AEROLORA: YOU NEED FEW OF THEM

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank adaptation (LoRA) reduces the number of trainable parameters for fine-tuning large language models (LLMs), but two limitations remain: (i) maintaining separate adapters for multiple tasks incurs cumulative storage costs; and (ii) all low-rank channels remain active for every input, leaving input-dependent sparsity unexploited. We propose AeroLoRA (Adaptive Expert Routing with Orthogonal LoRA), a two-stage sparse adaptation framework that reduces both trainable parameter counts and the number of active adapter parameters per token. In the first stage, we derive a task-specific binary mask from the weight magnitudes of the trained up-projection matrix B, restricting subsequent adaptation to the selected parameter subset. In the second stage, we update only the selected parameters and perform TopK routing based on activations from a frozen sparse projection, dynamically selecting a small subset of low-rank channels for each input without an additional trainable router. Experiments on Mistral and Llama across four benchmarks show that AeroLoRA achieves the best performance among the evaluated methods in single-task adaptation, multi-task adapter merging, and continual learning. During the second stage, trainable parameters account for only 0.24% and 0.22% of the total model parameters for Mistral and Llama, respectively, while AeroLoRA also activates the fewest adapter parameters per token among the evaluated methods. Spectral analysis shows that its adapter update matrices exhibit the highest effective rank among the compared methods. Subspace analysis further reveals that the update subspaces associated with TopK-selected channels align with the dominant directions of task gradients. These findings provide empirical evidence that diverse, task-relevant update directions and strong adaptation performance can coexist with very few trainable parameters and sparse, input-dependent channel activation.

1 INTRODUCTION

Adapting a pretrained language model requires learning a task-specific update, but not changing all of its parameters. Low-rank adaptation (LoRA) makes this practical by learning two small matrices while keeping the backbone fixed (Hu et al., 2022). Yet a standard adapter still trains every entry of both factors. Reducing its rank lowers this parameter count, but also narrows the space of possible updates. This motivates a different question: can we retain a rich set of update directions while learning only a small subset of adapter weights?

LoRI (Zhang et al., 2025a) addresses weight selection through task-specific masks of the up-projection, while FlyLoRA (Zou et al., 2025) uses a frozen sparse projection to select rank channels for each input. These restrictions operate at different levels: a mask determines which weights a task may learn, whereas routing determines which of them participate for a token. However, weight calibration observes all channels together, while routing changes which columns receive each token's gradient. Combining the two therefore requires the calibrated support to remain effective under a different pattern of channel use.

We introduce **AeroLoRA** (Adaptive Expert Routing with Orthogonal LoRA), which separates selecting the weights to learn from selecting the channels to use (Figure 1). Stage 1 trains the up-projection B with all channels active and derives a task mask M from its weight magnitudes. Stage 2 resets B to zero and retrains only the retained weights with input-dependent TopK selection. Both stages use the same frozen sparse projection A, so the coordinates selected during calibration re-

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.

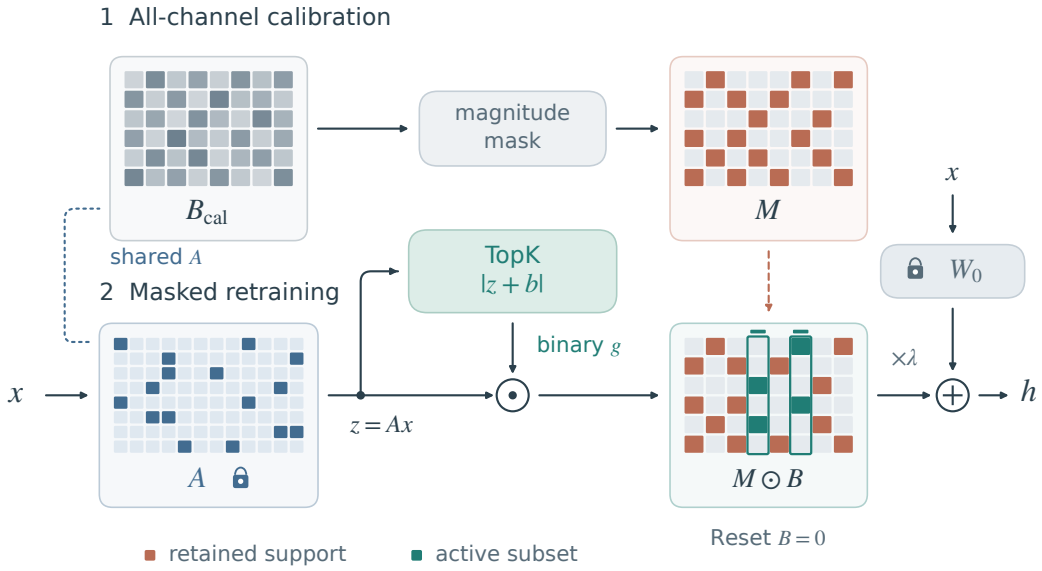


Figure 1: AeroLoRA reuses frozen A across stages (dotted link). The dashed arrow passes the mask M from calibration to Stage 2, which resets B to zero and masks weights and gradients. Outlined columns mark TopK selections; teal cells mark their retained entries. Selected activations preserve signed Ax, scaled by $\lambda = \alpha/r$. Locks denote frozen weights; the cells are schematic.

tain their meaning during retraining. Routing uses the projected activations without an additional trainable router.

On Llama-3.1-8B and Mistral-7B-v0.3, AeroLoRA trains approximately 17.6M weights in Stage 2—79% fewer than rank-32 LoRA, or 0.22% and 0.24% of the respective backbone sizes. With this small subset, it achieves the highest mean scores in all single-task and adapter-merging comparisons, and higher target-task accuracy in sequential adaptation on Llama. Structural diagnostics further show that the all-channel reference updates retain broadly distributed spectra, while selected channel subspaces align closely with task gradients.

Our main contributions are as follows:

- We develop a two-stage adapter that selects weights through calibration, then retrains them under dynamic channel selection while preserving their projection coordinates.
- We show that this sparse trainable subset supports strong single-task, merged, and sequential target-task performance, while routing further reduces per-token active support.
- We find that sparse learned coefficients retain broad update spectra, and that dynamic channel selections align more closely with task gradients than fixed or random selections.

2 Related Work

LoRA (Hu et al., 2022) represents a task update as a product of low-rank factors. AdaLoRA (Zhang et al., 2023b) allocates rank according to importance, DoRA (Liu et al., 2024) separates weight magnitude and direction, and MELoRA (Ren et al., 2024) combines smaller adapters into block-diagonal updates. Fixing projection factors further reduces the learned coefficients: LoRA-FA (Zhang et al., 2023a) freezes A and trains B, reducing activation storage, while VeRA (Kopiczko et al., 2024) learns scaling vectors over frozen random matrices shared across layers.

