LEARNING WHAT TO PRESERVE AND WHERE TO ADAPT IN LORA

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation (LoRA) enables parameter-efficient fine-tuning, yet learning a new task can still compromise a model's existing capabilities. To balance adaptation with retention, existing approaches guide capacity allocation by protecting dominant singular subspaces or amplifying task-specific directions. Yet identifying a task-dependent component leaves open whether to preserve its existing mapping or allocate new capacity along its input direction. We find that using components with high task dependence but low pre-trained spectral strength as adaptation support effectively balances downstream learning with performance on held-out tasks outside the fine-tuning domain. Our method, Capacity Allocation through Role Discovery for Low-Rank Adaptation (CARD-LoRA), probes task dependence with suppression gates and combines it with spectral strength to select the input directions a LoRA update can act on, while leaving its output correction freely learnable. On LLaMA-2-7B, CARD-LoRA achieves the highest combined downstream and held-out evaluation average among the compared methods in both commonsense and mathematical fine-tuning. The gains over vanilla LoRA are 1.23 and 0.61 percentage points, respectively. Restriction-side ablations further show the benefit of learning the output correction independently of the selected components' pre-trained output directions. More broadly, this work frames low-rank adaptation as deciding where to intervene in a pre-trained model, not only how to parameterize its update.

1 INTRODUCTION

Fine-tuning a pre-trained language model for a specialized task can improve target-domain performance at the expense of capabilities outside that domain (Biderman et al., 2024). Parameter-efficient fine-tuning (PEFT) adapts models through compact adapters (Houlsby et al., 2019; Karimi Mahabadi et al., 2021), continuous prompts (Li & Liang, 2021; Lester et al., 2021; Liu et al., 2022), or selective parameter updates (Ben-Zaken et al., 2022). Within this family, low-rank adaptation (LoRA) takes a weight-update reparameterization approach, learning a low-rank residual while freezing the pre-trained weights (Hu et al., 2022). This parameter efficiency does not eliminate the learning–retention tension: in mathematical and programming fine-tuning, standard low-rank LoRA generally learns less of the target domain than full fine-tuning while better preserving out-of-domain performance (Biderman et al., 2024). The rank constraint limits the dimensionality of the update but does not explicitly distinguish between pre-trained directions to adapt and those to protect (Hu et al., 2022; Xiong & Xie, 2026). We investigate how to allocate a limited-rank update across pre-trained directions to balance learning a new task with retaining existing capabilities.

A spectral component of a pre-trained weight maps an input direction to an output direction, and its singular value sets the gain of this mapping. Spectral approaches initialize adaptation from principal or minor components (Meng et al., 2024; Wang et al., 2025b), or constrain updates to protect dominant input and output subspaces (Xiong & Xie, 2026). These choices characterize pre-trained structure but do not directly measure how the target objective depends on it (Feng et al., 2026). Task-conditioned methods use early adapter updates to select directions for amplification or Fisher information to guide initialization (Si et al., 2025; Feng et al., 2026). These approaches motivate a distinction between changes learned during adaptation and the task's dependence on existing computation in the frozen model. Allocating capacity then requires distinguishing strong existing mappings

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

from weaker mappings that also support the task. The question is which task-dependent components should receive additional capacity and which should be preserved.

Our key finding is that components with high task dependence but low pre-trained spectral strength provide effective support for balancing downstream learning and held-out performance. Task dependence identifies which weak mappings the task uses, directing new capacity toward these inputs. CARD-LoRA probes this dependence with temporary suppression gates on the frozen model and combines it with spectral strength to select input directions for adaptation. Because the probe measures reliance on existing mappings rather than the correction needed for adaptation, we constrain only the adapter's input directions. After removing the gates and restoring the original model, we train a fresh LoRA adapter. Role discovery determines where adaptation acts; adapter training learns the output correction.

Following OPLoRA's cross-domain evaluation approach (Xiong & Xie, 2026), we compare CARD-LoRA with six baselines on LLaMA-2-7B under commonsense and mathematical fine-tuning. It achieves the highest combined downstream and held-out evaluation average in both settings, exceeding vanilla LoRA by 1.23 and 0.61 percentage points, respectively. Single-signal controls show that joint allocation achieves a higher combined downstream and held-out average than either strength-only or task-only selection. Restriction-side ablations favor input-only support over output-only and two-sided constraints. The support-size sweep further reveals different budgets for maximizing downstream and held-out performance, with a non-monotonic adaptation–retention trade-off.

Our contributions are:

- Joint spectral role discovery that combines supervised suppression dependence with pre-trained spectral strength to rank candidate input directions (Section 3.2).
- An input-support-constrained LoRA adapter that separates role-based capacity allocation from learning the output correction (Section 3.3).
- Empirical findings that spectral strength poorly predicts task dependence and that input-only restriction improves both downstream and held-out performance over a two-sided constraint (Section 4).

2 Related Work

Low-rank parameterization and optimization. LoRA learns a factorized residual with frozen pre-trained weights (Hu et al., 2022). Extensions refine rank allocation (Zhang et al., 2023b; Ke et al., 2024; Gu et al., 2025), update parameterization (Kopiczko et al., 2024; Liu et al., 2024; Zhang et al., 2025a), and factor optimization (Hayou et al., 2024; Wang et al., 2025c). CARD-LoRA addresses a complementary allocation question: which pre-trained input directions should support adaptation.

Spectral adaptation and subspace preservation. Spectral adaptation methods use the pre-trained spectrum to initialize or parameterize updates (Meng et al., 2024; Wang et al., 2025b; Lingam et al., 2024; Zhang & Pilanci, 2024; Wang et al., 2025a). Preservation-oriented methods use subspace-based initialization or training constraints to limit interference with pre-trained computation (Xiong & Xie, 2026; Chen & Tan, 2026; Zhang et al., 2026; Tang et al., 2026). CARD-LoRA instead interprets spectral strength jointly with suppression dependence, using component roles to decide which input directions should receive adaptation capacity.

Task-conditioned direction selection. Data-dependent methods initialize adapters using downstream information (Yang et al., 2024; Paischer et al., 2025; Wang et al., 2024; Zhang et al., 2025b; Feng et al., 2026; Si et al., 2026). Other approaches retain directions derived from activation geometry or early updates during training (Lin et al., 2026; Si et al., 2025; She et al., 2025). CARD-LoRA instead probes frozen components with suppression gates and combines this dependence with spectral strength to fix the input support of a fresh adapter.

3 METHOD

As illustrated in Figure 1, CARD-LoRA probes task dependence using temporary suppression gates and combines this dependence with spectral strength to identify the roles of pre-trained directions. It prioritizes task-dependent directions with low spectral strength and keeps the resulting input support fixed during LoRA training while leaving the output correction freely learnable. The resulting updates preserve unselected input mappings at each adapted weight, helping balance learning the new task with retaining existing capabilities.

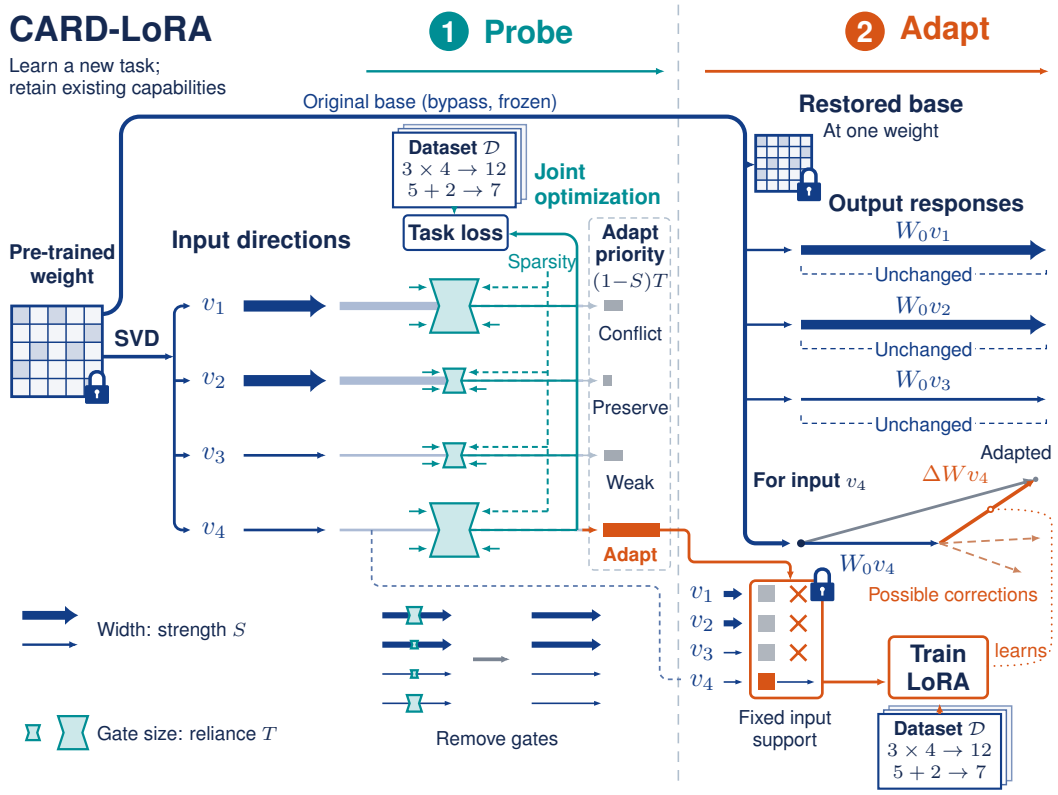


Figure 1: CARD-LoRA separates input-support discovery from learning the output correction. Arrow widths encode relative spectral strength S; gate sizes encode relative task dependence T, estimated by jointly optimizing gates shared across examples using the task loss and a sparsity penalty. The four rows illustrate roles; selection uses the continuous adapt score $(1 - S)T$ for ranking. After gate removal and base restoration, selected directions constrain only the input support of a fresh LoRA adapter. The v_4 sketch shows a learned output correction; unselected input mappings at this weight remain unchanged. Directions, priorities, and task examples are illustrative.

