NOT HOW MANY, BUT WHICH: PARAMETER PLACEMENT IN LOW-RANK ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

LoRA introduces low-rank adapter parameters for fine-tuning. We study whether, given a fixed budget of k trainable entries in the $\mathbf{B}$ matrix of a LoRA adapter ($\mathbf{A}$ frozen), it matters which entries are trained. We call this the *parameter placement problem*: which k entries should be trained? Under supervised fine-tuning, random and informed subsets perform comparably in most settings. Under GRPO on base models, random placement fails to improve over the base model, whereas, on mathematical reasoning and on text-to-SQL, gradient-informed placement recovers standard LoRA accuracy. The gradient structure explains this behavior: SFT gradients are low-rank and directionally stable, so any subset accumulates coherent updates; GRPO gradients are higher-rank and less aligned across steps, so only elements with consistently signed gradients retain the learning signal. Our scoring procedure identifies these parameters in under 10 seconds at less than 0.5% of training cost. The selected parameters are concentrated in the attention value projection matrices across model families and scales (1.5B–7B).

1 INTRODUCTION

Adapting large language models to new tasks is costly: full fine-tuning updates billions of parameters and requires substantial compute. Parameter-efficient fine-tuning (PEFT) methods address this by training a smaller set of parameters while freezing the rest. Among these, Low-Rank Adaptation (LoRA; Hu et al., 2022) has become widely adopted, decomposing weight updates into low-rank matrices $\mathbf{BA}$ and optimizing only the adapter parameters.

Recent work shows that the number of trainable parameters in such adapters can be dramatically reduced, via shared random bases (Kopiczko et al., 2024), spectral parameterizations (Gao et al., 2024b), or low-dimensional projections (Morris et al., 2026). These results suggest that strong performance can be achieved with small parameter budgets. However, all of them fix which parameters are trained by construction and vary only how many. We revisit this choice:

Our Research Question

We study what we term the **Parameter Placement Problem**: **given a fixed budget of trainable parameters within a LoRA adapter, how does parameter choice affect downstream performance?** Concretely, in a standard LoRA layer $\mathbf{W}_0\mathbf{x} + \frac{\alpha}{r}\mathbf{BAx}$, where $\mathbf{B} \in \mathbb{R}^{h\times r}$ is initialized to zero and $\mathbf{A} \in \mathbb{R}^{r\times d}$ is drawn randomly, we keep $\mathbf{A}$ frozen and restrict training to a subset of k entries of $\mathbf{B}$. We call such a subset a **circuit**: a set of individual matrix entries. This isolates parameter selection from parameter count.

Donoway et al. (2025) show that randomly selected subsets of LoRA parameters already capture much of the performance gain under supervised fine-tuning (SFT) with the optimizer tuned separately at each budget. We ask whether a good subset can be identified up front, before training.

Two measurements explain when placement matters. Under SFT, gradients are low-dimensional structure and highly aligned across training steps, so most subsets receive a coherent update direction. Under GRPO, gradients are directionally noisy and much less aligned across steps, while their per-step magnitude is comparable to or larger than SFT; subsets therefore can cancel over training. Only coordinates whose gradients are consistently signed accumulate meaningful displacement, which is

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.

what makes placement decisive. This signal-structure account complements information-theoretic arguments that policy-gradient methods absorb far less information per episode than supervised learning, and so need little adapter capacity (Schulman & Lab, 2025).

Motivated by these observations, we propose a lightweight selection procedure: we score each entry of $\mathbf{B}$ at initialization, either by the magnitude of its mean gradient, which rewards directional consistency, or by its mean squared gradient. Scoring adds negligible overhead and yields subsets that are stable across data samples. To isolate the role of gradient noise, we also study a synthetic setting where the signal-to-noise ratio can be set directly, and it reproduces the same dichotomy.

Our contributions are as follows:

- We formalize the parameter placement problem for LoRA, separating effect of parameter count from parameter identity, and identify gradient structure that decides when placement matters (§3).
- We propose a lightweight gradient-based selection procedure that identifies effective parameter subsets in seconds and matches full LoRA at extreme parameter budgets (§4).
- We locate these parameters, test them causally by knockout, and track their training dynamics (§5).

2 Related Work

Low-rank adaptation and its variants. LoRA (Hu et al., 2022) decomposes weight updates into low-rank matrices BA, training only the adapter while freezing the pre-trained model. Subsequent work seeks to improve LoRA along several axes: initialization (Meng et al., 2024; Yang et al., 2024b), training dynamics (Hayou et al., 2024; Liu et al., 2024; Kalajdzievski, 2023), and rank allocation (Zhang et al., 2023; Valipour et al., 2023; Ding et al., 2023). Instead of modifying LoRA, we study the complementary question: given a fixed adapter, *which elements* should receive gradient updates.

Parameter budget and fixed-basis methods. Several methods reduce adapter size by fixing most parameters and training a small residual. VeRA (Kopiczko et al., 2024) shares frozen random A and B across all layers and trains per-layer scaling vectors. FourierFT (Gao et al., 2024b) represents updates via learnable spectral coefficients over a fixed Fourier basis. RandLoRA (Albert et al., 2025) extends VeRA with element-wise learned scaling to recover full-rank expressiveness. LoRA-XS (Bałazy et al., 2024) inserts a tiny $r \times r$ trainable matrix between frozen SVD-initialized factors. Most recently, Morris et al. (2026) show that RL can succeed with as few as 13 trainable parameters on *instruction-tuned* models, using random low-dimensional projections.

These methods share a common structure: fix a basis (random, Fourier, SVD) and learn a small set of coefficients. Our approach differs in two respects. First, we do not change the LoRA parameterization: we use standard $\mathbf{BA}$ with a standard optimizer and simply mask which elements of $\mathbf{B}$ receive gradients. Second, our selection is informed: we use gradient statistics from the training distribution to choose elements, rather than relying on a fixed or random basis.

Importance scoring and parameter selection. Using gradient or curvature information to identify important parameters has a long history, from Optimal Brain Damage (LeCun et al., 1989) and Optimal Brain Surgeon (Hassibi et al., 1993) to recent methods like Wanda (Sun et al., 2024) and SparseGPT (Frantar & Alistarh, 2023). SPT (He et al., 2023) uses sensitivity scores to decide where to place adapters; we assume adapters are placed and select which elements within them to train. Our circuit scores are instances of this family, specialized to the LoRA setting where $\mathbf{B} = \mathbf{0}$ at initialization provides a clean starting point for gradient-based scoring.

Elicitation and minimal adaptation. Aghajanyan et al. (2021) show that fine-tuning has low intrinsic dimensionality: randomly projected subspaces of a few hundred dimensions suffice. Donoway et al. (2025) report that training as few as 10–100 randomly chosen parameters can recover 50% of the performance gap, framing adaptation as elicitation of latent capabilities, though which parameters achieve this varies with the seed. Both studies select parameters at random rather than by an informed criterion. Our results are consistent with this picture for SFT, where random placement remains a reasonable baseline. Under RL on base models, however, random placement does not improve over the base model, so elicitation via RL appears to require informed selection.

Reinforcement learning for reasoning. GRPO (Shao et al., 2024) enables efficient RL fine-tuning and DeepSeek-R1 (Guo et al., 2025) demonstrates that RL can elicit reasoning. Morris et al. (2026) show that RL succeeds with fewer parameters than SFT on instruction-tuned models. In concurrent work, Zhu et al. (2026) trace the SFT/RL difference to gradient structure, norm-limited interference under SFT versus variance-limited under RL. Our work complements these findings: on base models, RL with sparse adapters requires informed parameter placement, since random placement, which suffices for SFT, fails under GRPO.

Additional background is in §A.

3 METHOD

We formalize parameter placement problem (§3.1), describe a lightweight scoring method to select entries of $\mathbf{B}$ (§3.2), and analyze when placement matters, and validate in a controlled setting (§3.3).