Sparsity can restrict the weights learned for a task or the components used for an input. SparseAdapter (He et al., 2022) prunes adapters at initialization; LoRI (Zhang et al., 2025a) selects task-specific masks from calibrated B magnitudes under frozen projections. MixLoRA (Li et al., 2024) uses a learned router to choose among LoRA experts, while FlyLoRA (Zou et al., 2025)

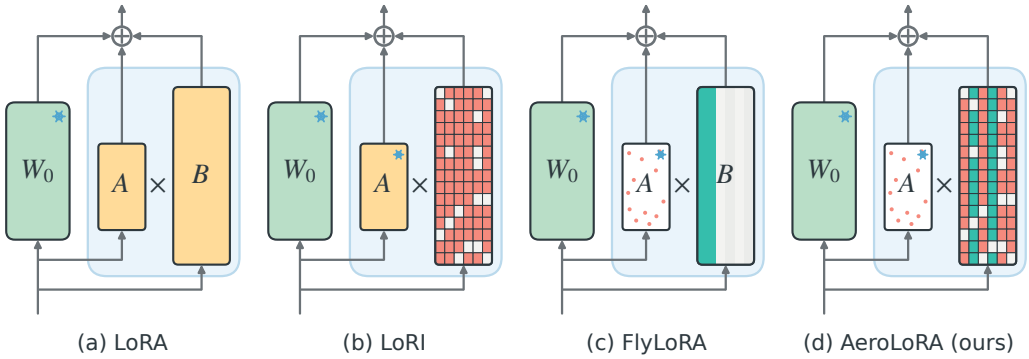


Figure 2: LoRA, LoRI, FlyLoRA, and AeroLoRA parameterizations. Snowflakes mark frozen matrices. LoRI masks entries of B, FlyLoRA selects channels for each input, and AeroLoRA applies both restrictions. Coral cells indicate retained entries; teal columns indicate selected channels. The matrix layouts are schematic.

selects rank channels within an adapter through a frozen sparse projection. An element-wise mask acts at a finer level, restricting the weights inside those channels. AeroLoRA combines a calibrated mask with channel selection, keeping the projection fixed so that calibration and retraining use the same coordinates. Figure 2 compares these parameterizations.

Compact adapters can also be combined or adapted to further tasks. Task Arithmetic (Ilharco et al., 2023) adds task-specific weight differences, TIES-Merging (Yadav et al., 2023) trims updates and resolves sign conflicts, and LoRAHub (Huang et al., 2024) searches for adapter-combination coefficients using target-task examples. Sequential adaptation instead continues training an existing adapter. Continual-learning methods such as O-LoRA (Wang et al., 2023) additionally seek to preserve earlier tasks through constraints on task subspaces. Our composition and sequential-adaptation experiments examine whether adapters learned with few trainable weights remain effective when combined or trained on a new target task.

3 Method

3.1 Fixed projection and two-stage design

Consider a pretrained linear map $W_0 \in \mathbb{R}^{m\times n}$ and input $x \in \mathbb{R}^n$. The adapter has a down-projection $A \in \mathbb{R}^{r\times n}$, an up-projection $B \in \mathbb{R}^{m\times r}$, and scale $\lambda = \alpha/r$. To identify a small trainable subset, Stage 1 calibrates B using all r channels and selects its weight support. Stage 2 then retrains the retained weights while selecting k channels per token. Both W_0 and A stay fixed throughout.

We initialize A once for each task adapter, before Stage 1, at target density $\rho = k/r$ for the Stage 2 channel budget k. Independently for each row, we sample $p = \max\{1, \lfloor n\rho \rfloor\}$ positions uniformly without replacement from the n input coordinates. Their values are drawn independently from $\mathcal{N}(0, 1/r^2)$; the remaining entries are zero. Each adapter reuses its sampled positions and values across both stages; different task adapters need not share the same draw.

The term *Orthogonal* refers to the initialization geometry of the random projection. Independently sampled directions have small average correlation in high dimensions. Appendix A.1 characterizes this statistical property through the moments of the sampled rows.

During training, the projected features are

$$z(x) = \mathrm{sg}(Ax), \tag{1}$$

where sg preserves the forward value Ax and blocks gradients through the projection branch, including to its input. Training updates B without differentiating through $z(x)$, while input gradients still pass through the frozen backbone map W_0x.

3.2 Stage 1: all-channel calibration and support selection

We initialize B_{cal} to zero and train all its entries with the task loss $\mathcal{L}$. Calibration uses every channel:

$$h_{\text{cal}}(x) = W_0 x + \lambda B_{\text{cal}} z(x). \tag{2}$$

Following the calibration-and-reset procedure of LoRI (Zhang et al., 2025a), we collect the magnitudes of all adapted B_{cal} weights in a vector v. A single global threshold then defines the task mask:

$$\tau_s = \text{Quantile}_s(v), \qquad M_{ij}^{(\ell)} = \mathbf{1}\Big\{|B_{\text{cal},ij}^{(\ell)}| \geq \tau_s\Big\}. \tag{3}$$

Here ℓ indexes adapted modules and s is the target fraction of excluded weights. The global threshold allows modules and columns to retain different numbers of entries; ties and integer counts can make the realized retention differ from $1 - s$.

Stage 2 reuses A and M but initializes every entry of B to zero, transferring the selected support from calibration and learning its coefficients afresh. Because column j of B multiplies row j of A, keeping A fixed preserves the projection direction associated with each retained coordinate. Routing can then select different columns for each token, while the mask consistently restricts learning to the calibrated entries within those columns.

3.3 Stage 2: masked training with dynamic channel selection

Following FlyLoRA (Zou et al., 2025), we route projected activations through selected rank channels. Each module has a bias $b \in \mathbb{R}^r$, applied inside the absolute value used for selection. For $1 \leq k \leq r$, the selected set and binary gate are

$$S(x) = \text{TopK}(|z(x) + b|, k), \qquad g_j(x) = \mathbf{1}\{j \in S(x)\}. \tag{4}$$

TopK returns exactly k indices, so $\sum_j g_j(x) = k$. The bias and absolute value determine the selection, while B receives the selected signed activations $z(x)$ without softmax or renormalization. With masked weights $C = M \odot B$, the layer output is

$$h(x) = W_0 x + \lambda C\big(z(x) \odot g(x)\big). \tag{5}$$

Inputs selecting the same channels share a linear forward update; across selections, the adapter defines an input-dependent family of updates. Backpropagation follows the stop-gradient rule in Equation 1.