3.1 PROBING TASK DEPENDENCE

We first probe which pre-trained computations the downstream task relies on, without training an adapter. Let $\mathcal{D} = \{(z_n, y_n)\}_{n=1}^{N}$ denote the downstream data and $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ a frozen weight selected for adaptation. Following spectral adaptation methods (Meng et al., 2024; Wang et al., 2025b), we compute the compact singular value decomposition (SVD) of W_0:

$$W_0 = U \operatorname{diag}(\sigma) V^\top = \sum_{i=1}^{q} \sigma_i u_i v_i^\top, \qquad q = \min(d_{\text{out}}, d_{\text{in}}). \tag{1}$$

Each component maps the right singular input direction v_i to the left singular output direction u_i, with gain σ_i. This decomposition provides a common basis for measuring pre-trained strength and probing task dependence on the same components. Let $\mathcal{C} \subseteq \{1, \ldots, q\}$ index the candidate components to probe.

Sparsity-regularized gating has been used to select network components (Louizos et al., 2018); here, we use gates as temporary probes of frozen spectral components. We attach a suppression gate $\eta_i = \mathrm{sigmoid}(\rho_i)$ to each candidate component and jointly optimize only the gate parameters ρ, keeping all pre-trained weights frozen. These parameters are shared across training examples. Smaller gate values suppress a component's pre-trained contribution more strongly. For a single target weight, the gated transformation and discovery objective are

$$\widetilde{W}(\eta) = W_0 + U_{\mathcal{C}} \,\mathrm{diag}((\eta - \mathbf{1}) \odot \sigma_{\mathcal{C}})\, V_{\mathcal{C}}^\top, \tag{2}$$

$$\rho^\star = \arg\min_{\rho} \; \mathcal{L}_{\mathrm{task}}\Big(\widetilde{W}(\eta); \mathcal{D}\Big) + \frac{\lambda_{\mathrm{gate}}}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \eta_i. \tag{3}$$

Here $\mathcal{L}_{\mathrm{task}}$ is the supervised loss of the full model with the indicated weight replaced, and $\lambda_{\mathrm{gate}} > 0$ controls sparsity. The sparsity penalty encourages gates to close, while the task loss opposes suppressions that impair prediction. For multiple target weights, we jointly optimize their gates and sum the sparsity penalties, each normalized by the number of candidate components in the corresponding target weight.

After optimization, we compute the gate values $\eta_i^\star = \mathrm{sigmoid}(\rho_i^\star)$ and normalize them within each target weight to obtain relative dependence scores:

$$T_i = \frac{\eta_i^\star - \min_{j \in \mathcal{C}} \eta_j^\star}{\max_{j \in \mathcal{C}} \eta_j^\star - \min_{j \in \mathcal{C}} \eta_j^\star + \epsilon}, \qquad i \in \mathcal{C}, \tag{4}$$

where $\epsilon > 0$ stabilizes the denominator. A larger T_i indicates stronger *supervised suppression dependence*: the component retains a larger gate under the joint discovery objective.

3.2 Role-Based Capacity Allocation

Suppression dependence identifies which existing computations support the task, but does not by itself determine where additional adaptation capacity is useful. We distinguish allocation roles by pairing this dependence with the strength of each pre-trained mapping. We measure spectral strength using normalized squared singular values:

$$S_i = \frac{\sigma_i^2 - \min_{j \in \mathcal{C}} \sigma_j^2}{\max_{j \in \mathcal{C}} \sigma_j^2 - \min_{j \in \mathcal{C}} \sigma_j^2 + \epsilon}, \qquad i \in \mathcal{C}. \tag{5}$$

Min–max normalization makes S_i a measure of relative strength among the candidate components of each target weight.

High T_i together with low S_i identifies mappings that the task relies on despite their relatively small pre-trained gain. Task dependence gives these weak mappings priority for new corrections, while spectral strength steers capacity away from the strongest pre-trained mappings. High dependence combined with high strength indicates a potential conflict between adaptation and reliance on a strong existing mapping. The remaining roles distinguish strong components with little target dependence from weak components with low target priority. We express these preferences through four continuous role scores:

$$\begin{aligned} p_i^{\mathrm{adapt}} &= (1 - S_i)T_i, & p_i^{\mathrm{conflict}} &= S_i T_i, \\ p_i^{\mathrm{preserve}} &= S_i(1 - T_i), & p_i^{\mathrm{weak}} &= (1 - S_i)(1 - T_i). \end{aligned} \tag{6}$$

The four nonnegative scores sum to one for each component and describe relative roles rather than hard categories. We use only the adapt score to select the support for CARD-LoRA; the other scores define alternative allocation preferences evaluated in Section 4.4.

We select k input directions, with $1 \le k \le |\mathcal{C}|$, by ranking the adapt score:

$$\begin{aligned} \mathcal{I}_{\mathrm{cap}} &= \mathrm{TopK}_{i \in \mathcal{C}}(T_i(1 - S_i), k)\,, \\ V_{\mathrm{cap}} &= V[:, \mathcal{I}_{\mathrm{cap}}], & P_{\mathrm{cap}} &= V_{\mathrm{cap}} V_{\mathrm{cap}}^\top. \end{aligned} \tag{7}$$

Here $\mathcal{I}_{\mathrm{cap}}$ contains the selected components, V_{cap} spans their right-singular input directions, and P_{cap} is the orthogonal projector onto this support. The support budget k is shared across target weights, but ranking is performed independently for each weight. Selection uses the continuous adapt score without separate hard thresholds on S_i or T_i. Under the input constraint described next, all unselected input mappings at the target weight remain unchanged, regardless of their role scores.

3.3 INPUT-SUPPORT-CONSTRAINED LORA

After role discovery, we remove the gates and restore the exact pre-trained weight W_0. We constrain the weight update ΔW of a fresh LoRA adapter using only the selected input-support projector P_{cap}:

$$\Delta W = \Delta W P_{\text{cap}}. \tag{8}$$

With P_{cap} fixed, we apply the standard LoRA update $s\,BA$ (Hu et al., 2022) to the projected input, yielding $\Delta W = s\,BAP_{\text{cap}}$. We optimize both newly initialized LoRA factors, $A \in \mathbb{R}^{r\times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}}\times r}$, using the downstream task loss. Here, r denotes the nominal rank and $s = \alpha/r$ is the LoRA scaling factor. For an input x to the target weight, the adapted transformation is

$$y = W_0x + b + s\,BAP_{\text{cap}}x = W_0x + b + s\,BAV_{\text{cap}}V_{\text{cap}}^{\top}x, \tag{9}$$

where b is the pre-trained bias, if present.

The suppression probe identifies reliance on existing mappings without specifying the output correction needed for adaptation. The selected components' left singular vectors u_i therefore need not span the output directions required by the new task. Equation 9 retains only their right singular vectors: P_{cap} determines which inputs can activate the residual, while the trainable B learns its output space. In Figure 1, v_4 is selected for its low spectral strength and large probe gate, while its learned correction $\Delta W v_4$ is free to point in a different direction from W_0v_4. Unlike LoRA-FA, which freezes the down-projection factor (Zhang et al., 2023a), we train both LoRA factors within the fixed input support and impose no output-side projector.

The resulting update is input-local and remains low-rank:

$$P_{\text{cap}}x = 0 \implies \Delta Wx = 0, \qquad \operatorname{rank}(\Delta W) \le \min(r, k). \tag{10}$$

Thus, k controls the input-support dimension and can further limit the effective update rank at a fixed nominal rank r. For an unselected right singular vector v_j, $\Delta W v_j = 0$ and hence $(W_0 + \Delta W)v_j = W_0v_j$, preserving that input mapping at the adapted weight. Section 4.5 evaluates the input-only design.

Discovery incurs SVD and gate-optimization costs before adapter training. Appendix A provides additional local analyses and computational details.

