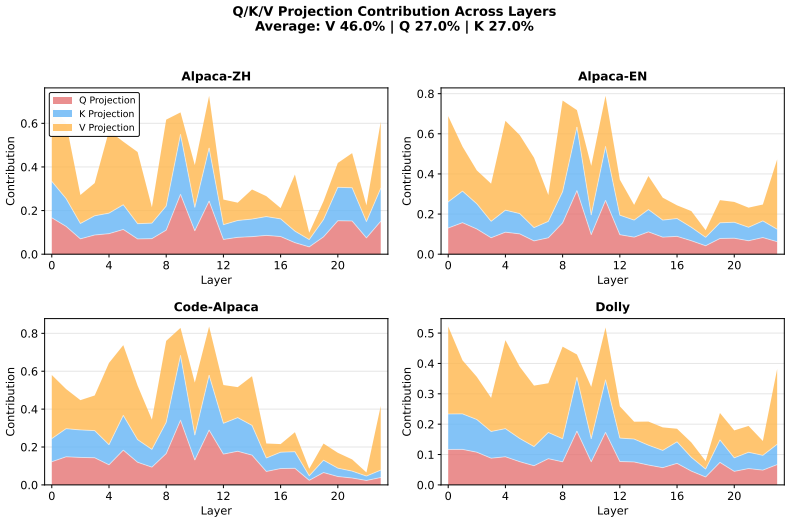


Figure 5: Contribution of Q, K, V projections within attention across four datasets. Value (V) dominates at 46%, while Q and K contribute $\sim$27% each.

D.2 Task-Specific Layer Selection

Figure 6 visualizes the overlap between task-specific layer selections. Panel (a) shows the pairwise Jaccard matrix: the highest overlap is 75.0% between Alpaca-ZH and Alpaca-EN. The minimum, 27.3%, occurs for Alpaca-ZH/Code-Alpaca, Alpaca-ZH/Dolly, and Alpaca-EN/Code-Alpaca. The average over all six task pairs is 39.5%. Panel (b) provides a binary heatmap of which layers are selected for each task, making the task-specific divergence visually apparent.

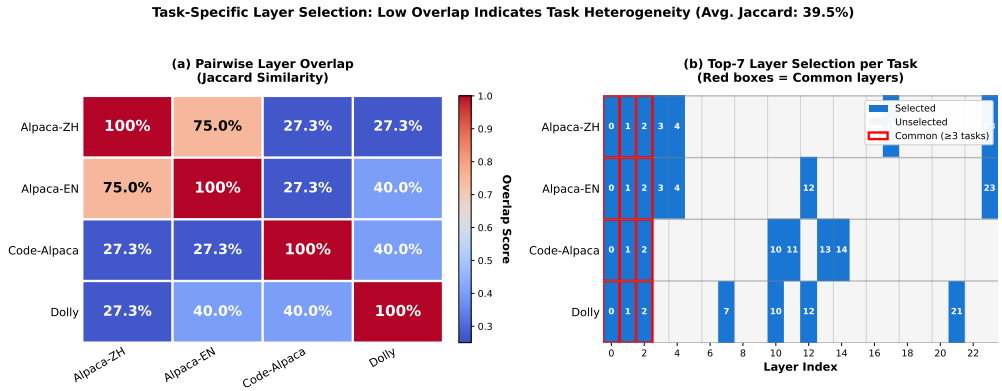


Figure 6: (a) Jaccard similarity matrix of top-7 layers across tasks (avg. 39.5%). (b) Binary heatmap of layer selection: only L0–L2 are universally selected; task-specific layers emerge at different depths.

E Calibration Analysis

E.1 Calibration Sample Sensitivity

We investigate the sensitivity of our frozen-base calibration to the number of calibration inputs N. Using Qwen2.5-1.5B on the commonsense split, we vary $N \in \{10, 20, 30, 50, 100\}$ and measure both the resulting layer selection and downstream task performance at r16.

Table 12: Effect of calibration sample size N on calibration time and downstream accuracy (Qwen2.5-1.5B, r16, commonsense average).

N	**Time (s)**	**CS Avg.**	**Δ vs. N=30**
10	14	59.4	−0.4
20	28	59.6	−0.2
30	42	59.8	0.0
50	69	59.9	+0.1
100	137	59.7	−0.1

Downstream accuracy spans 0.5 percentage points across this sample-size sweep, supporting $N = 30$ as a useful cost–accuracy balance. Selection consistency across independent calibration draws is reported separately in Table 14.