3.1 PROBLEM STATEMENT

Consider a model with LoRA-adapted layers $\{\mathbf{B}^{(\ell)}, \mathbf{A}^{(\ell)}\}_\ell$, where each layer outputs $\mathbf{y}^{(\ell)} = \mathbf{W}_0^{(\ell)}\mathbf{x}^{(\ell)} + \frac{\alpha}{r}\mathbf{B}^{(\ell)}\mathbf{A}^{(\ell)}\mathbf{x}^{(\ell)}$. We fix $\mathbf{A}^{(\ell)}$ and initialize $\mathbf{B}^{(\ell)} = 0$. We define the **parameter placement problem** under a global budget: given k trainable parameters, select per-layer masks $\mathbf{m}^{(\ell)} \in \{0,1\}^{h_\ell \times r_\ell}$ such that $\sum_\ell \|\mathbf{m}^{(\ell)}\|_0 = k$, and training only the selected entries $\{\mathbf{m}^{(\ell)} \odot \mathbf{B}^{(\ell)}\}_\ell$ maximizes performance. We consider two strategies: (1) **Random placement**, where k entries are sampled uniformly across all layers, and (2) **Informed placement**, where a scoring function scores each parameter across all layers and the top-k global entries are selected.

Freezing $\mathbf{A}$ and training only $\mathbf{B}$ is a well-studied LoRA variant (Zhang et al., 2024a; Zhu et al., 2024), including under RLVR (Yin et al., 2026). Here it is also natural: with $\mathbf{B} = \mathbf{0}$ at initialization, $\mathbf{BAx} = \mathbf{0}$ for any $\mathbf{A}$, so $\mathbf{A}$ receives no gradient on the first step, and $\mathbf{B}$-only training dominates joint training at matched budgets (§C.1).

3.2 CIRCUIT DISCOVERY

We need a method to score the entries most useful for learning within the $\mathbf{B} \in \mathbb{R}^{h \times r}$ matrices; given budget k, the **circuit** is the set of k elements with the highest scores. Each element is scored via gradient statistics from N forward-backward passes at initialization, with $N = 50$ in all experiments. With $\mathbf{B} = \mathbf{0}$ the model's predictions are identical to the pre-trained model, and therefore the gradient $\nabla_{\mathbf{B}}\mathcal{L}$ measures each element's utility for reducing the loss. For each example x_n, let $g_{n,ij} = [\nabla_{\mathbf{B}}\mathcal{L}(x_n)]_{ij}$ be the per-element gradient. We define two scores:

$$\hat{S}_{ij} = \left| \frac{1}{N} \sum_{n=1}^{N} g_{n,ij} \right|, \qquad (1) \qquad\qquad \hat{F}_{ij} = \frac{1}{N} \sum_{n=1}^{N} g_{n,ij}^2. \qquad (2)$$

The **Gradient magnitude score** $\hat{S}_{ij}$ (left) estimates $|\mathbb{E}[g_{ij}]|$, the magnitude of the population mean gradient: elements with high $\hat{S}$ have gradients that are directionally consistent across examples, i.e. the loss surface has a persistent preference for how that element should change.

The **Diagonal Fisher score** $\hat{F}_{ij}$(right) estimates the diagonal of the empirical Fisher $\mathbb{E}[g_{ij}^2]$: elements where the loss is most sensitive to perturbations.

The two scores are related by the bias-variance decomposition $\mathbb{E}[g_{ij}^2] = \mu_{ij}^2 + \sigma_{ij}^2$; Fisher captures both mean and variance, while $\hat{S}$ captures only the mean. The distinction matters: an element can have high $\hat{F}$ but low $\hat{S}$ if its gradients are large but flip sign across examples. Both estimators converge at rate $O(1/\sqrt{N})$, and because score distributions are heavy-tailed (top elements have scores orders of magnitude above the median), small N suffices (see §4.1).

Table 1: **Gradient structure under SFT vs. GRPO** (Qwen2.5 on MATH, matched gradient steps at initialization). Across all three scales, SFT gradients are low-rank, directionally stable, and accumulate coherently. GRPO gradients are higher-rank, near-orthogonal across steps, and mostly cancel; only informed placement recovers the signal. GSM8K results in §D.6.

	Eff. Gradient Rank		Cosine Similarity		Net / Step Norm	
Model	**SFT**	**GRPO**	**SFT**	**GRPO**	**SFT**	**GRPO**
Qwen2.5-1.5B	2.6	21.8	0.76	0.04	0.90	0.28
Qwen2.5-3B	5.8	7.7	0.61	−0.03	0.81	0.22
Qwen2.5-7B	2.0	8.8	0.85	−0.03	0.93	0.18

3.3 When Does Placement Matter?

Circuit discovery selects a subset of parameters, but this only helps if those parameters capture a large share of the gradient at each step and keep receiving it in a consistent direction over training.

Signal retention under sparse masking. Let $g \in \mathbb{R}^d$ be the gradient over all $\mathbf{B}$ elements (flattened), and let $m \in \{0,1\}^d$ be a mask with $\|m\|_0 = k$. The effective update is $m \odot g$, retaining only the masked entries. The fraction of gradient energy retained is: $\rho(m, g) = \|m \odot g\|^2 / \|g\|^2$.

For a random mask: $\mathbb{E}[\rho] = k/d$. For an informed mask selecting the top-k elements by $|g_i|$: $\rho = \sum_{i \in \text{top-k}} g_i^2 / \|g\|^2$, which can be much larger for heavy-tailed gradients. Empirically, the top 0.05% of the entries of $\mathbf{B}$ capture ∼8–15% of gradient energy, up to 300× amplification over random.

Informed placement achieves higher signal retention for SFT and GRPO, but random placement only succeeds for SFT (§4). The missing factor is how updates accumulate across training steps.

Gradient stability determines accumulation. Over T steps, $\Delta w_i = \eta \sum_t g_{t,i}$. When gradients align, updates accumulate coherently; when they are noisy, updates largely cancel. Under low alignment, then, only the few parameters with consistently signed gradients build up meaningful displacement (Table 16), while most stay near zero. Table 1 measures three properties of the $\mathbf{B}$ gradients $\mathbf{g}_1, \ldots, \mathbf{g}_N$ over matched steps at initialization. Let σ_i be the singular values of the matrix whose rows are those gradients, normalized to $\hat{\sigma}_i = \sigma_i^2 / \sum_j \sigma_j^2$. Effective rank, $\exp(-\sum_i \hat{\sigma}_i \log \hat{\sigma}_i)$, is the number of directions the steps span. Cosine similarity, $\frac{1}{N-1} \sum_t \langle \mathbf{g}_t, \mathbf{g}_{t+1} \rangle / (\|\mathbf{g}_t\| \|\mathbf{g}_{t+1}\|)$, is the alignment of consecutive steps. Net / Step Norm, $\|\sum_t \mathbf{g}_t\| / \sum_t \|\mathbf{g}_t\|$, is 1 when every step points the same way and about $1/\sqrt{N}$ over N unrelated steps.

Under SFT the gradient is low-rank and consecutive steps are strongly aligned, so the mean gradient retains most of the average per-step norm. Any k-element subset projects onto the dominant direction with high probability, and placement has little effect. Under GRPO the gradient spreads over more directions and consecutive steps are essentially uncorrelated, leaving far less. This is not a matter of small gradients: GRPO per-step norms are comparable to or larger than SFT's, yet the accumulated update is smaller at every scale (Table 18), so the steps must be cancelling.

Synthetic validation. We check this prediction by setting the signal: a 2-layer MLP with LoRA, trained toward either a dense target (rank-2 residual, SFT-like) or a sparse target in which only 5% of entries are large (RL-like). $\mathbf{A}$ is frozen for both placements, while full LoRA trains $\mathbf{A}$ and $\mathbf{B}$ (§C.6). Fig. 1 matches the prediction: under dense signal, random and informed placement are indistinguishable; under concentrated signal, informed placement at 2% of $\mathbf{B}$ nearly matches full LoRA while random placement lags at 50%. We test the mechanism on real LLMs next (§4).

4 Experiments

We first show that circuit discovery is stable and cheap (§4.1), then evaluate sparse circuits under supervised fine-tuning (§4.2) and under reinforcement learning (§4.3), where placement matters most, and finally ask whether the resulting circuits generalize out of distribution (§4.4).