4 EXPERIMENTS

We evaluate whether CARD-LoRA improves the balance between downstream learning and held-out performance. After reporting taskwise results, we examine this balance across methods and input-support sizes, analyze role-based allocation, and assess the effect of output flexibility. Data and training settings, role-discovery hyperparameters, and hardware specifications are summarized in Appendix C.

4.1 EXPERIMENTAL SETUP

Tasks and backbone. We use LLaMA-2-7B (Touvron et al., 2023) as the base model and fine-tune separately on Commonsense-170k and MetaMathQA (Yu et al., 2024). Commonsense adaptation is evaluated on BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA. Mathematical adaptation is evaluated on MathQA, GSM8K, MATH, SVAMP, and MAWPS. For each setting, the prior-capability group evaluates capabilities outside the targeted fine-tuning domain.

Baselines and capacity controls. We compare with LoRA, PiSSA, MiLoRA, LoRA-Null, DoRA, and OPLoRA (Hu et al., 2022; Meng et al., 2024; Wang et al., 2025b; Tang et al., 2026; Liu et al., 2024; Xiong & Xie, 2026). They cover standard low-rank adaptation, spectrum-based initialization, activation-based initialization, weight decomposition, and explicit subspace protection. The adaptation targets are $\{\texttt{q_proj}, \texttt{v_proj}, \texttt{o_proj}, \texttt{up_proj}, \texttt{down_proj}\}$. We use a shared training configuration for these comparisons. For CARD-LoRA, the suffixes 16 and 128 specify the selected input-support dimension k. For OPLoRA, these suffixes specify the number of dominant singular directions *protected* by its projection (Xiong & Xie, 2026). The same numerical suffix therefore does not imply the same available update space. The focused ablations use $k = 16$, except for the support-size sweep in Section 4.3.

Metrics and aggregation. All scores are percentages: MBPP uses pass@1, and other tasks use accuracy. In the main tables, Avg. equally weights the displayed evaluation columns; the mathematical table's commonsense 8-task aggregate counts as one column. In the role and restriction-side ablations, Down. Avg. and Prior Avg. report the downstream-task and prior-capability group averages, respectively; Overall Avg. equally weights these four group scores. Figure 3 reports these two objectives separately. Following OPLoRA (Xiong & Xie, 2026), we evaluate capability retention on held-out tasks outside the fine-tuning domain. Best and second-best results in each column are bolded and underlined, respectively.

4.2 Main Results

Commonsense reasoning. A support discovered from commonsense data can improve held-out capabilities while remaining competitive on the target tasks (Table 1). CARD-LoRA-16 and CARD-LoRA-128 achieve the two highest overall means, 51.82 and 51.71, respectively.

Table 1: Commonsense-170k fine-tuning. Avg. equally weights all displayed task columns. Best and second-best results are bolded and underlined, respectively.

	Downstream Tasks								Prior Capabilities			
Method	**BoolQ**	**PIQA**	**SIQA**	**HSwag**	**WinoG**	**ARC-e**	**ARC-c**	**OBQA**	**MathQA**	**RACE**	**MBPP**	**Avg.**
LoRA (2022)	81.73	74.55	54.42	64.02	64.33	66.67	42.68	43.24	27.60	34.51	2.76	50.59
MiLoRA (2025b)	81.39	74.81	53.70	61.52	65.01	65.67	42.25	43.27	**29.26**	**35.95**	3.73	50.60
LoRA-Null (2026)	82.25	74.12	54.61	61.05	65.96	67.84	43.70	43.60	27.33	33.05	2.53	50.55
PiSSA (2024)	77.03	74.76	**55.56**	59.78	**66.27**	**71.29**	43.60	44.73	28.06	34.67	6.13	51.08
DoRA (2024)	**82.48**	74.23	54.45	61.98	64.27	67.19	42.59	42.80	27.46	34.23	2.93	50.42
OPLoRA-16 (2026)	81.54	74.87	54.80	65.38	64.57	67.67	43.81	43.40	27.75	33.36	2.44	50.87
OPLoRA-128 (2026)	81.57	74.98	54.65	66.49	63.52	66.72	42.91	43.24	28.05	33.98	2.16	50.75
Our Methods												
CARD-LoRA-16	77.70	**75.39**	53.29	61.17	62.35	70.26	**46.72**	**45.13**	28.19	35.74	**14.05**	**51.82**
CARD-LoRA-128	80.05	75.34	53.93	**68.72**	64.17	69.72	45.95	44.16	28.58	35.10	3.08	51.71

The clearest held-out advantage is in code generation: CARD-LoRA-16 reaches 14.05 on MBPP, compared with 6.13 for the strongest baseline. The second-best RACE score, 35.74, extends this held-out advantage to reading comprehension.

Mathematical reasoning. Under mathematical fine-tuning, the combined advantage is accompanied by consistent gains across the held-out capability groups (Table 2). CARD-LoRA-128 achieves the highest overall mean, 38.29, followed by CARD-LoRA-16 at 38.01.

Table 2: MetaMathQA fine-tuning. Avg. equally weights the displayed columns; the commonsense 8-task aggregate counts as one column. Best and second-best results are bolded and underlined, respectively.

	Downstream Tasks					Prior Capabilities			
Method	**MathQA**	**GSM8K**	**MATH**	**SVAMP**	**MAWPS**	**8-task**	**RACE**	**MBPP**	**Avg.**
LoRA (2022)	23.56	53.52	5.74	60.57	46.77	52.72	37.48	21.07	37.68
MiLoRA (2025b)	23.59	53.17	5.81	60.89	45.81	52.60	39.26	22.47	37.95
LoRA-Null (2026)	24.81	53.10	5.55	**61.22**	43.35	51.72	37.48	22.07	37.41
PiSSA (2024)	24.44	51.35	5.83	57.89	47.31	52.17	38.78	14.07	36.48
DoRA (2024)	25.57	52.89	5.74	59.78	46.67	52.38	39.77	21.00	37.98
OPLoRA-16 (2026)	24.68	**53.55**	5.89	60.78	45.00	52.66	39.51	21.27	37.92
OPLoRA-128 (2026)	25.75	53.29	5.65	60.67	45.71	52.17	39.10	21.40	37.97
Our Methods									
CARD-LoRA-16	**26.33**	48.86	5.90	57.83	47.18	**53.53**	**41.33**	**23.13**	38.01
CARD-LoRA-128	25.61	51.48	**6.63**	58.95	**48.60**	52.34	40.89	21.85	**38.29**

The compact support leads across commonsense reasoning, reading comprehension, and code generation, scoring 53.53 on the commonsense suite, 41.33 on RACE, and 23.13 on MBPP. The config-

uration with the strongest held-out performance differs from that with the highest combined average, motivating a closer examination of the learning–retention balance.

Evaluation on an additional backbone. We also evaluate the methods on Qwen2.5-7B under Commonsense-170k fine-tuning (Figure 2). CARD-LoRA-128 achieves the highest combined average among the compared configurations (64.14), exceeding the strongest baseline, OPLoRA-128, by 1.10 percentage points. These results demonstrate favorable performance on an additional backbone.

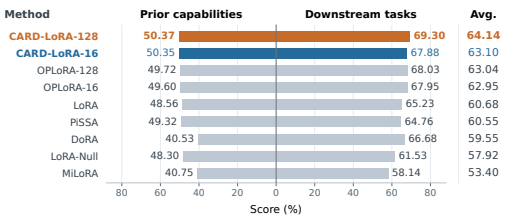


Figure 2: Commonsense fine-tuning on Qwen2.5-7B. Avg. equally weights eight downstream and three prior-capability evaluation entries.

4.3 Adaptation–Retention Trade-off

Figure 3 compares the observed learning–retention balance across methods and input-support sizes. The repeated-run protocol and numerical results are provided in Appendix D (Tables 9 and 10).

Comparison across methods. Panels (a,c) plot mean downstream-task scores on the horizontal axes and mean held-out prior-capability scores on the vertical axes. For commonsense adaptation, CARD-LoRA at $k = 32$ achieves mean downstream and prior-capability scores of 62.22 and 23.38, respectively, exceeding all plotted baselines on both mean metrics. This configuration offers a favorable observed balance between downstream learning and held-out performance.



Figure 3: Adaptation–retention comparison and support-size sensitivity. Rows show commonsense (top) and mathematical (bottom) fine-tuning. **(a,c):** method comparisons with vertical prior-score error bars. **(b,d):** downstream and prior scores versus k, shown separately. Points denote means; bars and shaded bands indicate ± 1 SD across repeated runs. Axes use independent scales.

Effect of input-support size. We vary $k \in \{16, 32, 48, 64, 80, 96, 112, 128\}$ in both domains (Figure 3(b,d)). Among the tested sizes, $k = 128$ yields the highest mean downstream scores, 62.76 for commonsense and 38.25 for mathematical reasoning, whereas $k = 16$ yields the highest mean prior-capability scores, 25.99 and 39.33, respectively. The intermediate results are non-monotonic, so larger supports do not uniformly improve either objective.