Let $\delta(x) = \partial\mathcal{L}/\partial h(x)$. At one invocation of the module, the contribution to the task-loss gradient of B is

$$G_B(x) = \lambda M \odot \Big[\delta(x)\big(z(x) \odot g(x)\big)^\top\Big]. \tag{6}$$

The gradient contribution is zero outside the mask and in unselected columns. Stage 1 uses no routing bias. To adjust channel usage in Stage 2, each module's bias starts at zero and is updated from channel counts, without task-loss gradients or an auxiliary balancing loss. For the tokens $\mathcal{T}$ counted in the current batch,

$$c_j = \sum_{x \in \mathcal{T}} g_j(x), \qquad \bar{c} = \frac{1}{r}\sum_{j=1}^{r} c_j, \qquad b_j \leftarrow b_j + \eta_b \,\text{sign}(\bar{c} - c_j). \tag{7}$$

Each token selects k channels, so $\bar{c} = |\mathcal{T}|k/r$. The count-based update increases the biases of below-average channels and decreases those of above-average channels. The bias is saved as adapter state and held fixed at inference; routing remains input-dependent. Training and inference both use $k = 8$ by default. Appendix A.2 details the state and score conventions.

3.4 Parameter budgets and support overlap

We distinguish trainable weights from per-token active support. For one module, let $d_j = \sum_i M_{ij}$ count retained entries in column j, $Q = \|M\|_0$ be the total retained support, and $q = Q/(mr)$ its realized fraction. The Stage 2 trainable count and active B support for token x are

$$N_{\text{train}}^{(2)} = Q = \sum_{j=1}^{r} d_j, \qquad N_B(x) = \sum_{j \in S(x)} d_j = \sum_{j=1}^{r} d_j g_j(x). \tag{8}$$

Summing over modules gives the model-level budgets. Table 1 uses Stage 2 counts; Stage 1 trains and uses all mr entries of each B. These are support counts, including coordinates with zero weights or activations. Nonuniform column retention allows tokens with the same k to activate different amounts of B support.

When retained counts are approximately uniform across columns, $d_j \approx qm$, the active-support ratio becomes

$$\gamma_B(x) = \frac{N_B(x)}{mr} \approx q\frac{k}{r}. \tag{9}$$

With $r = 32$, $k = 8$, and nominal retention $q = 0.4$, this gives an active B budget of approximately 10% of dense B. The ratio combines the retained fraction of weights with the fraction of selected channels. Appendix A.3 gives the counting scope and expectations over inputs.

Table 1: Adapter budgets for one module of rank r. For masked methods, q is the realized retained fraction of B. AeroLoRA's active-count approximations assume similar retained densities across columns. The last two columns count B entries only.

Method	Trainable weights	Active B / token	Rel. dense B
LoRA	$(n+m)r$	mr	1
FlyLoRA	mr	mk	k/r
LoRI-S	qmr	qmr	q
AeroLoRA	qmr	$\approx qmk$	$\approx qk/r$

Sparse weight support still permits multiple update directions. Each selected channel contributes one outer product: the all-channel reference $\overline{\Delta W} = \lambda CA$ has rank at most r, while a token's conditional update has rank at most k. Different tokens can therefore use different subspaces (Appendix A.4). For task adapters with aligned projection coordinates, gating also restricts coordinate overlap to entries selected by both tasks; composition further depends on their projection directions and learned coefficients (Appendix A.3).

3.5 Static composition of reference updates

To examine how compact adapters combine across tasks, write $C_t = M_t \odot B_t$ for task t. LoRI's static Concat construction sums weighted updates $\lambda C_t A_t$, whereas Linear combines the factors before multiplication, introducing cross-task products and squared within-task coefficients (Zhang et al., 2025a). Appendix B gives the full definitions, with merge coefficients distinct from λ. FlyLoRA also analyzes static update sums (Zou et al., 2025). These operators describe all-channel reference matrices; the active support of a routed composition additionally depends on how its channels are selected.

4 Experiments

We first test whether learning a small subset of adapter weights supports accurate single-task adaptation, then examine merging and adaptation to new targets. Ablations isolate the roles of weight selection and routing, while structural diagnostics assess the diversity and task relevance of the resulting updates.

4.1 Experimental setup

Our backbones are Meta-Llama-3.1-8B and Mistral-7B-v0.3. We adapt all attention and feed-forward projections in every Transformer layer. Evaluation covers general knowledge, science question answering, mathematical reasoning, and code generation: we report accuracy on MMLU (Hendrycks et al., 2021), ScienceQA (Lu et al., 2022), and GSM8K (Cobbe et al., 2021). For code, we train on CodeAlpaca (Chaudhary, 2023) and evaluate HumanEval (Chen et al., 2021) pass@1, pass@5, and pass@10.

The baselines are full fine-tuning (FFT), LoRA (Hu et al., 2022) at $r = 8$ and $r = 32$, LoRI-S (Zhang et al., 2025a) with a target exclusion fraction of 60% for B, and FlyLoRA (Zou et al., 2025). For a given backbone, adapter methods use the same task data and target modules, with each method's own training procedure. The defaults are $r = 32$ and $\alpha = 64$. AeroLoRA uses $k = 8$ in Stage 2 training and inference, $s = 0.6$ for B, and a bias-update step of 10^{-3}. Its two stages follow Section 3.

We train on four NVIDIA L20 GPUs with 48 GB each, using FSDP, bfloat16 mixed precision, and activation checkpointing. For two-stage methods, reported trainable fractions refer to Stage 2. We count active B support as defined in Section 3.4; the superscripts in Table 2 identify what each weight total includes.

All performance scores and structural diagnostic measurements are arithmetic means over the same fixed set of five random seeds. We compare methods by their mean scores for each backbone, task, and evaluation metric. Parameter counts and analytical support budgets follow the conventions in Section 3.4.

4.2 Single-task adaptation

AeroLoRA has the highest mean scores on all six metrics for both backbones (Table 2), exceeding the strongest baseline by 0.4–1.6 percentage points across the twelve comparisons. It does so while training approximately 17.6M weights in Stage 2, compared with 83.9M for rank-32 LoRA: a 79% reduction. This subset matches LoRI-S's trainable budget and amounts to 0.22% of Llama and 0.24% of Mistral. Routing further reduces nominal active B support to 4.40M entries per token, 75% below LoRI-S and 60% below FlyLoRA.

Table 2: Single-task performance (%; higher is better). Scores are means over five random seeds. Train. reports trainable parameters as a percentage of backbone parameters (Stage 2 for two-stage methods). Weight scopes: [a] full model; [b] adapter $A+B$; [c] per-token active B only (nominal budgets for masked methods). Bold marks the highest mean score per backbone.