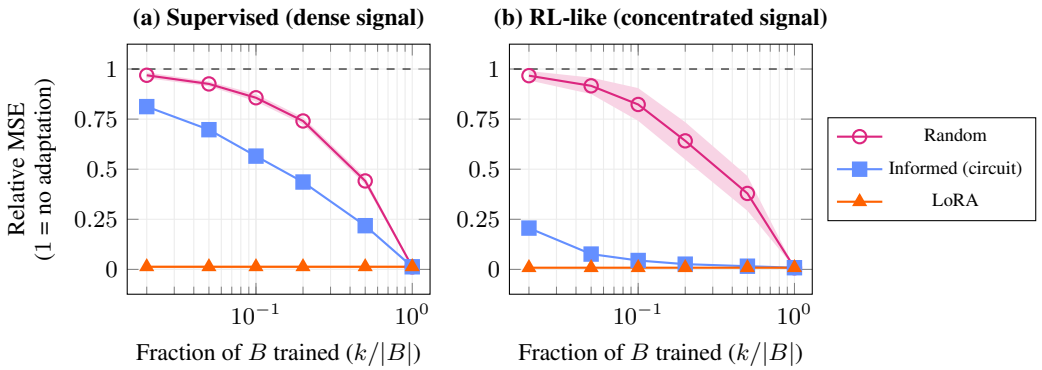

Figure 1: **Synthetic validation.** (a) Under dense signal, random and informed placement perform similarly, since any parameter subset captures the distributed gradient. (b) Under concentrated signal, informed placement at 2% of $\mathbf{B}$ nearly matches Full LoRA, while random placement barely improves even at 50%.

We evaluate on three Qwen2.5 base models (1.5B, 3B, 7B) (Team, 2024) and on four instruction-tuned models: Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Team, 2024), and Qwen2.5-VL-7B-Instruct (Team, 2025). LoRA is applied to all linear layers. We compare two gradient-based selection methods, $\hat{S}$ (gradient magnitude) and $\hat{F}$ (diagonal Fisher), against random placement, under both SFT and GRPO. The instruct models test whether placement still matters when the model already answers correctly often enough for the reward to be dense. Because only k entries of $\mathbf{B}$ are trainable, the adapter norm is smaller than full LoRA's at the same rank. We therefore scale α to $32r$ for every sparse run, informed and random alike, while full LoRA uses $\alpha = 32$ (Table 26). Random placement fails across a $128\times$ sweep of α, so this choice does not drive the results (§C.8). Full hyperparameters are in §E.

4.1 Circuit Discovery Validation

Before evaluating downstream performance, we verify that the discovery procedure produces stable circuits. We report results on Qwen2.5-1.5B and 3B / GSM8K with 10K-element circuits. For reference, two random 10K-element subsets of $\mathbf{B}$ overlap by $< 0.1\%$ in expectation, so any substantial overlap reflects structure in the scores (details in §C.2 and §C.3).

Stability and cost. Circuit discovery is stable to sampling. Using as few as $N{=}10$ examples recovers $> 90\%$ of the top-10K elements of the $N{=}100$ reference circuit, reaching $\approx 98\%$ at $N{=}50$ across scales. Changing the $\mathbf{A}$ initialization moves the exact elements selected, but not the modules and layers they sit in (§C.5). Discovery is inexpensive, requiring N forward-backward passes with no optimizer state and completing in < 10s on a single GPU ($< 0.5\%$ of SFT cost). The resulting circuit is highly compact (under 40 KB for 10K elements, $\sim 4000\times$ smaller than LoRA)(Table 8). Scores come from the supervised cross-entropy gradient, including for the GRPO runs without requiring any rollouts or reward evaluations, which is what keeps discovery cheap. Scoring with the RL objectives like GRPO, Dr.GRPO picks similar circuits, concentrated on the same modules (§D.7).

4.2 Supervised Fine-Tuning

We evaluate circuit placement under SFT, where gradients are concentrated and stable (§3.3).

We fine-tune on Alpaca (Taori et al., 2023) and Bespoke-Stratos (Labs, 2025), evaluating on instruction following benchmarks (7-task average; §B.1) and GPQADiamond (Rein et al., 2024). We also test the vision-language setting by fine-tuning Qwen2.5-VL-7B-Instruct (Team, 2025) on MathV360K (Shi et al., 2024) and evaluating on MathVista (Lu et al., 2024).

Under SFT, differences between methods are small across most settings, indicating that arbitrary placements already capture useful gradient signal. On MathVista, all methods converge to similar accuracy, with $\hat{S}$ at $k{=}10$K (67.8%) slightly ahead of Full LoRA (66.9%, Table 10).

Table 2: **7-benchmark average accuracy (%) after SFT on Alpaca** over 5 seeds. Informed placement provides a consistent edge over random, though the gap is modest under SFT's stable gradients. Per-benchmark results in §B.1; budget ablations in §C.7.

Model	Pretrained	Random	$\hat{S}$	$\hat{F}$	Full LoRA	k
SFT on Alpaca (Instruction Following); Accuracy over 7-benchmarks.						
Qwen2.5-1.5B	53.0	61.6	61.9	62.1	**62.4**	50K
Qwen2.5-3B	65.6	65.8	66.2	66.0	**66.7**	50K
Qwen2.5-7B	70.9	71.1	71.4	**71.5**	**71.5**	100K
Llama-3.2-3B-Inst	62.3	62.9	63.8	63.7	**63.9**	50K
Llama-3.1-8B-Inst	68.7	69.1	69.6	69.7	**69.9**	100K
SFT on Bespoke-Stratos (Reasoning); Accuracy on GPQADiamond.						
Qwen2.5-7B-Inst	31.6	37.4	**38.9**	**38.9**	36.6	50K

Table 3: **GRPO accuracy (%) on GSM8K and MATH-500.** Random placement fails for base models (Random ≈ Pretrained) while circuit placement recovers full LoRA accuracy (over 3 random seeds). $^\star k$=10K; otherwise k=50K. Additional ablations in §C.7 and §C.8.

			Circuit (k params)		
Model	Pretrained	Random	$\hat{S}$	$\hat{F}$	Full LoRA
GSM8K					
Qwen2.5-1.5B*	9.5	$11.6_{\pm 3.1}$	$\mathbf{65.2}_{\pm 1.7}$	$63.8_{\pm 2.6}$	$63.7_{\pm 1.8}$
Qwen2.5-3B*	62.1	$63.5_{\pm 1.7}$	$70.8_{\pm 0.2}$	$\mathbf{72.4}_{\pm 0.8}$	$71.5_{\pm 2.0}$
Qwen2.5-7B	79.3	$80.9_{\pm 1.2}$	$86.9_{\pm 1.0}$	$87.2_{\pm 1.7}$	$\mathbf{87.4}_{\pm 1.6}$
Llama-3.2-3B-Inst	70.4	$69.5_{\pm 1.9}$	$74.9_{\pm 1.6}$	$\mathbf{75.5}_{\pm 1.2}$	$75.2_{\pm 1.9}$
Llama-3.1-8B-Inst	82.9	$82.4_{\pm 2.7}$	$\mathbf{86.4}_{\pm 1.8}$	$85.2_{\pm 2.2}$	$86.1_{\pm 2.4}$
MATH-500					
Qwen2.5-1.5B*	17.4	$19.1_{\pm 1.3}$	$\mathbf{43.8}_{\pm 0.9}$	$42.5_{\pm 1.1}$	$43.5_{\pm 1.7}$
Qwen2.5-3B	50.6	$52.2_{\pm 1.4}$	$62.2_{\pm 0.8}$	$63.4_{\pm 0.9}$	$\mathbf{63.5}_{\pm 0.5}$
Qwen2.5-7B	53.6	$57.4_{\pm 0.9}$	$74.2_{\pm 0.9}$	$\mathbf{75.3}_{\pm 0.7}$	$75.1_{\pm 0.6}$
Llama-3.2-3B-Inst	43.6	$47.1_{\pm 2.3}$	$50.4_{\pm 1.5}$	$51.2_{\pm 1.8}$	$\mathbf{51.3}_{\pm 2.4}$
Llama-3.1-8B-Inst	50.4	$52.3_{\pm 2.6}$	$57.2_{\pm 1.5}$	$\mathbf{57.5}_{\pm 2.7}$	$56.6_{\pm 1.9}$

4.3 Reinforcement Learning (GRPO)

In contrast to SFT, placement under GRPO has a drastic effect. We compare circuit methods ($\hat{S}$, $\hat{F}$) to random placement and full LoRA under GRPO (Table 3). For all Qwen models, random placement stays at base level and is insensitive to budget or scaling (Table 13 in §C.8). Both $\hat{S}$ and $\hat{F}$ at the same budget (k=10K–50K, under 0.5% of full LoRA's trainable parameters) are comparable to full LoRA at every scale, exceeding it in several settings. Both instruction-tuned Llama models sit outside this regime: on MATH-500 random placement also gains, which is what prior work reports for RL on instruction-tuned models (Morris et al., 2026). The same pattern holds outside mathematics. Under an execution reward on text-to-SQL, informed placement beats random on all four Qwen2.5-7B variants (base, instruction-tuned, code-specialized, and both) on both Spider and BIRD, and tracks full LoRA within half a point everywhere, exceeding it in half the configurations (§F.4). The gap to random is widest on the base model and narrows as the model moves closer to the domain.