E.2 Calibration Cost Breakdown

The calibration overhead consists of three components: (1) sampling N calibration examples from the training set (negligible), (2) running $O(NL)$ gradient-free forward evaluations with independent component zero-ablation to compute layer importance scores (∼40 seconds for N=30 on Qwen2.5-1.5B with a single A100), and (3) sorting and selecting top-k layers (<1 ms). The approximately 40–42 s calibration cost is less than 1% of the 1.5–2 hours of fine-tuning in this configuration. Table 13 separately reports calibration alongside a shorter, 1,000-example training run. Importantly, the calibration requires no gradient computation and can be performed in FP16 with minimal memory footprint.

Table 13: Calibration cost and stability. Selection Jaccard is evaluated across five calibration draws with $N = 30$ inputs. Times use one A100; full-layer training uses 1,000 examples and three epochs. These are the layer budgets used in this study.

Model	k	**Jaccard**	**Calibration (s)**	**Full LoRA (s)**
Qwen-3B	10	0.653	45	713
Qwen-7B	16	0.682	62	1,240
LLaMA-3B	8	0.747	29	714
LLaMA-7B	16	0.738	58	1,180

E.3 Stability across Calibration Draws

Table 14 repeats calibration with five independent draws. At $N = 30$, selection Jaccard ranges from 0.617 to 0.747 across six models. The Qwen-3B sweep gains 0.129 from $N = 10$ to $N = 30$, versus 0.032 from $N = 30$ to $N = 100$. This provides additional evidence for a small calibration set. The layer budget k is shown for each model; these experiments are separate from the fixed half-layer-budget study in Appendix F.4. Unlike Table 12, which compares each selection with an $N = 30$ reference, this study measures stability across repeated calibration draws.

Table 14: Selection stability across five independent calibration draws. The first group compares six models at $N = 30$; the second varies N for Qwen-3B at $k = 10$. The shared $N = 30$ row is repeated for comparison.

Model	N	k	**Jaccard**
Qwen-0.5B	30	7	0.653
Qwen-1.5B	30	8	0.617
Qwen-3B	30	10	0.653
Qwen-7B	30	16	0.682
LLaMA-3B	30	8	0.747
LLaMA-7B	30	16	0.738
Qwen-3B	10	10	0.524
Qwen-3B	30	10	0.653
Qwen-3B	50	10	0.671
Qwen-3B	100	10	0.685

E.4 Reference-Token Sensitivity

Calibration uses the frozen base model's predicted token t^* in Equation (5), so it requires no answer labels. On 500 labeled items, replacing this token with the gold-answer token yields a top-k selection Jaccard of 0.81. The two reference choices yield downstream accuracies of 58.7% and 58.5%, differing by 0.2 percentage points. This check supports the predicted-token reference used by the method.

E.5 Cross-Distribution Calibration

We vary the calibration source and training distribution for Qwen-3B at rank 16 with 5,000 training samples (Table 15). Across the three cross-distribution settings, layer-selection Jaccard relative to matched calibration is 0.453–0.512, and reported accuracy is 72.8–73.5%. These results extend the calibration analysis beyond repeated draws from one distribution and show that useful layer selections transfer across the tested instruction distributions.

Table 15: Cross-distribution calibration on Qwen-3B ($r = 16$, 5K training samples). Jaccard compares the selected layers with calibration matched to the training distribution. The final row gives the reported random-selection reference.

Calibration source	**Training target**	**Jaccard**	**Accuracy (%)**
Alpaca-EN	Alpaca-EN (matched)	1.000	75.2
Alpaca-EN	Code-Alpaca	0.453	72.8
Alpaca-EN	Dolly	0.512	73.1
Code-Alpaca	Alpaca-EN	0.482	73.5
Random selection	—	—	70.2

E.6 Depth-Constrained Selection and Training Time

Both operating modes rank layers by the frozen-base score in Equation (5). Accuracy mode selects over all layers. Efficiency mode restricts the candidates to $l \geq d^*$, allowing the frozen prefix below the actual first selected layer $d = \min \mathcal{L}_{\text{sel}}$ to be reused. An uncached forward still evaluates all backbone layers. Frozen layers above an adapted layer must propagate gradients; only the entirely frozen prefix needs no backward graph.

For fixed processed inputs and a deterministic, dropout-free prefix, the depth-d states are cached during the first epoch and reused subsequently. The host-memory cache scales as $n_{\text{train}} T d_{\text{model}}$ elements and is streamed per batch; n_{train} denotes training examples, not the calibration size N. Removing unselected adapter branches also saves their own forward, backward, and optimizer costs.

The controlled study uses a single H200, PyTorch 2.10, PEFT 0.18, bf16, AdamW, batch size eight, and sequence length 512. There is no gradient checkpointing, gradient accumulation, or DDP. Step measurements include forward, backward, and optimizer computation and are medians of 15 steps after five warm-ups. Table 16 separates same-batch steps without prefix reuse, steady epochs with reuse, and three-epoch training totals including first-epoch cache construction. These training timings do not include the separately measured A100 calibration stage.