						HumanEval		
Method	Train. (%)	Weights (M)	MMLU	SciQA	GSM8K	pass@1	pass@5	pass@10
Llama-3.1-8B								
FFT	100	8030[a]	38.7	90.1	55.3	29.9	52.7	61.3
LoRA ($r = 8$)	0.26	21.0[b]	37.3	90.5	54.9	30.1	53.8	63.5
LoRA ($r = 32$)	1.04	83.9[b]	39.4	92.3	57.1	34.7	58.5	69.1
FlyLoRA ($k = 8$)	0.55	11.0[c]	40.1	93.0	57.9	35.3	60.4	70.6
LoRI-S	0.22	17.6[c]	39.7	92.7	57.5	34.9	59.1	69.5
AeroLoRA	0.22	4.40[c]	**41.6**	**94.1**	**58.5**	**36.8**	**61.7**	**72.2**
Mistral-7B-v0.3								
FFT	100	7248[a]	35.1	73.7	58.3	31.5	42.3	48.9
LoRA ($r = 8$)	0.29	21.0[b]	35.4	72.9	58.7	31.9	43.1	49.5
LoRA ($r = 32$)	1.16	83.9[b]	37.2	78.5	61.3	35.4	47.7	52.3
FlyLoRA ($k = 8$)	0.61	11.0[c]	37.9	79.1	61.6	36.1	48.3	54.1
LoRI-S	0.24	17.6[c]	37.6	78.9	61.5	35.7	47.9	54.8
AeroLoRA	0.24	4.40[c]	**39.3**	**80.6**	**62.9**	**36.5**	**49.8**	**55.6**

4.3 Multi-task adapter merging

We next examine whether the compact task adapters remain effective when combined. We merge the four adapters using Linear and Concat and evaluate each merged model on all four benchmarks. Table 3 reports the scores, while Appendix B provides the single-task references and per-metric changes.

AeroLoRA has the highest mean scores for both strategies, both backbones, and every metric, with margins of 0.7–2.6 percentage points over the strongest baseline under the same strategy. Its ad-

vantage therefore extends from individual adapters to their compositions across knowledge, science, mathematics, and code generation. Concat generally gives higher scores than Linear, with the exceptions confined to HumanEval pass@5.

Table 3: Adapter-merging performance (%). Scores are means over five random seeds. Bold marks the highest mean score per backbone and merge strategy; red shading marks AeroLoRA. Appendix B provides full before/after results and changes in percentage points.

Model	Method	Merge status	MMLU	SciQA	GSM8K	HumanEval		
						pass@1	pass@5	pass@10
Llama-3.1-8B	LoRA ($r = 8$)	Linear	30.3	28.7	23.2	16.1	46.2	58.0
		Concat	31.8	32.3	26.7	18.3	47.6	58.8
	LoRA ($r = 32$)	Linear	34.5	33.8	27.5	23.7	50.9	63.8
		Concat	35.1	37.1	30.2	24.9	51.7	64.6
	FlyLoRA ($k = 8$)	Linear	37.5	46.3	33.7	30.1	54.5	66.2
		Concat	37.8	47.9	35.1	30.6	54.1	66.3
	LoRI-S	Linear	37.3	46.5	33.9	29.8	53.4	65.3
		Concat	37.6	48.1	35.1	30.4	53.5	65.8
	AeroLoRA	Linear	**39.4**	**48.8**	**35.7**	**32.1**	**56.3**	**68.2**
		Concat	**39.7**	**50.1**	**36.8**	**32.6**	**55.9**	**68.7**
Mistral-7B-v0.3	LoRA ($r = 8$)	Linear	29.7	23.1	26.3	17.3	35.9	43.7
		Concat	30.1	25.4	28.4	19.2	36.6	44.4
	LoRA ($r = 32$)	Linear	32.5	30.8	30.7	25.8	42.9	50.8
		Concat	33.1	33.7	32.8	27.1	43.6	51.5
	FlyLoRA ($k = 8$)	Linear	35.1	40.3	37.8	32.3	45.1	52.5
		Concat	35.4	41.8	39.1	32.8	44.5	53.0
	LoRI-S	Linear	35.0	40.8	38.2	32.2	44.9	52.4
		Concat	35.3	42.2	39.4	32.9	44.4	52.9
	AeroLoRA	Linear	**37.1**	**43.3**	**40.7**	**33.4**	**45.9**	**53.2**
		Concat	**37.3**	**44.8**	**41.3**	**33.8**	**45.4**	**53.9**

4.4 Sequential adaptation

Beyond combining independently trained adapters, we test continued learning on a new task. On Llama-3.1-8B, we evaluate two independent sequences: GSM8K → MMLU and GSM8K → ScienceQA. Table 4 reports GSM8K accuracy before continuation and target-task accuracy afterward.

AeroLoRA reaches 40.5% on MMLU and 92.8% on ScienceQA, exceeding LoRI-S by 0.9 and 1.4 percentage points, respectively.

Table 4: Sequential adaptation on Llama-3.1-8B (accuracy, %). Scores are means over five random seeds. Each GSM8K adapter is continued separately on MMLU and ScienceQA. GSM8K scores describe the starting checkpoints; target scores are measured after continuation.

	Source task	Target task after continuation	
Method	GSM8K (before)	→ MMLU	→ ScienceQA
LoRA ($r = 8$)	54.90	35.20	88.76
LoRI-S ($r = 32$)	57.50	39.60	91.40
AeroLoRA ($r = 32$)	**58.50**	**40.50**	**92.80**

4.5 Ablation studies

To separate the roles of weight selection and channel selection, we remove either element-wise masking or TopK on Mistral-7B-v0.3 (Table 5). Removing the mask makes every entry of B trainable while retaining conditional channel selection. Removing TopK keeps the sparse trainable support but activates all channels. These variants distinguish the number of weights learned from the number used for each token.

The full model achieves the highest scores on all four benchmarks. The largest gains occur on ScienceQA: accuracy reaches 80.6%, exceeding the variants without masking and without TopK by 6.1 and 6.7 percentage points, respectively. It also uses 60% and 75% less nominal active B support than these variants. Together, the comparisons support combining calibrated weight support with dynamic channel selection for accurate adaptation with few learned parameters.

Table 5: Component ablations on Mistral-7B-v0.3. Task scores are percentages averaged over five random seeds; SciQA denotes ScienceQA accuracy. Train. reports the Stage 2 trainable fraction of backbone parameters; Active B is the nominal per-token budget. Full AeroLoRA uses $r = 32$, $k = 8$, and $s = 0.6$. Bold marks column optima, including ties: lower budgets and higher mean scores are preferred.