GRPO gradients are less aligned across steps than SFT's (Tables 1 and 17), so less of each step survives accumulation; random placement samples these directions and fails. $\hat{S}$ selects coordinates with consistent gradient sign, recovering the small coherent component; circuit elements move 1.4–2.1× more than random under GRPO (Table 16). $\hat{F}$ weights by expected gradient magnitude rather than sign alone; both methods concentrate on the same modules (Tables 4 and 14), which may account for their similar performance despite distinct selection criteria.

4.4 Out-of-Distribution Generalization

Does informed placement generalize beyond the training distribution? We evaluate our Qwen2.5-3B models, RL trained on MATH and on GSM8K, on held-out benchmarks: GSM8K (cross-task

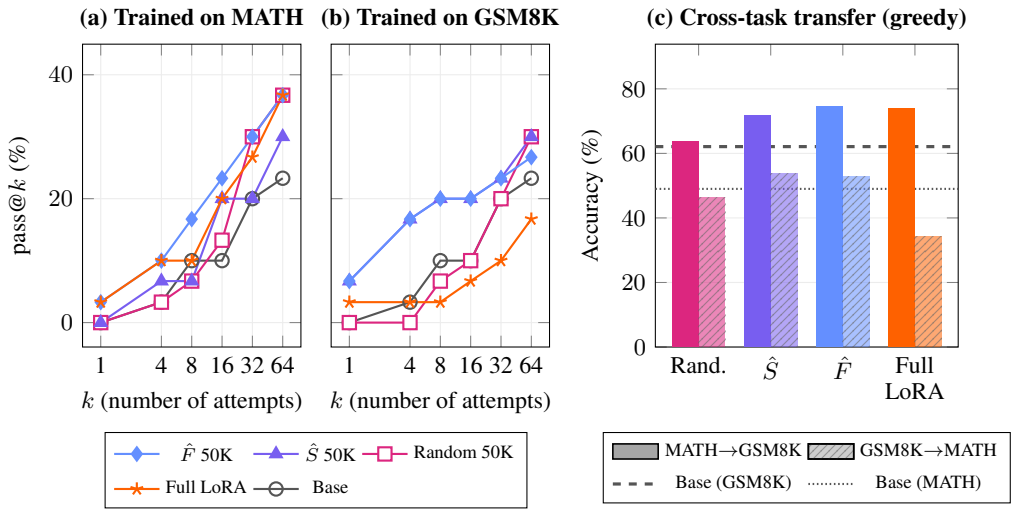

Figure 2: **OOD generalization** (Qwen2.5-3B, GRPO). (a, b) AIME 2024 pass@k for models trained on MATH and GSM8K respectively. (c) Cross-task greedy transfer: solid bars = MATH→GSM8K, hatched = GSM8K→MATH; dashed/dotted lines show base-model accuracy. $\hat{F}$ at <0.5% of full LoRA's trainable parameters matches or exceeds Full LoRA across all settings.

transfer), AMC 2023 (40 competition problems) (Mathematical Association of America, 2023), and AIME 2024 (30 problems) (Jia, 2024) (additional details in §B.2).

There is some positive transfer for all tasks with the circuit based methods. Trained on MATH, $\hat{F}$ at 50K parameters matches Full LoRA when transferred to GSM8K and exceeds it on AMC, where it solves every problem at pass@64, while random placement barely improves over the base model. Circuits trained on GSM8K behave the same way, outperforming Full LoRA on AIME 2024 (§B.2). The transfer on the models for MATH and GSM8K, shown in Fig. 2c, is positive for all methods between GSM8K and MATH. But for the reverse direction, it is only positive (above base model performance) for the circuit-based methods. This positive transfer aligns with the shared circuit structure: Circuits discovered on different math datasets share $\sim 30\%$ of their elements at k=50K, over $100\times$ above chance, so placement is not dataset-specific (§D.3).

5 Analysis

We analyze where the selected parameters sit (§5.1), whether they are causally responsible for the adapter's effect, and how they change the model's behavior over training (§5.2).

5.1 Anatomy of Selected Circuits

The module-level composition of the $\hat{S}$ circuits is structured, disproportionate to the number of parameters per module. Table 4 reports the module-level composition of $\hat{S}$ circuits (top 10K entries) across all four configurations. V is overrepresented in every setting relative to its share of parameters. At 7B scale, O absorbs most of the remaining circuit budget, while at 3B, the budget is more distributed across Down and the MLP modules. Gate and Up hold the majority of $\mathbf{B}$ parameters but capture a minority of circuit entries. $\hat{F}$ circuits show the same ordering with budget shifting slightly from Gate/Up towards V and O. Both $\hat{S}$ and $\hat{F}$ circuits concentrate in early layers, increasingly so at larger scale: at 7B over 90% of the budget falls in layers 0–4, while at 3B it spreads deeper. Additional details are presented in §D.1 and §D.2. Restricting dense LoRA to the two most overrepresented modules matches full LoRA in every configuration we tested (§F.2).

This concentration is not arbitrary: gradient-consistency and SVD-alignment analyses (§D.9) reveal that selected modules fall into two functional classes: **attention readers** (V, K) that align with

Table 4: $\hat{S}$**-circuit composition** (top 10K entries, % of circuit). "Param" shows each module type's share of all B parameters. V is overrepresented in all configurations; O dominates at 7B. Gate and Up are consistently underrepresented despite holding the majority of parameters.

		% of circuit (top 10K)			
Module	**Param %**	**3B MATH**	**3B GSM8K**	**7B MATH**	**7B GSM8K**
Q	7.1	2.9	2.2	0.5	0.6
K	0.9	6.1	8.5	1.9	2.4
V	0.9	23.6	29.2	21.3	39.2
O	7.1	11.1	2.1	59.9	38.2
Gate	38.4	17.3	27.8	7.7	8.2
Up	38.4	19.8	16.4	8.5	11.1
Down	7.1	19.1	13.9	0.1	0.2

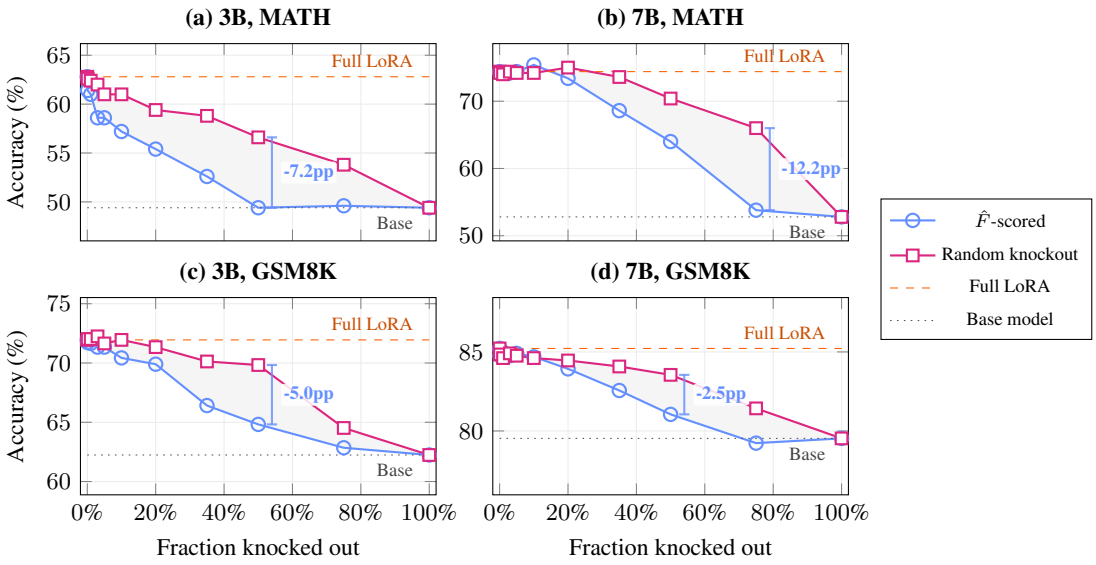

Figure 3: $\hat{F}$ **knockout sweep** (3B/7B × MATH/GSM8K). Accuracy vs. fraction of B entries zeroed by $\hat{F}$ score (circuit, red) or at random (blue). Dashed: full LoRA; dotted: base model. Circuit-ordered knockout degrades faster in all configurations, with the gap scaling with LoRA effect size.

the pretrained weight's dominant singular vectors, and **residual writers** (O, Down) that produce spectrally concentrated updates, both at the interface of the residual stream.