The support size controls the number of input directions available to the update, with $\mathrm{rank}(\Delta W) \leq \min(r, k)$ (Eq. 10). With the nominal rank fixed at $r = 8$, all tested sizes satisfy $k \geq r$; the sweep therefore varies the available input support without increasing the rank ceiling.

4.4 Role Discovery and Allocation Analysis

Spectral strength and task dependence. Spectral strength and task dependence exhibit near-zero median per-weight rank correlations in both domains (Figure 4(a)), indicating that strength rankings do not closely track dependence rankings. The cross-domain correlation of task-dependence scores is also low. Joint supports differ across domains but overlap more than expected under independent uniform selection (Figure 4(c)), whereas strength-only supports are task-invariant by construction.

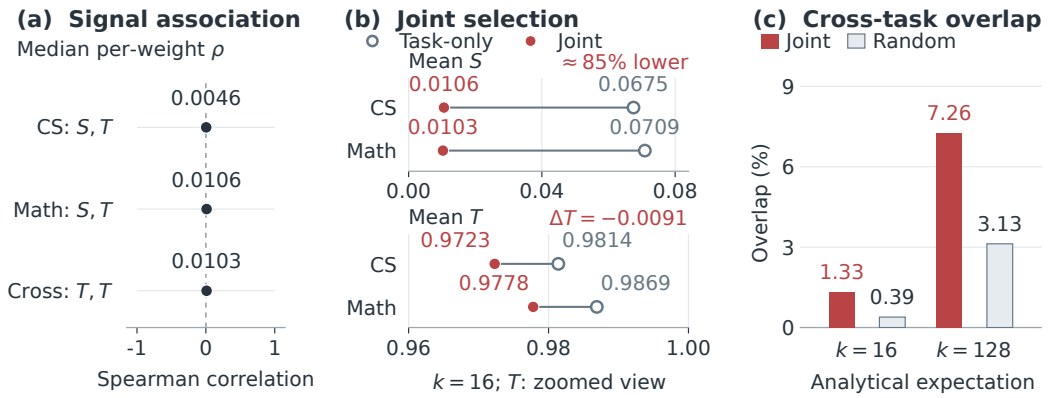


Figure 4: Spectral signals and support selection. **(a)** Median per-weight full-spectrum Spearman correlations: within-domain S–T and cross-domain T–T. **(b)** Mean S and T of task-only and joint supports at $k = 16$; axes use independent scales, with T zoomed to $[0.96, 1.00]$. **(c)** Cross-task joint-support overlap and its analytical reference, $100k/4096\%$, for independent uniform k-subsets of 4,096 directions. Overlap is the intersection size divided by k, averaged across target weights and expressed as a percentage. S and T are normalized to $[0, 1]$ within each target weight. CS and Math denote commonsense and mathematical fine-tuning.

Joint support selection. At $k = 16$, we compare strength-only $(1 - S_i)$, task-only (T_i), and joint $((1 - S_i)T_i)$ rankings. Joint selection yields higher mean task dependence than strength-only selection in both domains. Relative to task-only selection, it reduces mean spectral strength by approximately 85% while decreasing mean task dependence by approximately 0.009 in both domains (Figure 4(b)). High task dependence is therefore not confined to strong pre-trained components. Figure 5 illustrates the resulting supports: $k = 16$ favors low-strength, high-dependence directions, whereas $k = 128$ also admits stronger or less task-dependent directions. Numerical summaries of the selected-direction scores and support overlaps are provided in Appendix B (Tables 5 and 6).

Allocation effectiveness. At $k = 16$, we compare joint selection (adapt-only) with single-signal baselines and alternative roles (Table 3). Adapt-only achieves the highest Overall Avg. and exceeds task-only in all four groups. Relative to strength-only, it improves both commonsense scores and mathematical downstream performance, with a slightly lower mathematical prior score (39.33 versus 39.54). These results support combining spectral strength and task dependence for allocation.

Table 3: Support selection at $k = 16$. Overall Avg. equally weights the four group averages.

	Commonsense		Math		
Support variant	Down. Avg.	Prior Avg.	Down. Avg.	Prior Avg.	Overall Avg.
Strength-only	60.06	23.18	35.32	**39.54**	39.53
Task-only	61.14	21.29	36.93	37.91	39.32
Weak only	**61.88**	21.32	37.03	36.65	39.22
Conflict only	59.48	21.15	34.38	35.33	37.59
Preserve only	60.51	20.30	36.84	30.77	37.11
Our Method					
Adapt only	61.50	**25.99**	**37.22**	39.33	**41.01**

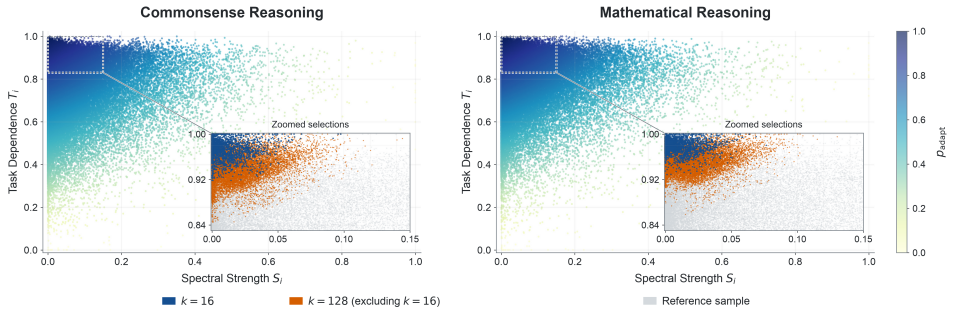


Figure 5: Input-support selection for commonsense (left) and mathematical (right) fine-tuning. Main panels: fixed random sample colored by adaptation score. Insets: $k = 16$ (blue), additions at $k = 128$ (orange), and the same sample (grey, not unselected-only). Boxes mark zooms, not thresholds; ranking uses all candidates per weight.

Among low-strength directions, adapt-only favors high task dependence, whereas weak-only favors low dependence. Weak-only achieves slightly higher commonsense downstream accuracy (61.88 versus 61.50) but lower prior-capability performance (21.32 versus 25.99); prioritizing dependence yields better held-out performance at similar downstream accuracy.

4.5 Input-Side Restriction and Output Flexibility

We compare input-only, output-only, and two-sided restrictions at $k = 16$. The input-only variant is CARD-LoRA; the other variants constrain the output space, either alone or together with the input space. Comparing input-only and two-sided restrictions tests whether adding an output constraint to the selected input support helps preserve held-out capabilities.

Table 4: Input-only restriction (CARD-LoRA) improves both downstream and held-out scores over two-sided restriction ($k = 16$). Overall Avg. equally weights the four group averages.

	Commonsense		Math		
Restriction	Down. Avg.	Prior Avg.	Down. Avg.	Prior Avg.	Overall Avg.
Output-only	<u>52.35</u>	<u>24.51</u>	<u>30.12</u>	29.80	<u>34.20</u>
Two-sided	44.22	19.38	29.37	<u>32.54</u>	31.38
Our Method					
Input-only	**61.50**	**25.99**	**37.22**	**39.33**	**41.01**

Input-only achieves the highest group average in each of the four groups in Table 4. Compared with the two-sided restriction, it achieves downstream scores of 61.50 versus 44.22 under commonsense adaptation and 37.22 versus 29.37 under mathematical adaptation. Its prior-capability scores are also higher in both settings. The additional output constraint therefore worsens both objectives in these experiments rather than improving retention at the expense of learning. These results support separating the choice of where to adapt from the correction to be learned: discovered spectral components guide the input support, while the output correction remains freely learnable.

5 Conclusion and Limitations

Low-rank adaptation benefits from separating where capacity is allocated from what output correction is learned. CARD-LoRA combines spectral strength and task dependence to identify input directions for adaptation while leaving the correction learnable. Across commonsense and mathematical fine-tuning, high-dependence, low-strength supports provide a favorable learning–retention balance, and input-only restriction improves both objectives over a two-sided constraint. The support-size sweep further identifies the extent of input support as a control of this balance. Together, these findings place the location and extent of adaptation alongside update rank as central choices in allocating limited capacity.

Limitations. CARD-LoRA incurs additional discovery cost from SVD and supervised gate optimization before adapter training. The current design also uses a shared support budget k across target modules, without optimizing module-specific support sizes. Reducing discovery overhead and exploring adaptive support-budget allocation are directions for future work.

AI USE STATEMENT

We used generative AI tools primarily to assist with manuscript formatting and language polishing. This assistance focused on improving typographic consistency, the presentation of experimental tables, and the clarity, fluency, and conciseness of the text.

The authors reviewed and revised the AI-assisted material, checked numerical values against the experimental records, and ensured that the final wording accurately reflected the methods, results, and intended scientific claims. The authors retain full responsibility for the accuracy and integrity of the final manuscript.