Variant	Train. (%)	Active B (M)	MMLU	SciQA	GSM8K	HumanEval pass@1
w/o element mask	0.61	11.0	37.8	74.5	61.7	35.2
w/o TopK selection	**0.24**	17.6	37.5	73.9	61.4	34.8
AeroLoRA	**0.24**	**4.40**	**39.3**	**80.6**	**62.9**	**36.5**

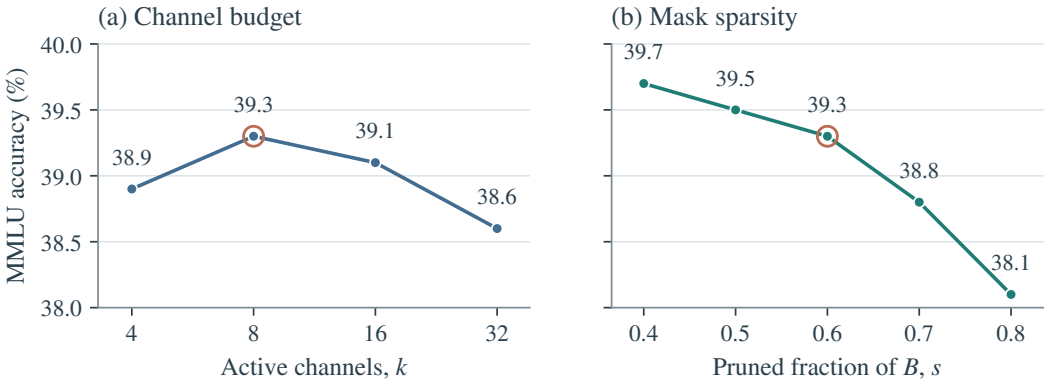


Figure 3: Hyperparameter sensitivity on Mistral-7B-v0.3, evaluated on MMLU. Each point is the mean over five random seeds. (a) Each k configuration is trained separately with $r = 32$, $s = 0.6$, and projection density $\rho = k/r$. (b) Varying the pruned fraction of B at $r = 32$, $k = 8$. Circled points mark the defaults.

Figure 3 examines how much support each selection requires. Among the tested channel budgets, $k = 8$ gives the highest MMLU accuracy, 39.3%; this sweep also changes projection density $\rho = k/r$. At fixed $k = 8$, the default $s = 0.6$ nominally retains 40% of B, compared with 60% at $s = 0.4$, reducing support by one-third with accuracies of 39.3% and 39.7%, respectively. Increasing sparsity further to $s = 0.8$ yields 38.1%, showing the accuracy tradeoff as fewer weights are learned.

4.6 Effective rank and gradient alignment

The preceding experiments show that AeroLoRA adapts effectively with few trainable weights. We now examine whether these weights represent diverse update directions and whether routing se-

lects directions aligned with task gradients. These diagnostics use GSM8K with $r = 32$ on both backbones; AeroLoRA uses its default configuration.

We measure the spectral-entropy effective rank (Roy & Vetterli, 2007) of the all-channel reference update $\overline{\Delta W} = \lambda CA$, averaging across adapted modules within each run and then over five seeds (Appendix A.4). AeroLoRA attains the highest effective ranks: 31.29 on Llama and 31.24 on Mistral, corresponding to 97.8% and 97.6% of $r = 32$ (Table 6). These values indicate broadly distributed reference spectra. Each conditional update has rank at most $k = 8$, while different inputs can select different columns of C and use different subspaces.

To assess the task relevance of these input-dependent subspaces, we follow task-gradient analyses of LoRA (Zhang et al., 2025b) and compare the span of the selected columns of $C = M \odot B$ with the leading k-dimensional left-singular subspace of $G = \nabla_W \mathcal{L}$. Alignment is the mean principal-angle cosine, $d^{-1} \sum_{i=1}^{d} \cos\theta_i$, where d is the number of principal angles; larger values indicate closer alignment. This angle average defines each subspace-pair diagnostic, whose summaries are then averaged over five seeds. For each AeroLoRA checkpoint, we fix the evaluation samples and reference gradients and compare three eight-channel rules: random selection, a fixed set of the most frequently selected channels, and input-dependent TopK.

Dynamic TopK reaches 0.89 on Llama-3.1-8B and 0.91 on Mistral-7B-v0.3, compared with 0.62 and 0.66 for fixed high-frequency selection. Because all three rules select eight channels from the same learned adapter, this comparison isolates the effect of channel selection on alignment. Input-dependent selection aligns more closely with task-gradient directions than a fixed set chosen by aggregate selection frequency.

Table 6: Update diagnostics on GSM8K, averaged over five random seeds. Parentheses report effective rank as a percentage of $r = 32$. Gradient alignment compares selection rules within each AeroLoRA checkpoint. Bold marks the highest mean per group and backbone.

Method / selection rule	Llama-3.1-8B	Mistral-7B-v0.3
Effective rank of $\overline{\Delta W}$ ($r = 32$)		
LoRA	29.57 (92.4%)	29.04 (90.8%)
LoRI-S	30.25 (94.5%)	30.13 (94.2%)
FlyLoRA	30.03 (93.8%)	29.98 (93.7%)
AeroLoRA	**31.29 (97.8%)**	**31.24 (97.6%)**
Gradient alignment: mean principal-angle cosine ($k = 8$)		
Random	0.57	0.52
Fixed high-frequency	0.62	0.66
Dynamic TopK	**0.89**	**0.91**

5 Conclusion

AeroLoRA adapts pretrained models by learning a small set of adapter weights while keeping the backbone and sparse projection fixed. All-channel calibration identifies the task-specific trainable support; retraining then learns the retained weights under input-dependent channel selection. With approximately 17.6M Stage 2 trainable parameters, AeroLoRA achieves the highest mean scores in the single-task and merging comparisons on Llama-3.1-8B and Mistral-7B-v0.3. It also improves target-task accuracy in sequential adaptation.

Structural diagnostics show that updates learned on sparse support retain broad all-channel spectra, while dynamic channel selection yields closer alignment with task gradients than fixed or random selection. A small set of learned adapter weights can therefore support accurate adaptation while retaining diverse update directions.

AI USE STATEMENT

AI assistance was used to reorganize the existing manuscript for the ICLR format and prepare wording and technical descriptions. The authors are responsible for the manuscript, including its citations, mathematical statements, code, and reported results.

REPRODUCIBILITY STATEMENT

The implementation and analysis code are provided in the supplementary material.

REFERENCES

Sahil Chaudhary. Code Alpaca: An instruction-following LLaMA model for code generation. GitHub repository, 2023. URL https://github.com/sahil280114/codealpaca.

Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. SparseAdapter: An easy approach for improving the parameter-efficiency of adapters. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 2184–2190, 2022. doi: 10.18653/v1/2022.findings-emnlp.160. URL https://aclanthology.org/2022.findings-emnlp.160/.

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

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 *Proceedings of the International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. LoRAHub: Efficient cross-task generalization via dynamic LoRA composition. In *Proceedings of the Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2307.13269.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *Proceedings of the International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2212.04089.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based random matrix adaptation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NjNfLdxr3A.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. MixLoRA: Enhancing large language models fine-tuning with LoRA-based mixture of experts, 2024. URL https://arxiv.org/abs/2404.15159.

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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121, 2024. URL https://proceedings.mlr.press/v235/liu24bn.html.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2209.09513.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. MELoRA: Mini-ensemble low-rank adapters for parameter-efficient fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3052–3064, 2024. doi: 10.18653/v1/2024.acl-long.168. URL https://aclanthology.org/2024.acl-long.168/.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *Proceedings of the 15th European Signal Processing Conference*, pp. 606–610, 2007. URL https://www.eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/a5p-h05.pdf.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023. doi: 10.18653/v1/2023.findings-emnlp.715. URL https://aclanthology.org/2023.findings-emnlp.715/.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html.

Juzheng Zhang, Jiacheng You, Ashwinee Panda, and Tom Goldstein. LoRI: Reducing cross-task interference in multi-task low-rank adaptation. In *Proceedings of the Conference on Language Modeling*, 2025a. URL https://arxiv.org/abs/2504.07448.

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

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=lq62uWRJjiY.

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, 2025b. URL https://proceedings.mlr.press/v267/zhang25ax.html.

Heming Zou, Yunliang Zang, Wutong Xu, Yao Zhu, and Xiangyang Ji. FlyLoRA: Boosting task decoupling and parameter efficiency via implicit rank-wise mixture-of-experts. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2510.08396.

A METHOD DETAILS AND DERIVATIONS

A.1 STATISTICAL GEOMETRY OF THE RANDOM PROJECTION

The frozen projection supplies the directions whose coefficients are learned in B. To characterize its initialization, let $a_j \in \mathbb{R}^n$ be the transpose of row j of A, with $p = \max\{1, \lfloor nk/r \rfloor\}$ nonzero entries. Rows are initialized independently: each selects p positions uniformly without replacement and assigns them independent values from $\mathcal{N}(0, 1/r^2)$. Since each coordinate is selected with probability p/n, the first and second moments are

$$\mathbb{E}[a_j] = 0, \qquad \mathbb{E}[a_j a_j^\top] = \frac{p}{nr^2} I_n, \qquad \mathbb{E}[AA^\top] = \frac{p}{r^2} I_r. \tag{10}$$

To examine directional correlation, define $\hat{a}_j = a_j / \|a_j\|_2$ for analysis; the algorithm leaves the rows of A unnormalized. The row norm is positive with probability one because $p \geq 1$. Sign symmetry gives zero off-diagonal second moments, and permutation symmetry makes the diagonal moments equal. Since $\|\hat{a}_j\|_2^2 = 1$,

$$\mathbb{E}[\hat{a}_j] = 0, \qquad \mathbb{E}[\hat{a}_j \hat{a}_j^\top] = \frac{1}{n} I_n. \tag{11}$$

For distinct rows $j \neq \ell$, independence then gives

$$\begin{aligned} \mathbb{E}[\hat{a}_j^\top \hat{a}_\ell] &= 0, \\ \mathbb{E}[(\hat{a}_j^\top \hat{a}_\ell)^2] &= \operatorname{tr}\big(\mathbb{E}[\hat{a}_j \hat{a}_j^\top]\mathbb{E}[\hat{a}_\ell \hat{a}_\ell^\top]\big) = \frac{1}{n}. \end{aligned} \tag{12}$$

Thus the root-mean-square directional correlation is $n^{-1/2}$ under both sparse and dense initialization, although their correlation distributions can differ. This statistic characterizes the independently initialized rows of A. The geometry of a learned task update additionally depends on its up-projection coefficients and mask.

A.2 SUPPORT SELECTION AND INFERENCE STATE

Global magnitude selection can retain different fractions of entries across modules and columns. Threshold ties and integer rounding can also cause the realized retention to differ from the target $1 - s$. The final masks determine exact trainable support; the channels selected for each token determine how much of that support is active.

The adapter state consists of A, B, M, the routing bias b, and the configuration values r, k, α. Stage 2 retains the frozen projection and the mask from Stage 1, and initializes B and b to zero. The mask constrains both forward weights and gradients of B. Inference restores this trained state and uses $C = M \odot B$. The bias affects channel selection, while the selected activations retain their original signed Ax values.

Writing $A_{j:}$ for row j, the routing score is $|A_{j:}x + b_j|$: the bias shifts the signed projection before its magnitude is taken. The effect of a bias adjustment on that magnitude depends on the sign of $A_{j:}x + b_j$. Equation 7 updates the bias from assignment counts during training. At inference, the saved bias remains fixed while each token is routed using its own projected activations.

Routing uses projected activations as a surrogate for channel contribution. The norm of channel j's unscaled output is $\|C_{:j}(A_{j:}x)\|_2 = \|C_{:j}\|_2 |A_{j:}x|$. Channel output magnitudes therefore depend on both the projected activation and the norm of the masked column, whereas routing ranks $|A_{j:}x + b_j|$. Section 4.6 evaluates the task relevance of the selected channel subspaces.

In both stages, $z(x) = \operatorname{sg}(Ax)$ preserves the forward value Ax and blocks gradients through the projection branch. Gradients still propagate through the backbone map $W_0 x$ to its input, with W_0 and A frozen.

A.3 PARAMETER COUNTS AND SUPPORT OVERLAP

For a module with $B \in \mathbb{R}^{m \times r}$, let $d_j = \sum_i M_{ij}$, $Q = \|M\|_0$, and $q = Q/(mr)$. The count Q includes all coordinates permitted by the mask, while $N_B(x) = \sum_j d_j g_j(x)$ includes those selected

for the current token. Both count permitted support, including entries whose weights or activations happen to be zero.

Computing the routing scores requires all r projected activations, after which the gate selects columns of B. Accordingly, $N_B(x)$ counts the mask-permitted active support in B; projection through A, channel selection, backbone computation, and the two training stages are separate components of execution cost.