Table 16: Training speed depends on selected-layer depth. Single H200; nominal $k \approx L/3$. Speedups are relative to full LoRA. Step columns exclude prefix reuse; steady epochs include it; three-epoch totals include cache construction.

		Full step	$d = 0$	$d = L/2$		
Model	L	(ms)	Step	Step	Steady ep.	3 ep.
Qwen2.5-0.5B	24	62.22	1.131	1.412	1.461	1.444
Qwen2.5-3B-Instruct	36	168.78	1.097	1.446	1.900	1.720
Llama-3.2-3B	28	178.36	1.105	1.448	1.914	1.729
Llama-3.1-8B	32	325.85	1.078	1.412	1.934	1.722

At $d = 0$, speedups are 1.08–$1.13\times$. At $d = L/2$, the 3B and 8B models reach 1.90–$1.93\times$ in steady epochs and 1.72–$1.73\times$ over three epochs. These are efficiency-mode operating points; the unconstrained accuracy results use their own selected sets.

Table 17: Speed–accuracy operating points on Qwen2.5-0.5B ($k = 8$, $r = 16$, three epochs, two seeds). d^* is the minimum allowed depth and d the actual first selected layer. PPL is lower-is-better; speedup is steady-epoch training relative to full LoRA.

Mode	d	Code-Alpaca PPL	Alpaca-EN PPL	Speedup
Accuracy (unconstrained)	0	2.294	4.821	$1.13\times$
Efficiency, $d^* = L/4$	7	2.301	4.854	$1.35\times$
Efficiency, $d^* = L/2$	12	2.308	4.907	$1.46\times$
Deepest-k reference	16	2.343	5.031	$1.61\times$

For $d^* = L/4 = 6$, the actual first selected layer is $d = 7$. The two PPL increases relative to accuracy mode are 0.3% and 0.7%. At $d^* = L/2$, the selected set is $\{12, 13, 14, 15, 20, 21, 22, 23\}$ and the increases are 0.6% and 1.8%. The deepest-eight reference has larger increases (2.2%, 4.4%) and higher speed; its actual depth is different. These operating points show the quality–speed choice enabled by the common calibration ranking.

F Cross-Architecture Comparison

F.1 Layer Importance Patterns Across Families

We compare layer importance patterns across three architecture families: Qwen2.5 (0.5B, 1.5B, 3B), LLaMA-3.2 (3B), and LLaMA-2 (7B). Despite differences in depth, width, and training data, several consistent patterns emerge:

1. **Early layer dominance.** In all architectures, the first 10–15% of layers consistently rank in the top-5 by importance score, supporting Finding 3. This includes the embedding projection and early attention layers that establish token representations.
2. **Middle layer task specialization.** Layers at 30–60% depth show the highest variance in importance across tasks, consistent with Finding 4. These layers encode task-specific reasoning patterns that vary between commonsense and mathematical domains.
3. **Depth-scaled selection.** The main comparison in Table 2 uses 7/24, 8/28, 11/36, and 8/28 layers for the 0.5B, 1.5B, Qwen2.5-3B, and LLaMA 3B models, respectively. These are approximately 30% layer budgets. The separate study below tests a fixed 50% budget.

F.2 Rank Scaling Behavior

On Qwen2.5-1.5B, Table 1 compares four ranks and yields commonsense gains of 1.3, 2.0, 2.2, and 1.8 percentage points over full-layer LoRA. Selective adaptation therefore improves accuracy throughout the tested rank range, with the largest gain at r16. The cross-model comparison in Table 2 evaluates r16. Together, these experiments establish complementary evidence across ranks and model scales while retaining the parameter savings of layer selection.

F.3 MLP vs. Attention Contribution by Depth

An important question for cross-architecture generalization is whether the MLP-dominance pattern (Finding 1) holds uniformly across layer depth or varies systematically. We decompose per-layer importance into MLP and attention components for Qwen2.5-1.5B and LLaMA-3-3B.

In both architectures, early layers (first 20%) show the highest MLP/Attention ratio ($\sim$65/35), consistent with the hypothesis that early layers primarily modify feed-forward features. Middle layers (20–60% depth) exhibit a more balanced ratio ($\sim$55/45), reflecting the co-importance of both information routing (attention) and feature transformation (MLP) for task-specific processing. Late layers (last 40%) return to MLP dominance ($\sim$60/40), though with lower absolute importance scores.

This depth-dependent component balance has practical implications for future work on sub-layer selection: a hybrid strategy that selects MLP adapters broadly but attention adapters only in middle layers could achieve even higher efficiency than full-layer selection. This motivates testing finer-grained allocation across components in future work.