REFERENCES

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. LoRA learns less and forgets less. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=aloEru2qCG.

Lixian Chen and Jianhong Tan. HiP-LoRA: Budgeted spectral plasticity for robust low-rank adaptation. *arXiv preprint arXiv:2604.17751*, 2026. URL https://arxiv.org/abs/2604.17751.

Zhi-Quan Feng, Ying-Jia Lin, and Hung-Yu Kao. Learning in the fisher subspace: A guided initialization for LoRA fine-tuning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=P76Q8Pi1E9.

Naibin Gu, Zhenyu Zhang, Xiyu Liu, Peng Fu, Zheng Lin, Shuohuan Wang, Yu Sun, Hua Wu, Weiping Wang, and Haifeng Wang. BeamLoRA: Beam-constraint low-rank adaptation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11871–11883, 2025.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *International Conference on Machine Learning*, pp. 17783–17806, 2024.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 2790–2799, 2019.

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

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Wenjun Ke, Jiahao Wang, Peng Wang, Jiajun Liu, Dong Nie, Guozheng Li, and Yining Li. Unveiling LoRA intrinsic ranks via salience analysis. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024.

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

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Weicheng Lin, Yi Zhang, Jiawei Dang, and Liang-Jie Zhang. TLoRA: Task-aware low rank adaptation of large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29252–29269. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1348. URL https://aclanthology.org/2026.acl-long.1348/.

Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. SVFT: Parameter-efficient fine-tuning with singular vectors. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2024.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 61–68, 2022.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through L_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Fabian Paischer, Lukas Hauzenberger, Thomas Schmied, Benedikt Alkin, Marc Deisenroth, and Sepp Hochreiter. Parameter efficient fine-tuning via explained variance adaptation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Yifei She, Xinhao Wei, and Yulong Wang. DisLoRA: Task-specific low-rank adaptation via orthogonal basis from singular value decomposition. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13740–13755, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.694. URL https://aclanthology.org/2025.emnlp-main.694/.

Chongjie Si, Zhiyi Shi, Shifan Zhang, Xiaokang Yang, Hanspeter Pfister, and Wei Shen. Unleashing the power of task-specific directions in parameter efficient fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=RYrJqz44p4.

Chongjie Si, Zhiyi Shi, Shifan Zhang, Xiaokang Yang, Hanspeter Pfister, and Wei Shen. Task-specific directions: Definition, exploration, and utilization in parameter efficient fine-tuning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(6):6754–6770, 2026. doi: 10.1109/TPAMI.2026.3659168. URL https://doi.org/10.1109/TPAMI.2026.3659168.

Pengwei Tang, Xiaolin Hu, Yong Liu, Lizhong Ding, Dongjie Zhang, Xing Wu, and Debing Zhang. Put the space of LoRA initialization to the extreme to preserve pre-trained knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33232–33240, 2026. doi: 10.1609/aaai.v40i39.40608. URL https://ojs.aaai.org/index.php/AAAI/article/view/40608.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Fan Wang, Juyong Jiang, Chansung Park, Sunghun Kim, and Jing Tang. KaSA: Knowledge-aware singular-value adaptation of large language models. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=OQqNieeivq.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. MiLoRA: Harnessing minor singular components for parameter-efficient LLM finetuning. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4823–4836, 2025b.

Shaowen Wang, Linxi Yu, and Jian Li. LoRA-GA: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 54905–54931, 2024. doi: 10.52202/079017-1741.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. LoRA-Pro: Are low-rank adapters properly optimized? In *International Conference on Learning Representations*, 2025c. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/ea184f920a0f0f8d8030aa1bd7ac9fd4-Abstract-Conference.html.

Yifeng Xiong and Xiaohui Xie. OPLoRA: Orthogonal projection LoRA prevents catastrophic forgetting during parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 34088–34096, 2026. doi: 10.1609/aaai.v40i40.40703. URL https://ojs.aaai.org/index.php/AAAI/article/view/40703.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Leon Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. CorDA: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, 2024.

Fangzhao Zhang and Mert Pilanci. Spectral adapter: Fine-tuning in spectral space. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Geyuan Zhang, Xiaofei Zhou, Shihao Liu, Jingyuan Tian, and Jizheng Ma. SDC-LoRA: Singular-subspace drift controlled LoRA to mitigate knowledge forgetting. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24116–24132. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1207. URL https://aclanthology.org/2026.findings-acl.1207/.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning. *arXiv preprint arXiv:2308.03303*, 2023a. URL https://arxiv.org/abs/2308.03303v1.

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

Xin Zhang, Guang-Ze Chen, Shuzhen Li, Zhulin Liu, C. L. Philip Chen, and Tong Zhang. An orthogonal high-rank adaptation for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 18815–18833, 2025a.

Yuanhe Zhang, Fanghui Liu, and Yudong Chen. LoRA-One: One-step full gradient could suffice for fine-tuning large language models, provably and efficiently. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 75513–75574. PMLR, 2025b. URL https://proceedings.mlr.press/v267/zhang25ax.html.

A Additional Method Details

A.1 Suppression Dependence and Adaptation-Score Ordering

A.1.1 Local Interpretation of Suppression Gates

The gates in Equation 2 have limiting values $\eta_i = 1$ and $\eta_i = 0$, corresponding to retaining and fully suppressing a component, respectively. Finite sigmoid parameters yield gates strictly between these limits.

Proposition 1 (Gate monotonicity under a separable surrogate). Assume a local model of the suppression loss with zero linear term and positive diagonal curvature:

$$\mathcal{L}_{\text{task}}(\eta) \simeq \mathcal{L}_{\text{task}}(\mathbf{1}) + \frac{1}{2}\sum_{i\in\mathcal{C}} h_i(1-\eta_i)^2, \qquad h_i > 0. \tag{11}$$

Its relaxed optimum is

$$\eta_i^\star = \max\left\{0,\, 1 - \frac{\mu}{h_i}\right\}, \qquad \mu = \lambda_{\text{gate}}/|\mathcal{C}|, \tag{12}$$

Hence, the relaxed gate is non-decreasing in the local quadratic suppression sensitivity h_i. The optimization here is over the closed interval $[0, 1]$; a boundary value may only be approached by the sigmoid parameterization. This result explains one setting in which gates rank suppression sensitivity. It is not a general property of the nonconvex discovery objective, whose linear terms and cross-component interactions need not vanish. The derivation follows.

Under the local surrogate in Eq. 11, the relaxed gate objective, up to constants independent of η, is separable:

$$\sum_{i\in\mathcal{C}} \left[\frac{1}{2}h_i(1-\eta_i)^2 + \mu\eta_i\right], \qquad 0 \le \eta_i \le 1. \tag{13}$$

For one component, differentiating the bracketed term gives

$$\frac{\partial}{\partial\eta_i}\left[\frac{1}{2}h_i(1-\eta_i)^2 + \mu\eta_i\right] = -h_i(1-\eta_i) + \mu. \tag{14}$$

The unconstrained stationary point is $1-\mu/h_i$. Since $h_i > 0$, the objective is convex in η_i; projection of this stationary point onto $[0, 1]$ therefore gives Eq. 12. On the interval where it is not clipped, its derivative with respect to h_i is $\mu/h_i^2 > 0$. Clipping preserves non-decreasing order, proving the claim. □

A.1.2 Module-Specific Selection Cutoffs

The selection in Equation 7 does not impose separate hard thresholds on S_i or T_i, nor does it use a global threshold on p_i^{adapt}. For each target module ℓ with candidate set $\mathcal{C}_\ell$, let

$$p^{\text{adapt}}_{\ell,(1)} \ge \cdots \ge p^{\text{adapt}}_{\ell,(|\mathcal{C}_\ell|)}$$

denote its adaptation scores in descending order. The support budget k induces the module-specific cutoff

$$\tau_\ell(k) = p^{\text{adapt}}_{\ell,(k)}.$$

Thus, k is shared across target modules, whereas the numerical cutoff depends on each module's score distribution.