For a fixed trained adapter and input distribution, let $p_j = \Pr_x[j \in S(x)]$ be the marginal activation probability of channel j. By linearity of expectation,

$$\mathbb{E}_x[N_B(x)] = \sum_{j=1}^{r} d_j \mathbb{E}_x[g_j(x)] = \sum_{j=1}^{r} d_j p_j. \tag{13}$$

When each channel has marginal activation probability k/r, the expected count is $\mathbb{E}_x[N_B(x)] = Qk/r$, regardless of dependencies between channel selections. The per-token approximation in Figure 4 uses a different condition: if retained column counts satisfy $d_j \approx qm$, selecting exactly k channels gives $N_B(x) \approx qmk$. The first condition concerns activation frequencies across inputs; the second concerns the distribution of retained weights across columns.

For task adapters t, u with shared or aligned projection coordinates, define $\widetilde{M}_t(x) = M_t \operatorname{Diag}(g_t(x))$. Binary gating can only remove retained coordinates, so the mask overlap and input-dependent active overlap, normalized by all mr entries, satisfy

$$\begin{aligned} O^{\text{mask}}_{t,u} &= \frac{\langle M_t, M_u \rangle_F}{mr}, \\ O^{\text{act}}_{t,u}(x, x') &= \frac{\langle \widetilde{M}_t(x), \widetilde{M}_u(x') \rangle_F}{mr} \le O^{\text{mask}}_{t,u}. \end{aligned} \tag{14}$$

As a reference, suppose each mask is drawn uniformly from binary masks with fixed retained count Q_t or Q_u, independently across tasks. Each coordinate then has retention probability $q_t = Q_t/(mr)$ or $q_u = Q_u/(mr)$. Independence across masks gives $\mathbb{E}[M_{t,ij} M_{u,ij}] = q_t q_u$, and therefore

$$\mathbb{E}[O^{\text{mask}}_{t,u}] = \frac{1}{mr} \sum_{i,j} \mathbb{E}[M_{t,ij} M_{u,ij}] = q_t q_u. \tag{15}$$

For $q_t = q_u = 0.4$, the expected overlap is 16% of all mr coordinates. The calculation uses independence across task masks and allows dependencies within each mask, providing an analytical reference for the data-dependent supports selected by calibration.

The active-overlap bound holds for every input pair. Even when both tasks activate each channel with marginal probability k/r, their joint activation probability need not be $(k/r)^2$. The bound quantifies shared support in aligned coordinates; functional interactions also depend on the task-specific projection directions associated with those coordinates.

A.4 Conditional forward maps and structural diagnostics

With $C = M \odot B$ and $\lambda = \alpha/r$, the all-channel reference update and the input-conditioned update are

$$\overline{\Delta W} = \lambda C A, \qquad \Delta W(x) = \lambda C \operatorname{Diag}(g(x)) A. \tag{16}$$

The reference matrix has rank at most r. Once the routing decision is fixed, $\Delta W(x)$ contains at most k outer products $\lambda C_{:j} A_{j:}$ and has rank at most k. Across tokens, different selected channels can produce adapter outputs that span more than k dimensions.

These matrices describe the forward map, while backpropagation follows the stop-gradient rule. We examine the trained adapter using two complementary post-training diagnostics: effective rank measures the spread of the full reference spectrum, and gradient-subspace alignment measures the task relevance of the selected directions.

For a nonzero reference update $\overline{\Delta W}$ with singular values $\{\sigma_i\}$, let $p_i = \sigma_i / \sum_j \sigma_j$. The spectral-entropy effective rank is

$$\operatorname{erank}(\overline{\Delta W}) = \exp\left(-\sum_i p_i \log p_i \right), \tag{17}$$

where logarithms are natural and $0 \log 0 = 0$. For each seed, we average the effective ranks across adapted modules; Table 6 reports the mean of these five run-level values. We use $C = B$ for methods without an element mask. Effective rank measures how evenly the all-channel reference update distributes its singular-value mass across directions. It can approach r even though each conditional update has rank at most k.

B ADAPTER COMPOSITION AND COMPLETE MERGING RESULTS

Consider T task adapters with common rank r and scale $\lambda = \alpha/r$. Let $C_t = M_t \odot B_t$, or $C_t = B_t$ for an unmasked adapter. In this paper's column-vector convention, LoRI's static Concat construction (Zhang et al., 2025a) is

$$\Delta W_{\text{cat}} = \lambda [C_1 \ \cdots \ C_T] \begin{bmatrix} w_1 A_1 \\ \vdots \\ w_T A_T \end{bmatrix} = \lambda \sum_{t=1}^{T} w_t C_t A_t, \tag{18}$$

The merge coefficient w_t is distinct from λ, and each masked factor C_t remains paired with its own A_t. Linear instead combines factors separately:

$$\begin{aligned} \Delta W_{\text{lin}} &= \lambda \left(\sum_{t=1}^{T} w_t C_t \right) \left(\sum_{t=1}^{T} w_t A_t \right) \\ &= \lambda \sum_{t=1}^{T} w_t^2 C_t A_t + \lambda \sum_{s \neq t} w_s w_t C_s A_t. \end{aligned} \tag{19}$$

Linear changes the update in two ways: it introduces cross-task products and replaces each within-task coefficient w_t with w_t^2. Concat and Linear have rank at most Tr and r, respectively, and both allow arbitrary merge coefficients. A normalized average is the particular Concat choice $w_t = 1/T$. These static constructions act on all-channel reference updates; Equation 16 additionally accounts for input-dependent channel selection.

The concatenated factor width is $r_{\text{cat}} = Tr$, but Equation 18 retains the original scale $\lambda = \alpha/r$ once. If a factorized representation computes its scale as $\alpha_{\text{cat}}/r_{\text{cat}}$, representing this equation requires $\alpha_{\text{cat}} = T\alpha$, unless the original scale has already been absorbed into a factor. Keeping α unchanged while replacing r by Tr would shrink the update by an additional factor of T.

Table 7 places each merged result beside its single-task reference: Before reproduces the means in Table 2, and Linear and Concat give the merged scores, all averaged over five random seeds. For each metric, we compute $\Delta_{\text{best}} = \max(\bar{a}_{\text{Linear}}, \bar{a}_{\text{Concat}}) - \bar{a}_{\text{Before}}$, where each $\bar{a}$ is a mean across seeds. Strategy selection therefore follows seed averaging and is performed separately for each metric; different metrics can summarize different merging strategies.

Across two backbones, five methods, and six metrics, Concat has the higher mean score in 55 of 60 comparisons. The five exceptions all occur in HumanEval pass@5: FlyLoRA and AeroLoRA on Llama-3.1-8B, and FlyLoRA, LoRI-S, and AeroLoRA on Mistral-7B-v0.3. AeroLoRA has the highest mean merged scores throughout; Table 7 also reports the changes relative to each single-task reference.