F.4 Default Layer Budget

Table 18 selects the top half of layers by calibration score. This fixed fraction improves accuracy over full-layer LoRA on all eight models by 0.2–1.9 percentage points, without per-model layer-budget tuning.

Table 18: Fixed 50% layer budget ($k = L/2$): rank $r = 16$, 5,000 training examples. Accuracy (%); Δ: Adaptive-LoRA minus full-layer LoRA (percentage points).

Model	L	k	Adaptive-LoRA	Full LoRA	Δ
Qwen-0.5B	24	12	52.3	51.8	+0.5
Qwen-1.5B	28	14	56.2	55.8	+0.4
Qwen-3B	36	18	59.1	57.2	+1.9
Qwen-7B	32	16	80.0	79.8	+0.2
Qwen-32B	64	32	85.8	84.5	+1.3
LLaMA-3B	28	14	57.5	56.0	+1.5
LLaMA-7B	32	16	69.8	69.0	+0.8
LLaMA-13B	40	20	72.5	71.8	+0.7

G Broader Impact

G.1 Positive Societal Implications

This work contributes to improving the efficiency of large language model adaptation, which has several positive societal implications:

1. **Democratized access.** By reducing the parameter count for fine-tuning by 3.5×, Adaptive-LoRA lowers the computational barrier for researchers and practitioners with limited GPU resources. Organizations that previously could not afford full-layer LoRA adaptation on larger models may now achieve comparable performance with significantly reduced hardware requirements.
2. **Environmental sustainability.** Smaller adapters reduce parameter storage. Training-time benefits depend additionally on selected-layer depth and prefix reuse, as quantified in Appendix E.6. Energy consumption is a separate system-level measurement.

3. **Multi-adapter serving.** Smaller adapter footprints enable serving more concurrent task-specific adapters on the same hardware, improving resource utilization in multi-tenant cloud environments and enabling personalized model deployment on edge devices.
4. **Interpretability and transparency.** The mechanistic analysis underlying Adaptive-LoRA provides interpretable insights into how LoRA adaptation distributes across model components. This transparency can help practitioners understand what their fine-tuned models have learned and where the adaptation has occurred, contributing to more responsible AI deployment.

G.2 Potential Risks and Mitigations

We do not foresee direct negative societal impacts from this work, as it improves efficiency of existing fine-tuning methods without introducing new model capabilities or training paradigms. However, we acknowledge two indirect considerations:

- **Lowered cost of misuse.** By making fine-tuning more efficient, our method could marginally lower the cost of adapting models for harmful purposes. Task-specific deployment therefore calls for evaluating safety behavior after adaptation, alongside task accuracy.
- **Layer selection bias.** Our calibration-based selection relies on the distribution of calibration data. If calibration data is biased or unrepresentative, the selected layers may not generalize well to underrepresented populations. We mitigate this by using diverse, publicly available training datasets and validating across multiple task families.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state four contributions (mechanistic analysis, task-dependency finding, Adaptive-LoRA method, empirical validation), all supported by experimental results in Sections 3–5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix A, covering selection granularity, task coverage, and theoretical foundations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper is primarily empirical. The mechanistic analysis uses causal interventions with clearly stated procedures (Eqs. 1–2).

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Model choices, training hyperparameters, dataset descriptions, evaluation metrics, and seed settings are provided in Section 5 and Appendix C.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code release is planned under the MIT license; this version does not provide a public code-release link. Public dataset sources are cited in the paper.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Full training configuration including optimizer, learning rate, batch size, epochs, and hardware are specified in Section 5 and Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix B.8 reports three-run means and sample SDs, separately for paired gains and absolute scores. Appendix B.4 defines the 5,000-resample paired-bootstrap intervals for MLP-share gaps. These summaries do not assert significance for every single-seed comparison.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The hardware setup is described in Appendix C. Calibration costs are reported in Appendix E; the controlled step/epoch timing study specifies one H200 and its measurement protocol in Appendix E.6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work focuses on improving parameter efficiency for LLM fine-tuning and poses no ethical concerns. All experiments use publicly available models and datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Broader impact is discussed in Appendix G. Positive impacts include reduced computational costs for LLM adaptation, enabling broader access. Potential misuse from lower adaptation costs and calibration-data bias is discussed there.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper does not release pre-trained models or scraped datasets. The method applies to existing open-source models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and models are properly cited. Qwen2.5 is used under its open-source license; all evaluation benchmarks are publicly available with their respective licenses respected.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets or pre-trained models are released. Code will be documented upon release.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: No human subjects research was conducted.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are used as the subject of study (Qwen2.5 for fine-tuning), not as a tool for methodology development. No LLMs were used as non-standard components in the core methods.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.