A.1.3 Adaptation-score dominance

Let $a_i = T_i(1 - S_i)$ denote the adaptation score. Consider two non-degenerate directions i, j with $T_i, T_j \in (0, 1]$ and $S_i, S_j \in [0, 1)$. If $T_i \ge T_j$ and $S_i \le S_j$, then $1 - S_i \ge 1 - S_j > 0$, hence

$$a_i = T_i(1-S_i) \ge T_j(1-S_i) \ge T_j(1-S_j) = a_j. \tag{15}$$

If either of the two premise inequalities is strict, one of the displayed inequalities is strict as well. Therefore, top-k ranking by a_i never places a Pareto-dominated non-degenerate direction ahead of the direction that dominates it. The result only characterizes the consistency of this scalarization; it does not establish that a_i is an optimal score for every downstream task. □

A.2 Local Analysis of Input and Output Constraints

Let $X \in \mathbb{R}^{d_{\text{in}} \times n}$ contain a fixed batch of layer inputs, $R \in \mathbb{R}^{d_{\text{out}} \times n}$ be a desired local correction, and $P = P_{\text{cap}}$. Consider

$$\mathcal{L}_X(\Delta W) = \frac{1}{2}\|\Delta W X - R\|_F^2. \tag{16}$$

Theorem 1 (Data-conditioned input sufficiency). If $X = PX$, every rank-r update has an input-local counterpart $\Delta W P$ with identical outputs on X and therefore the same value of Eq. 16. More generally,

$$\|\Delta W X - \Delta W P X\|_F \le \|\Delta W\|_2 \, \|(I - P)X\|_F, \tag{17}$$

so the discrepancy is controlled by uncaptured activation energy. By contrast, adding a fixed orthogonal output projector $Q \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$ introduces the irreducible term

$$\min_{\text{rank}(C) \le r} \|QCPX - R\|_F^2 \ge \|(I - Q)R\|_F^2, \tag{18}$$

whenever the desired correction R extends outside $\text{range}(Q)$. These statements concern a fixed batch and a local linear surrogate. The role score does not guarantee $X = PX$, and the theorem therefore does not prove that the discovered support is sufficient for a downstream task. It identifies what input coverage would make the constraint lossless and why an additional output constraint can remove otherwise useful corrections. The proof follows; the output-space bound is derived in Appendix A.3.

Proof of Theorem 1. Let $P = P_{\text{cap}}$. Because V_{cap} has orthonormal columns, P is an orthogonal projector and $P^2 = P$. For an arbitrary rank-r update ΔW, define

$$\widetilde{\Delta W} = \Delta W P. \tag{19}$$

The update is input-local because

$$\widetilde{\Delta W} P = \Delta W P^2 = \Delta W P = \widetilde{\Delta W}. \tag{20}$$

Moreover, right multiplication cannot increase rank, so $\text{rank}(\widetilde{\Delta W}) \le \text{rank}(\Delta W) \le r$. If $X = PX$, then

$$\widetilde{\Delta W} X = \Delta W P X = \Delta W X. \tag{21}$$

Thus every unconstrained rank-r update has an input-local counterpart with exactly the same batch predictions and the same value of Eq. 16. Conversely, every input-local rank-r update is a valid unconstrained rank-r update. The two optimization problems consequently have equal optima.

To verify Eq. 17, decompose $X = PX + (I - P)X$. Then

$$\Delta W X - \Delta W P X = \Delta W (I - P) X. \tag{22}$$

The standard spectral–Frobenius norm inequality yields

$$\|\Delta W (I - P) X\|_F \le \|\Delta W\|_2 \|(I - P)X\|_F, \tag{23}$$

which proves the approximation bound. □

A.3 Output-Side Restriction Lower Bound

Let Q be an orthogonal projector. For any C, the fitted residual $QCPX$ lies in $\text{range}(Q)$. Decompose the desired residual as $R = QR + (I - Q)R$. Orthogonality of the two projector ranges gives

$$\|QCPX - R\|_F^2 = \|Q(CPX - R) - (I - Q)R\|_F^2 \tag{24}$$

$$= \|Q(CPX - R)\|_F^2 + \|(I - Q)R\|_F^2 \tag{25}$$

$$\ge \|(I - Q)R\|_F^2. \tag{26}$$

The inequality holds for every C, including every rank-r matrix. Taking the minimum over the rank-constrained set proves Eq. 18. This lower bound vanishes only when the desired residual already lies entirely in $\text{range}(Q)$; that is an additional output-space assumption, not one supplied by CARD-LoRA's input-side role discovery.

A.4 Effective Rank and Equivalent Core Parameterization

The right-sided projector also makes the effective capacity of the update explicit. Define $A_{\text{core}} = AV_{\text{cap}} \in \mathbb{R}^{r\times k}$. Then

$$\Delta W = sBAV_{\text{cap}}V_{\text{cap}}^{\top} = sBA_{\text{core}}V_{\text{cap}}^{\top}. \tag{27}$$

Applying the rank inequality to this product gives

$$\operatorname{rank}(\Delta W) \le \min\{\operatorname{rank}(B), \operatorname{rank}(A_{\text{core}}), \operatorname{rank}(V_{\text{cap}}^{\top})\} \le \min(r, k). \tag{28}$$

Thus the support size k constrains the maximum realized update rank when $k < r$.

Conversely, every reduced-core matrix $A_{\text{core}} \in \mathbb{R}^{r\times k}$ can be represented by a full factor: choosing $A = A_{\text{core}}V_{\text{cap}}^{\top}$ yields $AV_{\text{cap}} = A_{\text{core}}$, because $V_{\text{cap}}^{\top}V_{\text{cap}} = I_k$. The full and reduced-core forms therefore realize the same set of update matrices for a fixed V_{cap}, but their nominal parameterizations differ. The full form contains $rd_{\text{in}} + rd_{\text{out}}$ scalar parameters in A, B, whereas the reduced-core form contains $rk + rd_{\text{out}}$. We use the full form in the main method to retain the standard LoRA parameterization, and distinguish the nominal parameter count from the dimensions of the functionally active core in Eq. 27.

Computation and storage. The projector need not be materialized as a dense $d_{\text{in}} \times d_{\text{in}}$ matrix. Compute $V_{\text{cap}}^{\top}x$, map back with V_{cap}, and apply the trainable factors; this is equivalent to using the factorization in Equation 27. The fixed support basis requires storage for $d_{\text{in}}k$ coefficients in addition to the trainable factors. SVD and gate optimization incur additional discovery cost. Thus, matching nominal LoRA rank does not, by itself, match preprocessing cost or effective capacity.

B Additional Role-Discovery Statistics

Analysis scope. The offline structural analysis in this appendix and Figures 4 and 5 uses one full-spectrum LLaMA-2 discovery outcome per fine-tuning domain. The identifiers are `asymmetric_rotated_role` for Commonsense-170k and `asymmetric_rotated_role_metamathqa` for MetaMathQA. Each covers 160 target weights with 4,096 directions per weight, or 655,360 weight–component pairs. Statistics summarize components and target weights within these outcomes, rather than averaging across discovery seeds. This analysis is separate from the five-run performance evaluation in Appendix D.

Score definition and aggregation. We use the score definitions in Equations 5 and 4: S_i is the min–max-normalized squared singular value σ_i^2, and T_i is the min–max-normalized final sigmoid gate $\eta_i^{\star}$. Both normalizations are performed independently within each target weight over its full candidate set, with the stabilizing ϵ in the denominators. For this offline analysis, scores are computed in float32 directly from the raw singular values and final gate values of the specified discovery outcomes, rather than from pre-normalized cache fields. Joint selection then ranks the per-direction score $(1 - S_i)T_i$ and applies Top-k independently within each weight, as in Equation 7. Spearman correlation and linear R^2 are computed over the full spectrum within each weight and then summarized by their median across weights. Selected-score means and support overlaps are first computed within each weight and then averaged across weights. Each task contributes one distinct archived outcome, so no discovery-seed variance or between-discovery uncertainty is estimated. These statistics characterize offline score-based selections for the specified discovery outcomes; they do not constitute checkpoint-level verification of the supports used in the performance runs. No adapter is retrained for this analysis.

Selected-direction scores. Strength-only, task-only, and joint selection rank $1 - S_i$, T_i, and $(1 - S_i)T_i$, respectively. Selection uses all 4,096 candidates independently within each weight; exact score ties favor the smaller original component ID. Table 5 reports selected-direction means at $k = 16$ and $k = 128$. At $k = 16$, joint selection reduces mean S_i relative to task-only selection by 84.2% for commonsense and 85.5% for mathematics. The corresponding decreases in mean T_i are 0.009022 and 0.009094, respectively. The adaptation score is calculated per direction before averaging, rather than reconstructed from the separate means of S_i and T_i.

Table 5: Selected-direction score summaries for full-spectrum LLaMA-2. Scores are averaged within each target weight and then across weights; $p_i^{\mathrm{adapt}} = (1 - S_i)T_i$. These are within-outcome statistics, not cross-seed averages.

Domain	Selection	k	Mean S_i	Mean T_i	Mean p_i^{adapt}
Commonsense	Strength-only	16	0.001381	0.7289	0.7279
	Task-only	16	0.067456	0.981351	0.915115
	Joint	16	0.010639	0.972329	0.961983
	Strength-only	128	0.004187	0.748015	0.744873
	Task-only	128	0.069346	0.956980	0.890532
	Joint	128	0.015577	0.946307	0.931538
Mathematical	Strength-only	16	0.001381	0.7578	0.7567
	Task-only	16	0.070864	0.986875	0.916901
	Joint	16	0.010284	0.977781	0.967726
	Strength-only	128	0.004187	0.778845	0.775537
	Task-only	128	0.072566	0.966980	0.896783
	Joint	128	0.015403	0.955812	0.941062

Signal associations and cross-task pairing. The within-task median per-weight Spearman correlations are 0.004634 for commonsense and 0.010634 for mathematics; the corresponding median linear R^2 values are 0.000275 and 0.000340. Cross-task statistics use the single compatible pair of full-spectrum archives identified above, matched by model, candidate universe, target weight, singular values, and component index. This is a pairing of archived outcomes, not a pairing by recorded discovery seed. The median per-weight correlation between the two tasks' dependence scores is 0.010345. Table 6 reports within-task and cross-task support overlaps. The joint cross-task overlaps exceed the independent uniform-selection reference but do not establish repeatability across discovery seeds.