Table 7: Complete adapter-merging results (%). Before, Linear, and Concat are means over five random seeds. $\Delta_{\text{best}} = \max(\bar{a}_{\text{Linear}}, \bar{a}_{\text{Concat}}) - \bar{a}_{\text{Before}}$ is computed from these means per metric, in percentage points. Red shading highlights AeroLoRA's change rows; bold marks the largest change per backbone and metric.

Model	Method	Merge status	MMLU	SciQA	GSM8K	HumanEval		
						pass@1	pass@5	pass@10
Llama-3.1-8B	LoRA ($r = 8$)	Before	37.3	90.5	54.9	30.1	53.8	63.5
		Linear	30.3	28.7	23.2	16.1	46.2	58.0
		Concat	31.8	32.3	26.7	18.3	47.6	58.8
		Δ_{best}	−5.5	−58.2	−28.2	−11.8	−6.2	−4.7
	LoRA ($r = 32$)	Before	39.4	92.3	57.1	34.7	58.5	69.1
		Linear	34.5	33.8	27.5	23.7	50.9	63.8
		Concat	35.1	37.1	30.2	24.9	51.7	64.6
		Δ_{best}	−4.3	−55.2	−26.9	−9.8	−6.8	−4.5
	FlyLoRA ($k = 8$)	Before	40.1	93.0	57.9	35.3	60.4	70.6
		Linear	37.5	46.3	33.7	30.1	54.5	66.2
		Concat	37.8	47.9	35.1	30.6	54.1	66.3
		Δ_{best}	−2.3	−45.1	−22.8	−4.7	−5.9	−4.3
	LoRI-S	Before	39.7	92.7	57.5	34.9	59.1	69.5
		Linear	37.3	46.5	33.9	29.8	53.4	65.3
		Concat	37.6	48.1	35.1	30.4	53.5	65.8
		Δ_{best}	−2.1	−44.6	−22.4	−4.5	−5.6	−3.7
	AeroLoRA	Before	41.6	94.1	58.5	36.8	61.7	72.2
		Linear	39.4	48.8	35.7	32.1	56.3	68.2
		Concat	39.7	50.1	36.8	32.6	55.9	68.7
		Δ_{best}	**−1.9**	**−44.0**	**−21.7**	**−4.2**	**−5.4**	**−3.5**
Mistral-7B-v0.3	LoRA ($r = 8$)	Before	35.4	72.9	58.7	31.9	43.1	49.5
		Linear	29.7	23.1	26.3	17.3	35.9	43.7
		Concat	30.1	25.4	28.4	19.2	36.6	44.4
		Δ_{best}	−5.3	−47.5	−30.3	−12.7	−6.5	−5.1
	LoRA ($r = 32$)	Before	37.2	78.5	61.3	35.4	47.7	52.3
		Linear	32.5	30.8	30.7	25.8	42.9	50.8
		Concat	33.1	33.7	32.8	27.1	43.6	51.5
		Δ_{best}	−4.1	−44.8	−28.5	−8.3	−4.1	**−0.8**
	FlyLoRA ($k = 8$)	Before	37.9	79.1	61.6	36.1	48.3	54.1
		Linear	35.1	40.3	37.8	32.3	45.1	52.5
		Concat	35.4	41.8	39.1	32.8	44.5	53.0
		Δ_{best}	−2.5	−37.3	−22.5	−3.3	−3.2	−1.1
	LoRI-S	Before	37.6	78.9	61.5	35.7	47.9	54.8
		Linear	35.0	40.8	38.2	32.2	44.9	52.4
		Concat	35.3	42.2	39.4	32.9	44.4	52.9
		Δ_{best}	−2.3	−36.7	−22.1	−2.8	**−3.0**	−1.9
	AeroLoRA	Before	39.3	80.6	62.9	36.5	49.8	55.6
		Linear	37.1	43.3	40.7	33.4	45.9	53.2
		Concat	37.3	44.8	41.3	33.8	45.4	53.9
		Δ_{best}	**−2.0**	**−35.8**	**−21.6**	**−2.7**	−3.9	−1.7

C ANALYTICAL SUPPORT BUDGET

Figure 4 shows the approximation $N_B(x)/(mr) \approx qk/r$ derived in Appendix A.3 for approximately uniform retained column counts, with q the realized retained fraction of B. The nominal default $q = 1 - s = 0.4$, $k = 8$, and $r = 32$ gives 10% of dense B support. The final mask and selected channels determine the exact count.

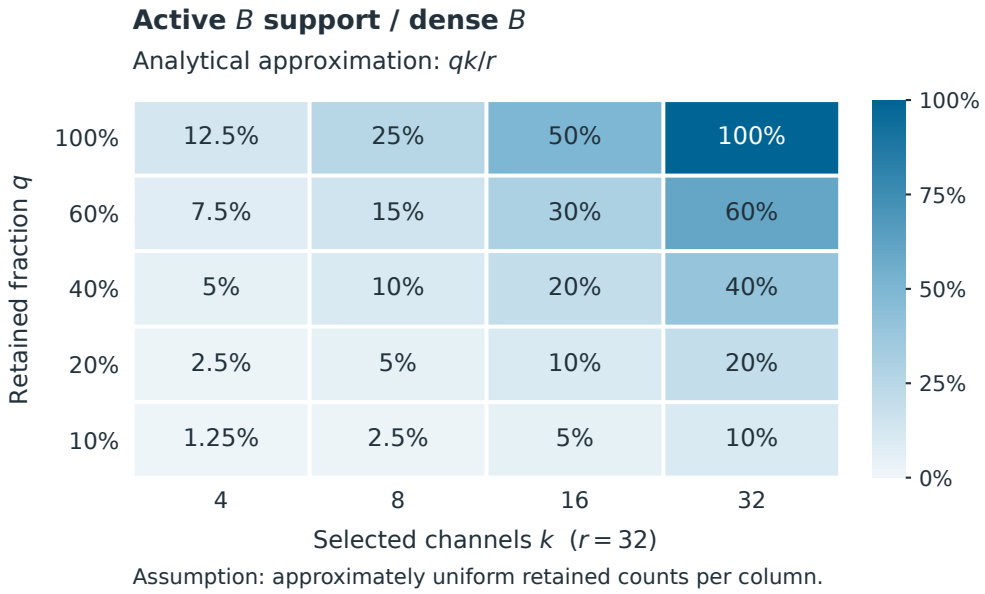


Figure 4: Analytical estimates of per-token active B support relative to all mr entries, for $r = 32$. Each cell gives qk/r under approximately uniform retained column counts, $d_j \approx qm$.