Knockout validation. Our results show that the selected elements form circuits that learn effectively. Here we answer whether the elements matter after the initial stages of training using a training-free knockout sweep. Starting from a fully trained LoRA, we zero out the top fraction of B entries by $\hat{F}$ score and measure accuracy after merging the residual adapter into the base model. If $\hat{F}$ identifies critical parameters, circuit-ordered knockout should degrade accuracy faster than removing entries at random. These findings further suggest that our selection process selects generically useful elements, not elements only useful early in training.

Zeroing $\hat{F}$-ranked entries degrades accuracy faster than random removal in all four configurations (Fig. 3), causally confirming that scored parameters mediate the adapter's effect. The gap scales with the adapter's total contribution: it is largest where full LoRA gains most over the base model (7B MATH) and smallest where it gains least (7B GSM8K). $\hat{S}$-scored knockout shows a similar qualitative pattern (§D.8).

5.2 Weight Norm and Behavior Change Dynamics

In Fig. 4, we show how selected parameters and model predictions change. Informed placement circuits have similar weight change to random placement circuits, but more behavioral change. Al-

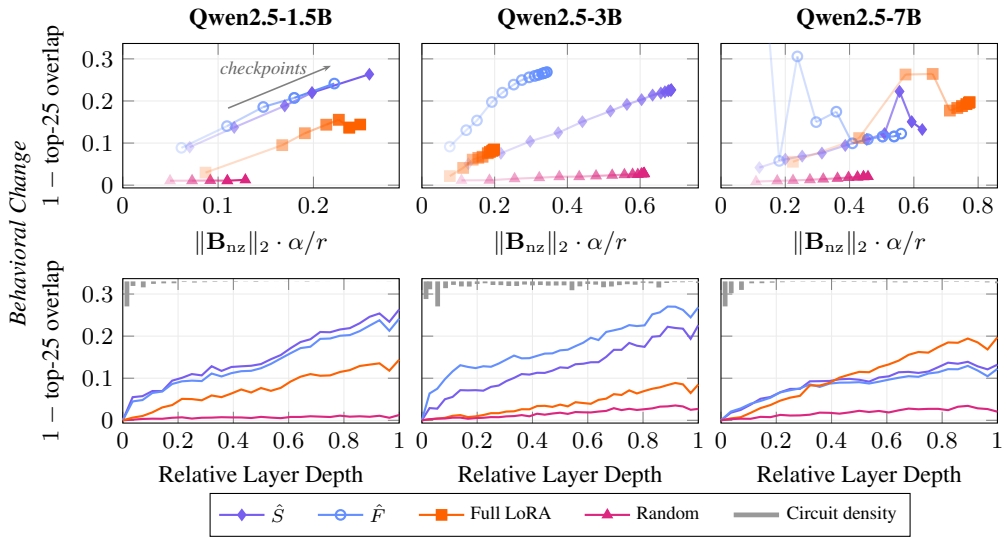

Figure 4: **Behavioral change vs. effective update norm** (Qwen2.5 MATH-500, GRPO). For both rows, the y-axis is $1 - $ *top-25 token overlap*, measuring behavioral change via logit lens relative to the base model. Top: each trajectory traces training checkpoints, opacity increases with step. x-axis: $|\mathbf{B}_{\text{nz}}|_2 \cdot \alpha/r$, the scaled norm of non-zero $\mathbf{B}$ weights. Bottom: per-layer behavioral change at the final checkpoint. x-axis: relative layer depth. Gray bars report the relative circuit density.

though most of the selected elements are in early layers, the token-level behavior change accumulates most in the later layers. Both of these findings highlight the leverage of the selected parameters.

Methodology. We measure the dynamics of Qwen2.5 (GRPO on MATH as compared to the respective base models) over 100 test examples. We report the *behavioral change* at the token level between the base and fine-tuned models, computing the $1-$ *top-25 overlap*, where overlap is the Jaccard overlap of the top-25 predicted tokens, averaged over positions and examples including only the prompts and problem text. Results using Llama models, or KL instead of Jaccard, are similar (§D.11).

Fig. 4 (top) shows the behavioral change over training checkpoints. The x-axis reports the effective norm of the LoRA trainable weights, $\|\mathbf{B}_{\text{nz}}\|_2 \cdot \alpha/r$. Since $\mathbf{B} = 0$ at initialization, the non-zero $\mathbf{B}_{\text{nz}}$ weights are the total displacement. Random circuits serve as a baseline: despite similar levels of weight change, behavioral change remains near zero. By contrast, the informed circuits shift the output distribution substantially more at the same effective weight change.

Fig. 4 (bottom) shows the behavioral change of vocabulary projections across model layers (x-axis) at the final checkpoint using logit lens (Belrose et al., 2023). Most selected parameters sit in the lower layers (Fig. 12; gray bars report circuit density), yet change is highest in later layers. This complements Meng et al. (2022), who found that early-layer edits were often corrected downstream, whereas training produces updates that compound through depth.

6 Conclusion

We studied the parameter placement problem in LoRA: given a fixed budget of trainable parameters, does it matter which are trained? The answer depends on the training regime. Under SFT, gradients are concentrated and directionally stable, so arbitrary subsets accumulate coherent updates and placement has limited effect. Under GRPO on base models, gradients are distributed across many dimensions and much less aligned across steps, so little of each step survives accumulation. A lightweight procedure, 50 forward-backward passes at initialization, identifies a sparse subset of $\mathbf{B}$ elements with consistently signed gradients. At under 0.5% of full LoRA's trainable parameters, these circuits recover full LoRA performance under GRPO on Qwen2.5 base models from 1.5B to 7B, with smaller gains on instruction-tuned models at 3B and 8B. The same ordering holds outside mathematics, under an execution reward on text-to-SQL.

AI USE STATEMENT

In this work, we used generative AI tools for code assistance and running experiments, generating the plots and for grammar correction. We have not used generative AI tools for idea generation or designing the experiments and the general writing of the paper. We have manually vetted and reviewed all the code generated by AI as well as made sure that the plots faithfully and accurate describe our results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work is about understanding the widely used LoRA adaptation and so we do not believe that there are any ethical concerns in our work.

REPRODUCIBILITY STATEMENT

We have detailed all hyperparameters, experimental setting and the hardware required to run the experiments in § E. Our method is clearly described in § 3 (equation 1 and equation 2). We will open source the code upon acceptance of the manuscript.

REFERENCES

F.T. Adilova and Samariddin Kushmuratov. OLoRA+: A hybrid approach to parameter-efficient fine-tuning of large language models. In Conference of Mathematics of AI, 2026. URL https://openreview.net/forum?id=c75JefyklT.

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), 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. 7319–7328, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568/.

Paul Albert, Frederic Z. Zhang, Hemanth Saratchandran, Cristian Rodriguez-Opazo, Anton van den Hengel, and Ehsan Abbasnejad. RandloRA: Full rank parameter-efficient fine-tuning of large models. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=Hn5eoTunHN.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. LoRA-XS: Low-rank adaptation with extremely small number of parameters, 2024. URL https://openreview.net/forum?id=l80AgHoRaN.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. arXiv preprint arXiv:2303.08112, 2023.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 1–9, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL https://aclanthology.org/2022.acl-short.1/.

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. ISSN 2835-8856. URL https://openreview.net/forum?id=aloEru2qCG. Featured Certification.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020.