Table 6: Support overlap (%) for the same archived outcomes. Each entry is $100|\mathcal{I}_1 \cap \mathcal{I}_2|/k$, averaged across target weights. Cross-task rows use one compatible pair of archives. The uniform-random reference is the analytical expectation $100k/4096$, not a measured discovery result.

Scope	Comparison	$k = 16$	$k = 128$
Commonsense	Strength-only vs. task-only	0.27	3.33
Mathematical	Strength-only vs. task-only	0.39	2.50
Commonsense	Joint vs. strength-only	2.58	9.98
Mathematical	Joint vs. strength-only	2.19	8.79
Commonsense	Joint vs. task-only	47.93	57.92
Mathematical	Joint vs. task-only	42.15	54.38
Cross-task	Strength-only	100.00	100.00
Cross-task	Task-only	0.35	3.15
Cross-task	Joint	1.33	7.26
Reference	Uniform random	0.39	3.13

Scatter provenance and sampling. Figure 5 visualizes the offline structural analysis of these two named full-spectrum discovery outcomes, using the within-weight min–max score definitions above. For display, 60,000 of the 655,360 weight–component pairs in each task are sampled without replacement using `numpy.random.default_rng(0)`. The plotting seed is not a discovery seed. Top-k ranking uses all candidates within each weight, not the plotting sample, and all selected points are passed to the plot. The reference-data configuration takes the first up-to-2,048 training rows per task; the caches do not retain the actual number of unique examples seen during gate optimization. This cap must therefore not be interpreted as a measured number of unique discovery examples.

C Experimental Configuration

C.1 Datasets and Supervision

Backbone and data. We fine-tune LLaMA-2-7B separately on Commonsense-170k and a 100,000-example training subset of MetaMathQA-395K. The target modules are `q_proj`, `v_proj`, `o_proj`, `up_proj`, and `down_proj` in all 32 layers, giving 160 target weights. Table 7 summarizes the data sizes and adapter-training schedules. The validation counts refer to the fine-tuning data, not to the held-out evaluation benchmarks.

Table 7: Data and adapter-training schedules. Effective batch size counts examples per optimizer update; step counts exclude role discovery.

Setting	Commonsense	Mathematical
Training examples	165,420	100,000
Validation examples	5,000	5,000
Adapter training steps	2,585	12,500
Effective batch size	64	8
Maximum sequence length	192	512

Sequence construction and loss. Commonsense training uses the instruction as the prompt and the answer as the target; mathematical training uses the query in a question-and-answer format and the response as the target. An end-of-sequence token is appended to the response, and sequences are right-padded to the configured length. Prompt and response fields are tokenized without adding special tokens separately to each field. When truncation is needed, the prompt is truncated first while reserving at least 16 response-token positions. Training minimizes response-token causal language-model loss, with prompt and padding positions masked. The sequence-length limits in Table 7 are training limits, not evaluation-generation budgets.

C.2 Training Parameters and Baselines

We compare CARD-LoRA with LoRA, PiSSA, MiLoRA, LoRA-Null, DoRA, and OPLoRA. All methods use nominal rank $r = 8$, scaling parameter $\alpha = 16$, AdamW with learning rate 2×10^{-4}, and the same domain-specific training-step counts in Table 7. LoRA and OPLoRA also share the remaining adapter settings listed for CARD-LoRA in Table 8, as well as the effective batch sizes and sequence lengths in Table 7.

Table 8: Shared adapter parameters and CARD-LoRA-specific settings. The discovery optimizer applies only to temporary gate logits.

Setting	Value
Shared across all compared methods	
Nominal rank r; scaling parameter α	8; 16
Adapter optimizer; learning rate	AdamW; 2×10^{-4}
Additional adapter settings for CARD-LoRA, LoRA, and OPLoRA	
LoRA dropout; weight decay	0.05; 0
Learning-rate schedule	Linear decay with 3% warmup
CARD-LoRA precision and discovery settings	
Adapter-training precision	bfloat16
Candidate directions per target weight	4,096
SVD precision	float32
Initial gate logit	2.0
Gate optimizer; learning rate	Adam; 0.05
Gate sparsity coefficient λ_{gate}	0.01

Method-specific mechanisms. The baselines retain their distinct adaptation mechanisms: LoRA supplies the standard low-rank update; PiSSA and MiLoRA use principal and minor spectral initialization, respectively; LoRA-Null provides activation-based initialization; DoRA uses weight decomposition; and OPLoRA protects dominant subspaces. For OPLoRA, suffixes 16 and 128 indicate protected singular directions; for CARD-LoRA, they indicate selected input-support dimensions. Shared rank and adapter-training steps do not imply equal parameter counts, effective update spaces, or total preprocessing cost.

C.3 Role Discovery and Controlled Comparisons

Discovery and adapter initialization. For CARD-LoRA, we retain all 4,096 compact-SVD components of every target weight, giving 655,360 candidates per discovery run. Only the temporary gate logits are optimized during discovery; the pre-trained weights remain frozen. As in the main method, spectral strength S_i is obtained by min–max normalization of σ_i^2, and task dependence T_i by min–max normalization of the final sigmoid gate $\eta_i^\star$, independently within each target weight over all candidates. The denominators include the stabilizing ϵ specified in Equations 5 and 4. We form the joint score $(1 - S_i)T_i$ from these normalized quantities and select the Top-k directions within each weight, following Equation 7. After selection, we remove the gates, restore the pre-trained weights, and train a fresh adapter with both full-sized factors A and B learnable. The selected right-singular basis is fixed, and the gate penalty is not added to the adapter-training loss. The offline structural analysis applies this same score definition to the discovery outcomes specified in Appendix B; its analysis scope is distinct from the repeated-run performance evaluation.

Allocation and restriction controls. Allocation ablations use $k = 16$ and independently rank candidates within each weight. Strength-only and task-only use $1-S_i$ and T_i; adapt-only, weak-only, conflict-only, and preserve-only use the role scores in Equation 6. Adapt-only is the joint selection used by CARD-LoRA. Restriction-side ablations compare input-only, output-only, and two-sided constraints at the same support size. The support sweep uses $k \in \{16, 32, 48, 64, 80, 96, 112, 128\}$ with $r = 8$ fixed. All tested sizes satisfy $k \geq r$, so the sweep changes the available input support without increasing the rank ceiling.

C.4 Evaluation, Repeated Runs, and Resources

Evaluation groups. The commonsense downstream suite contains BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA; its prior group contains MathQA, RACE, and MBPP. The mathematical downstream suite contains MathQA, GSM8K, MATH, SVAMP, and MAWPS; its prior group contains the commonsense 8-task aggregate, RACE, and MBPP. The latter 8-task score is evaluated on the mathematically adapted model. MBPP uses pass@1, and the other tasks use accuracy; all scores are percentages.

Aggregation and performance repeats. Group scores equally weight their evaluation entries. Main-table averages equally weight the displayed columns, treating the mathematical table's 8-task aggregate as one column. The allocation and restriction-side tables instead equally weight their four domain–group averages. The two mean–SD tables in Appendix D use five independent runs. Each CARD-LoRA replicate reruns discovery and trains a fresh adapter; within a support-sweep replicate, all k values share the task-specific discovery outcome but use separate fresh adapters. Group scores are computed within each run before the cross-run mean and sample SD, whose variance denominator is $R - 1$. These performance repetitions are distinct from the single archived outcome per task used for the structural analysis in Appendix B.

Hardware and code. Experiments use a server with two NVIDIA A800 GPUs, each with 80 GB of memory. CARD-LoRA uses bfloat16 for adapter training and float32 for SVD. Its preprocessing includes spectral decomposition and task-conditioned gate optimization in addition to the adapter updates. An anonymized implementation is available at https://anonymous.4open.science/r/rotating_lora_research-8648/.

D NUMERICAL RESULTS FOR ADAPTATION–RETENTION ANALYSIS

This section provides the numerical results underlying Figure 3 and explains their aggregation and interpretation. Table 9 compares adaptation methods, while Table 10 isolates the input-support budget within CARD-LoRA. Both tables report downstream learning and held-out prior-capability performance separately, rather than combining them into a single objective.