Kerim Büyükakyüz. Olora: Orthonormal low-rank adaptation of large language models. arXiv preprint arXiv:2406.01775, 2024.

Nick Cammarata, Gabriel Goh, Shan Carter, Chelsea Voss, Ludwig Schubert, and Chris Olah. Curve circuits. Distill, 2021. doi: 10.23915/distill.00024.006. https://distill.pub/2020/circuits/curve-circuits.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), Advances in Neural Information Processing Systems, volume 33, pp. 15834–15846. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b6af2c9703f203a2794be03d443af2e3-Paper.pdf.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In NAACL, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. ArXiv, abs/1803.05457, 2018.

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. arXiv preprint arXiv:2110.14168, 2021.

Michael Han Daniel Han and Unsloth team. Unsloth, 2023. URL https://github.com/unslothai/unsloth.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. Advances in neural information processing systems, 36:10088–10115, 2023.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 4133–4145, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.252. URL https://aclanthology.org/2023.emnlp-main.252/.

Elizabeth Donoway, Hailey Joren, Arushi Somani, Henry Sleight, Julian Michael, Michael R DeWeese, John Schulman, Ethan Perez, Fabien Roger, and Jan Leike. Quantifying elicitation of latent capabilities in language models. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In International Conference on Learning Representations, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In International conference on machine learning, pp. 10323–10337. PMLR, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024a. URL https://zenodo.org/records/12608602.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. In International Conference on Machine Learning, pp. 14884–14901. PMLR, 2024b.

Xuyang Ge, Fukang Zhu, Wentao Shu, Junxuan Wang, Zhengfu He, and Xipeng Qiu. Automatically identifying local and global circuits with linear computation graphs, 2024. URL https://arxiv.org/abs/2405.13868.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. Nature, 645(8081):633–638, 2025.

Demi Guo, Alexander Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), 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. 4884–4896, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.378. URL https://aclanthology.org/2021.acl-long.378/.

Wes Gurnee, Emmanuel Ameisen, Isaac Kauvar, Julius Tarng, Adam Pearce, Chris Olah, and Joshua Batson. When models manipulate manifolds: The geometry of a counting task. Transformer Circuits Thread, 2025. URL https://transformer-circuits.pub/2025/linebreaks/index.html.

Tal Haklay, Hadas Orgad, David Bau, Aaron Mueller, and Yonatan Belinkov. Position-aware automatic circuit discovery, 2025. URL https://arxiv.org/abs/2502.04577.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In Forty-first International Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=uubBZKM99Y.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In IEEE international conference on neural networks, pp. 293–299. IEEE, 1993.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. arXiv preprint arXiv:2402.12354, 2024.

Haoyu He, Jianfei Cai, Jing Zhang, Dacheng Tao, and Bohan Zhuang. Sensitivity-aware visual parameter-efficient fine-tuning. In Proceedings of the IEEE/CVF international conference on computer vision, pp. 11825–11835, 2023.

Shwai He, Liang Ding, Daize Dong, Jeremy Zhang, and Dacheng Tao. SparseAdapter: An easy approach for improving the parameter-efficiency of adapters. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), Findings of the Association for Computational Linguistics: EMNLP 2022, pp. 2184–2190, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. 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. Proceedings of the International Conference on Learning Representations (ICLR), 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. NeurIPS, 2021b.

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

Uijeong Jang, Jason D Lee, and Ernest K Ryu. Lora training in the ntk regime has no spurious local minima. In International Conference on Machine Learning, pp. 21306–21328. PMLR, 2024.

Maxwell Jia. Aime problem set 2024, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Songtao Wang, deqing wang, and Fuzhen Zhuang. MoRA: High-rank updating for parameter-efficient fine-tuning, 2024. URL https://openreview.net/forum?id=SxOrhLuuVz.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. arXiv preprint arXiv:2312.03732, 2023.

Dmitrii Kharlapenko, Stepan Shabalin, Fazl Barez, Arthur Conmy, and Neel Nanda. Scaling sparse feature circuit finding for in-context learning, 2025. URL https://arxiv.org/abs/2504.13756.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Soroush Abbasi Koohpayegani, Navaneet K L, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. NOLA: Compressing loRA using linear combination of random basis. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=TjfXcDgvzk.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. Vera: Vector-based random matrix adaptation. In The Twelfth International Conference on Learning Representations, 2024.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing llm behaviour to components, 2024. URL https://arxiv.org/abs/2403.00745.

Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. https://www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2025-01-22.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. Advances in neural information processing systems, 2, 1989.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: single-shot network pruning based on connection sensitivity. CoRR, abs/1810.02340, 2018. URL http://arxiv.org/abs/1810.02340.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In International Conference on Learning Representations, 2018. URL https://openreview.net/forum?id=ryup8-WCW.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. Advances in Neural Information Processing Systems, 36, 2024a.

Yang Li, Shaobo Han, and Shihao Ji. Vb-lora: Extreme parameter efficient fine-tuning with vector banks. In The 38th Conference on Neural Information Processing Systems (NeurIPS), 2024b.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: a simple, effective, and efficient reinforcement learning method for aligning large language models. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org, 2024c.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229.

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In Conference on Language Modeling (COLM), 2025.

Pan Lu et al. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In ICLR, 2024.

Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. Sql-r1: Training natural language to sql reasoning model by reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), Advances in Neural Information Processing Systems, volume 38, Main Conference, pp. 174505–174537. Curran Associates, Inc., 2025. doi: 10.52202/085713-5805. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/ff00aae6ce2b5e519ca4cf769f0c4a47-Paper-Conference.pdf.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. Advances in Neural Information Processing Systems, 35:17703–17716, 2022.

math ai. Aime problem set 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Mathematical Association of America. Amc 2023: American mathematics competitions 2023 dataset. https://huggingface.co/datasets/math-ai/amc23, 2023. Accessed: 2024-05-05.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. Advances in Neural Information Processing Systems, 37: 121038–121072, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. Advances in neural information processing systems, 35:17359–17372, 2022.

John X Morris, Niloofar Mireshghallah, Mark Ibrahim, and Saeed Mahloujifar. Learning to reason in 13 parameters. arXiv preprint arXiv:2602.04118, 2026.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. arXiv preprint arXiv:2301.05217, 2023.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. arXiv preprint arXiv:2403.17919, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In First Conference on Language Modeling, 2024. URL https://openreview.net/forum?id=Ti67584b98.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3052–3064, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.168. URL https://aclanthology.org/2024.acl-long.168/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. arXiv preprint arXiv:1907.10641, 2019.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. Advances in neural information processing systems, 33:20378–20389, 2020.

Michael Santacroce, Yadong Lu, Han Yu, Yuanzhi Li, and Yelong Shen. Efficient rlhf: Reducing the memory usage of ppo. arXiv preprint arXiv:2309.00754, 2023.

John Schulman and Thinking Machines Lab. Lora without regret. Thinking Machines Lab: Connectionism, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=ozX92bu8VA.

Wenhao Shi, Zhiqiang Hu, Yi Bin, Junhua Liu, Yang Yang, See-Kiong Ng, Lidong Bing, and Roy Ka-Wei Lee. Math-llava: Bootstrapping mathematical reasoning for multimodal large language models, 2024.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. CoRR, abs/2006.05467, 2020. URL https://arxiv.org/abs/2006.05467.

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

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwen2.5-vl, January 2025. URL https://qwenlm.github.io/blog/qwen2.5-vl/.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In Andreas Vlachos and Isabelle Augenstein (eds.), Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pp. 3274–3287, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.239. URL https://aclanthology.org/2023.eacl-main.239/.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Chaoqi Wang, Guodong Zhang, and Roger Baker Grosse. Picking winning tickets before training by preserving gradient flow. CoRR, abs/2002.07376, 2020. URL https://arxiv.org/abs/2002.07376.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. arXiv preprint arXiv:2211.00593, 2022.

Changyi Xiao, Mengdi Zhang, and Yixin Cao. BNPO: Beta normalization policy optimization, 2026. URL https://openreview.net/forum?id=eXigrPOI2G.

Xinyu Yang, Jixuan Leng, Geyang Guo, Jiawei Zhao, Ryumei Nakada, Linjun Zhang, Huaxiu Yao, and Beidi Chen. S2ft: Efficient, scalable and generalizable llm fine-tuning by structured sparsity. Advances in Neural Information Processing Systems, 37:59912–59947, 2024a.