D.1 REPEATED RUNS AND STATISTICAL AGGREGATION

Repeated runs. Each configuration in Tables 9 and 10 is evaluated over five independent runs. For CARD-LoRA, each replicate reruns role discovery and trains a fresh adapter using the resulting support. Within each replicate of the support-size sweep, the same task-specific discovery results are reused across all tested values of k, with a separate adapter trained for each support size. The reported SDs summarize variation across the discovery-and-adapter-training pipeline; they do not separately estimate discovery-only uncertainty. The structural statistics in Appendix B are computed from one discovery outcome per task and are not averages over these five performance runs.

Reported statistics. For each run, we first compute each evaluation-group score using the aggregation weights defined in the corresponding experiment. We then report the mean and sample standard deviation across these run-level group scores, using $R - 1$ as the variance denominator, where R is the number of runs for the corresponding configuration. Scores are expressed in percent and SDs in percentage points. Tables 9 and 10 report both downstream and prior-capability statistics; the method-comparison panels in Figure 3 display only prior-capability SDs as vertical error bars.

Run-level group scores. For a fixed method and fine-tuning configuration, let $a_t^{(b)}$ denote the percentage score on evaluation entry t in run b. For evaluation group $\mathcal{G}$, its run-level score is

$$g_{\mathcal{G}}^{(b)} = \sum_{t \in \mathcal{G}} w_t a_t^{(b)}, \qquad \sum_{t \in \mathcal{G}} w_t = 1, \tag{29}$$

where $w_t = 1/|\mathcal{G}|$ for the equally weighted entries used here. The commonsense downstream group has eight entries and its prior group has three; the mathematical downstream group has five and its prior group has three. In the mathematical prior group, the eight-task commonsense aggregate is one entry, alongside RACE and MBPP. It is computed within the same run using the mathematically adapted model. Across the $R = 5$ runs, the table entries are

$$\bar{g}_{\mathcal{G}} = \frac{1}{R} \sum_{b=1}^{R} g_{\mathcal{G}}^{(b)}, \qquad s_{\mathcal{G}} = \sqrt{\frac{1}{R-1} \sum_{b=1}^{R} \left(g_{\mathcal{G}}^{(b)} - \bar{g}_{\mathcal{G}} \right)^2}. \tag{30}$$

Thus, an SD measures variation of a group score across runs, not variation between the tasks within a group. It is not obtained by averaging task-specific SDs.

D.2 COMPARISON ACROSS ADAPTATION METHODS

Table 9 retains both group means and their SDs, including the downstream SDs omitted from the vertical error bars in Figure 3(a,c). Comparisons below refer to observed group means.

Table 9: Method comparisons: mean ± SD for downstream-task and prior-capability scores.

	Commonsense		Mathematical	
Method	Downstream avg.	Prior avg.	Downstream avg.	Prior avg.
LoRA	61.46 ± 1.43	21.62 ± 1.26	38.03 ± 1.33	37.09 ± 0.21
MiLoRA	60.95 ± 0.87	22.98 ± 0.67	37.85 ± 0.45	38.11 ± 0.25
LoRA-Null	61.64 ± 1.31	20.97 ± 0.73	37.61 ± 0.72	37.09 ± 0.14
PiSSA	61.63 ± 1.30	22.95 ± 0.51	37.36 ± 0.11	35.01 ± 0.10
DoRA	61.25 ± 1.14	21.54 ± 0.97	38.13 ± 0.14	37.72 ± 0.23
OPLoRA-16	62.01 ± 0.81	21.18 ± 1.48	37.98 ± 0.54	37.81 ± 0.37
OPLoRA-128	61.76 ± 0.90	21.40 ± 1.10	38.21 ± 0.50	37.56 ± 0.16
Our Method				
CARD-LoRA-16	61.50 ± 0.50	25.99 ± 2.98	37.22 ± 0.48	39.33 ± 0.25
CARD-LoRA-32	62.22 ± 0.48	23.38 ± 0.35	37.53 ± 0.71	38.70 ± 0.42
CARD-LoRA-128	62.76 ± 0.78	22.25 ± 1.11	38.25 ± 0.19	38.36 ± 0.30

Note. Mathematical CARD-LoRA results at $k = 32$ are additionally included from the support-size sweep; this configuration is not plotted in Figure 3(c).

Commonsense adaptation. Among the listed configurations, CARD-LoRA at $k = 16$ has the highest mean prior-capability score, 25.99, whereas $k = 128$ has the highest mean downstream score, 62.76. The $k = 32$ configuration occupies an intermediate position: its downstream mean of 62.22 and prior mean of 23.38 exceed the corresponding means of every listed baseline. Relative to OPLoRA-16, these differences are 0.21 and 2.20 percentage points, respectively. This comparison identifies a favorable observed balance, while the separate means show why the support size with the strongest prior score need not maximize downstream performance. The prior-score SD at $k = 16$ is 2.98, the largest in that column, so its higher mean should be read together with its greater observed run-to-run variation.

Mathematical adaptation. CARD-LoRA at $k = 16$ has the highest listed mean prior-capability score, 39.33, compared with 38.11 for MiLoRA, the strongest baseline on this group mean. The difference is 1.22 percentage points, while its downstream mean is 0.63 points lower than MiLoRA's. At $k = 128$, CARD-LoRA instead reaches the highest listed downstream mean, 38.25, with a prior mean of 38.36. Its downstream advantage over OPLoRA-128 is only 0.04 points in the reported means. These results distinguish an advantage in observed mean score from a claim about how consistently one method outperforms another across runs.

D.3 Sensitivity to Input-Support Size

Table 10 varies k while retaining nominal rank $r = 8$. Within a replicate, each support is selected from the same task-specific discovery scores, and each adapter is trained separately. The configurations within that replicate therefore share the discovery outcome; they are not independent discovery samples for assessing the effect of k.

Table 10: Support-size sweep for CARD-LoRA: mean ± SD corresponding to Figure 3(b,d).

	Commonsense		Mathematical	
k	Downstream avg.	Prior avg.	Downstream avg.	Prior avg.
16	61.50 ± 0.50	25.99 ± 2.98	37.22 ± 0.48	39.33 ± 0.25
32	62.22 ± 0.48	23.38 ± 0.35	37.53 ± 0.71	38.70 ± 0.42
48	61.84 ± 0.75	22.99 ± 0.59	37.88 ± 0.84	38.41 ± 0.29
64	62.20 ± 0.59	22.52 ± 0.99	37.32 ± 0.97	37.84 ± 0.43
80	62.25 ± 1.36	22.26 ± 1.51	37.86 ± 0.31	38.44 ± 0.29
96	62.61 ± 0.47	22.14 ± 1.13	37.59 ± 0.51	38.45 ± 0.30
112	62.65 ± 0.37	22.05 ± 1.15	38.18 ± 0.88	38.40 ± 0.51
128	62.76 ± 0.78	22.25 ± 1.11	38.25 ± 0.19	38.36 ± 0.30

Endpoint comparison. From $k = 16$ to $k = 128$, the commonsense downstream mean increases by 1.26 percentage points, while its prior mean decreases by 3.74 points. For mathematical adaptation, the corresponding changes are an increase of 1.03 points downstream and a decrease of 0.97 points on the prior group. In both domains, the largest tested support attains the highest mean downstream score and the smallest attains the highest mean prior score. These are comparisons among the tested configurations, not claims of optimality over all possible support sizes.

Intermediate configurations. The intermediate values do not follow a uniformly improving trajectory. For commonsense adaptation, the downstream mean decreases from 62.22 at $k = 32$ to 61.84 at $k = 48$, then rises at larger tested sizes. For mathematical adaptation, it decreases from 37.88 at $k = 48$ to 37.32 at $k = 64$, then rises to 37.86 at $k = 80$. Prior performance is also non-monotonic: the mathematical prior mean rises from 37.84 at $k = 64$ to 38.44 at $k = 80$. These observations concern the reported means and do not establish the reliability of each adjacent difference. Because all tested k satisfy $k \geq r$, the sweep changes the input span available to the update without increasing its rank ceiling.

D.4 Interpreting Group Scores and Error Bars

Relation to combined averages. The downstream and prior columns are group means, not the combined averages in the main results tables. Writing D and P for these two group scores, the main commonsense average weights them as $(8D + 3P)/11$, whereas the mathematical average uses $(5D + 3P)/8$. The role and restriction-side ablations instead equally weight four domain–group scores. Consequently, the configuration preferred by a combined average can differ from the one preferred by the prior score alone. An SD for a combined score must likewise be computed from the run-level combined scores: the group SDs alone do not determine it without their covariance. The absolute prior means in the two fine-tuning domains also refer to different evaluation groups and should not be interpreted as measurements on an identical benchmark suite.

Scope of uncertainty. The error bars and shaded bands represent one sample SD, not standard errors or confidence intervals. They describe empirical variation across the five runs and do not, by themselves, constitute a significance test of method or support-size differences. Neither overlap nor separation of these SD intervals alone supplies such a test. The reported performance variation also does not isolate the contributions of discovery and adapter optimization, or measure the stability of selected direction identities. The tables therefore support descriptive comparisons of observed means and variability; the structural analysis in Appendix B addresses the scores and supports of its specified discovery outcomes separately.