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 The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024b. URL https://openreview.net/forum?id=Gi00NVru6n.

Qingyu Yin, Yulun Wu, Zhennan Shen, Sunbowen Lee, Zhilin Wang, Yanshu Li, Chak Tou Leong, Jiale Kang, and Jinjin Gu. Evaluating parameter efficient methods for RLVR. In Forty-third International Conference on Machine Learning, 2026. URL https://openreview.net/forum?id=76CIL1O0bz.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2026. URL https://openreview.net/forum?id=2a36EMSSTp.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 3911–3921, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. In The Twelfth International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=likXVjmh3E.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. LoRA-FA: Memory-efficient low-rank adaptation for large language models fine-tuning, 2024a. URL https://openreview.net/forum?id=RbKThNNFxr.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. LoRAPrune: Structured pruning meets low-rank parameter-efficient fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Findings of the Association for

Computational Linguistics: ACL 2024, pp. 3013–3026, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.178. URL https://aclanthology.org/2024.findings-acl.178/.

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, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Zicheng Zhang, Haoran Li, Yifeng Zhang, Guoqiang Gong, Jiaxing Wang, Junxing Hu, pengzhang liu, and Qixia Jiang. The primacy of magnitude in low-rank adaptation. In The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2026. URL https://openreview.net/forum?id=s4LnWgjacg.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In Forty-first International Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=hYHsrKDiX7.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. Lora-drop: Efficient lora parameter pruning based on output evaluation. In Proceedings of the 31st International Conference on Computational Linguistics, pp. 5530–5543, 2025.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. In Forty-first International Conference on Machine Learning, 2024. URL https://openreview.net/forum?id=txRZBD8tBV.

Kejian Zhu, Zhuoran Jin, Shangqing Tu, Hongbang Yuan, Yushi Bai, Kang Liu, Juanzi Li, and Jun Zhao. Sft conflicts, rl coexists: A theoretical and empirical analysis of multi-task learning for llms, 2026. URL https://arxiv.org/abs/2608.03573.

APPENDICES

CONTENTS

Appendix A Extended Related Work 18
Appendix B Detailed Results 20
Appendix C Ablations 23
Appendix D Additional Circuit Analysis 28
Appendix E Hyperparameters 42
Appendix F Additional Experiments 46

A EXTENDED RELATED WORK

We provide a broader survey of LoRA variants and parameter-efficient fine-tuning methods, organized by the axis of improvement each addresses.

Initialization. Standard LoRA initializes A from $\mathcal{N}(0, \sigma^2)$ and $B = 0$. PiSSA (Meng et al., 2024) initializes both factors from the principal singular vectors of the pre-trained weight, training in the most important subspace. OLoRA (Adilova & Kushmuratov, 2026; Büyükakyüz, 2024) uses QR decomposition for orthogonal initialization. CorDA (Yang et al., 2024b) performs task-context-aware SVD, orienting the decomposition toward the target data distribution. LoRAM (Zhang et al., 2026) incorporates weight magnitude information into initialization. These methods improve where in weight space the adapter starts; our work addresses which elements of the adapter are active.

Training dynamics. LoRA+ (Hayou et al., 2024) sets different learning rates for A and B, correcting an inefficiency in vanilla LoRA. rsLoRA (Kalajdzievski, 2023) introduces a $1/\sqrt{r}$ scaling factor to stabilize training across ranks. DoRA (Liu et al., 2024) decomposes weights into magnitude and direction, applying LoRA only to the directional component. FLoRA (Hao et al., 2024) resamples the A projection matrix periodically, interpreting LoRA as gradient compression. These are orthogonal to our element-level selection and could in principle be combined with circuit-based sparsity.

Rank allocation. AdaLoRA (Zhang et al., 2023) parameterizes adapters in SVD form and prunes singular values based on importance, allocating higher rank to more critical layers. DyLoRA (Valipour et al., 2023) trains across a range of ranks simultaneously. SoRA (Ding et al., 2023) learns sparse gates over rank-1 components. MELoRA (Ren et al., 2024) uses mini-ensembles of very low-rank adapters. These methods allocate budget at the rank or layer level; our circuits allocate at the individual element level.

Extreme parameter efficiency. VeRA (Kopiczko et al., 2024) achieves $\sim$10$\times$ reduction over LoRA by sharing frozen random matrices. FourierFT (Gao et al., 2024b) uses spectral coefficients over a Fourier basis. LoRA-XS (Bałazy et al., 2024) trains only a tiny $r \times r$ middle matrix. NOLA (Koohpayegani et al., 2024) expresses A and B as weighted sums of frozen random basis matrices, training only the mixture coefficients. VB-LoRA (Li et al., 2024b) composes adapters from a shared vector bank. RandLoRA (Albert et al., 2025) learns element-wise scaling over frozen random factors. Morris et al. (2026) scale LoRA down to as few as 13 trainable parameters. BitFit (Ben Zaken et al., 2022) trains only bias terms ($<$0.1% of parameters). Our sparse circuits operate in the same ultra-low-parameter regime but with a standard LoRA parameterization and informed selection.

Adapter compression and pruning. SparseAdapter (He et al., 2022) prunes adapter weights after training, showing that adapters are highly compressible. LoRA-Drop (Zhou et al., 2025) removes entire LoRA layers based on output contribution. LoRA-Prune (Zhang et al., 2024b) jointly prunes the base model and adapter. Diff pruning (Guo et al., 2021) learns a sparse weight diff with L_0 regularization. S^{2}FT (Yang et al., 2024a) selects structured groups (attention heads and FFN channels) in the full weight matrices and permutes them into dense submatrices for efficient training; our work operates at individual elements within LoRA adapters rather than structured blocks in the full model. These methods determine sparsity during or after training, whereas our circuit discovery operates

before training begins, selecting elements at initialization from gradient statistics: the sparsity pattern is fixed throughout and requires no iterative pruning schedule.

Parameter selection beyond adapters. LISA (Pan et al., 2024) importance-samples which layers to update at each step. LASER (Sharma et al., 2024) shows that removing higher-order weight components at specific layers can improve performance. Movement pruning (Sanh et al., 2020) selects fine-tuning parameters based on gradient-weight products. SNIP, GRASP, and SynFlow (Lee et al., 2018; Wang et al., 2020; Tanaka et al., 2020) score parameters at initialization without training, addressing failure modes such as layer collapse. The lottery ticket hypothesis (Frankle & Carbin, 2019) and its extensions to pre-trained models (Chen et al., 2020) show that sparse trainable subnetworks exist within dense networks. Our element-level circuit discovery within LoRA adapters is a new instantiation of this broad family.

Memory-efficient training. QLoRA (Dettmers et al., 2023) combines 4-bit quantization with LoRA. GaLore (Zhao et al., 2024) projects gradients into low-rank subspaces, enabling full-parameter training with LoRA-level memory. MoRA (Jiang et al., 2024) achieves high-rank updates via a square matrix with non-parametric operators. These address orthogonal memory concerns and could be combined with circuit-based sparsity.

RL fine-tuning. GRPO (Shao et al., 2024) uses group-relative rewards to eliminate the critic. DeepSeek-R1 (Guo et al., 2025) applies GRPO at scale for reasoning. DPO (Rafailov et al., 2023) reformulates RLHF as a classification problem. ReMax (Li et al., 2024c) simplifies REINFORCE for LLM alignment. Efficient RLHF methods (Santacroce et al., 2023) use LoRA for both policy and value networks. Our work studies how parameter placement within LoRA interacts with the RL training signal.

Theoretical foundations. Aghajanyan et al. (2021) establish low intrinsic dimensionality of fine-tuning. Li et al. (2018) introduce intrinsic dimension measurement for loss landscapes. Jang et al. (2024) prove LoRA training has no spurious local minima in the NTK regime. Zeng & Lee (2024) analyze the expressive power of low-rank adaptation. Biderman et al. (2024) show LoRA learns less but forgets less than full fine-tuning. Fisher information for parameter importance dates to Optimal Brain Damage (LeCun et al., 1989) and has been applied to continual learning (Kirkpatrick et al., 2017) and model merging (Matena & Raffel, 2022).

Circuits and Mechanistic Interpretability. Parameter selection is tied methodologically to Mechanistic Interpretability, which aims to explain how models perform given tasks. Circuits, minimal subgraphs of a model's computation graph that execute a specific task (Haklay et al., 2025), are found using ablations or interventions in the model activations, often requiring exhaustive searches. These circuits generally correspond to a specific task or capability (Cammarata et al., 2021; Wang et al., 2022; Gurnee et al., 2025). In our work, we mean something different by "circuit". Instead of a subgraph of the total computation graph, we aim to find, at a given budget, a set of individual entries of the LoRA $\mathbf{B}$ matrices that contribute most to efficient learning. This set is not necessarily unique, minimal, or complete. Several methods find circuits automatically (Ge et al., 2024; Haklay et al., 2025; Kharlapenko et al., 2025), whereas earlier methods require manual exploration (Nanda et al., 2023; Kramár et al., 2024). Our work repurposes neuron attribution techniques to identify parameter subsets for more effective PEFT.

B Detailed Results

Contents

Instruction Tuning: Full Results . . . . . B.1, p. 20
OOD Generalization: Full Results . . . . . B.2, p. 20
Cross-Task Transfer – GSM8K-Trained Models.

B.1 Instruction Tuning: Full Results

We conduct instruction tuning on 7 well-known datasets: MMLU (Hendrycks et al., 2021a), PiQA (Bisk et al., 2020), TruthfulQA (Lin et al., 2022), HellaSwag (Zellers et al., 2019), Arc-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019) and Winogrande (Sakaguchi et al., 2019). Table 5 reports per-benchmark SFT accuracy on Alpaca (mean ± SE, 5 seeds). We see that the seeds are extremely stable across all configurations. In all cases the circuits edge out the random placement.

Table 5: **Per-benchmark SFT accuracy (%) on Alpaca** (5 seeds). Subscripts denote standard error.

Model	Method	ARC-C	BoolQ	HellaS	MMLU	PIQA	TQA	Wino	Avg
Q2.5-1.5B	$\hat{F}$	$46.4_{\pm 0.2}$	$73.2_{\pm 0.6}$	$67.9_{\pm 0.0}$	$59.6_{\pm 0.1}$	$75.9_{\pm 0.1}$	$47.9_{\pm 0.2}$	$63.6_{\pm 0.3}$	$62.1_{\pm 0.1}$
	$\hat{S}$	$46.7_{\pm 0.8}$	$73.3_{\pm 0.6}$	$67.9_{\pm 0.0}$	$59.0_{\pm 0.2}$	$76.1_{\pm 0.1}$	$47.2_{\pm 0.5}$	$63.4_{\pm 0.2}$	$61.9_{\pm 0.2}$
	Random	$45.6_{\pm 0.5}$	$72.7_{\pm 0.9}$	$67.8_{\pm 0.0}$	$59.1_{\pm 0.1}$	$76.0_{\pm 0.2}$	$47.0_{\pm 0.4}$	$63.4_{\pm 0.0}$	$61.6_{\pm 0.2}$
	Full LoRA	$46.6_{\pm 0.5}$	$75.3_{\pm 0.3}$	$68.1_{\pm 0.1}$	$59.9_{\pm 0.0}$	$76.0_{\pm 0.0}$	$47.7_{\pm 0.5}$	$63.0_{\pm 0.1}$	**62.4**$_{\pm 0.2}$
Q2.5-3B	$\hat{F}$	$48.8_{\pm 0.1}$	$76.7_{\pm 0.1}$	$73.8_{\pm 0.0}$	$65.0_{\pm 0.0}$	$78.9_{\pm 0.1}$	$49.7_{\pm 0.1}$	$69.1_{\pm 0.1}$	$66.0_{\pm 0.0}$
	$\hat{S}$	$48.2_{\pm 0.1}$	$78.7_{\pm 0.2}$	$74.0_{\pm 0.0}$	$65.0_{\pm 0.1}$	$78.9_{\pm 0.0}$	$49.9_{\pm 0.2}$	$68.9_{\pm 0.1}$	$66.2_{\pm 0.0}$
	Random	$47.7_{\pm 0.2}$	$77.5_{\pm 0.1}$	$73.7_{\pm 0.0}$	$65.0_{\pm 0.0}$	$78.8_{\pm 0.1}$	$49.7_{\pm 0.2}$	$68.4_{\pm 0.2}$	$65.8_{\pm 0.1}$
	Full LoRA	$48.7_{\pm 0.4}$	$79.4_{\pm 0.4}$	$74.0_{\pm 0.0}$	$65.0_{\pm 0.1}$	$78.8_{\pm 0.1}$	$51.2_{\pm 0.3}$	$69.3_{\pm 0.3}$	**66.7**$_{\pm 0.2}$
Q2.5-7B	$\hat{F}$	$52.8_{\pm 0.1}$	$84.7_{\pm 0.1}$	$79.1_{\pm 0.0}$	$71.8_{\pm 0.1}$	$80.5_{\pm 0.1}$	$57.5_{\pm 0.2}$	$74.0_{\pm 0.1}$	**71.5**$_{\pm 0.0}$
	$\hat{S}$	$52.5_{\pm 0.2}$	$85.2_{\pm 0.1}$	$79.1_{\pm 0.0}$	$71.9_{\pm 0.1}$	$79.9_{\pm 0.0}$	$57.5_{\pm 0.2}$	$73.4_{\pm 0.1}$	$71.4_{\pm 0.1}$
	Random	$51.8_{\pm 0.1}$	$85.1_{\pm 0.0}$	$78.9_{\pm 0.0}$	$71.8_{\pm 0.1}$	$80.0_{\pm 0.0}$	$57.1_{\pm 0.1}$	$73.2_{\pm 0.1}$	$71.1_{\pm 0.0}$
	Full LoRA	$52.8_{\pm 0.1}$	$85.3_{\pm 0.1}$	$79.1_{\pm 0.0}$	$71.7_{\pm 0.1}$	$80.4_{\pm 0.1}$	$58.3_{\pm 0.2}$	$72.5_{\pm 0.2}$	**71.5**$_{\pm 0.0}$
L3.2-1B	$\hat{F}$	$37.7_{\pm 0.3}$	$65.4_{\pm 0.3}$	$63.1_{\pm 0.1}$	$35.8_{\pm 0.4}$	$75.7_{\pm 0.2}$	$40.3_{\pm 0.3}$	$61.2_{\pm 0.3}$	$54.2_{\pm 0.1}$
	$\hat{S}$	$37.7_{\pm 0.2}$	$66.1_{\pm 0.3}$	$63.0_{\pm 0.2}$	$35.6_{\pm 0.2}$	$75.6_{\pm 0.2}$	$41.2_{\pm 0.2}$	$61.5_{\pm 0.1}$	$54.4_{\pm 0.1}$
	Random	$37.2_{\pm 0.2}$	$64.5_{\pm 0.2}$	$63.0_{\pm 0.0}$	$37.6_{\pm 0.1}$	$75.2_{\pm 0.2}$	$39.4_{\pm 0.1}$	$61.1_{\pm 0.2}$	$54.0_{\pm 0.1}$
	Full LoRA	$37.4_{\pm 0.3}$	$65.7_{\pm 0.5}$	$63.4_{\pm 0.1}$	$37.0_{\pm 0.1}$	$75.6_{\pm 0.2}$	$40.8_{\pm 0.2}$	$61.2_{\pm 0.1}$	**54.5**$_{\pm 0.1}$
L3.2-3B	$\hat{F}$	$46.0_{\pm 0.2}$	$77.7_{\pm 0.3}$	$73.0_{\pm 0.1}$	$54.6_{\pm 0.3}$	$78.2_{\pm 0.1}$	$45.7_{\pm 0.3}$	$70.6_{\pm 0.2}$	$63.7_{\pm 0.1}$
	$\hat{S}$	$46.6_{\pm 0.1}$	$77.0_{\pm 0.3}$	$73.2_{\pm 0.1}$	$54.5_{\pm 0.3}$	$78.3_{\pm 0.1}$	$46.3_{\pm 0.3}$	$70.8_{\pm 0.1}$	$63.8_{\pm 0.1}$
	Random	$46.1_{\pm 0.1}$	$75.5_{\pm 0.2}$	$72.6_{\pm 0.0}$	$54.9_{\pm 0.1}$	$78.3_{\pm 0.1}$	$42.5_{\pm 0.2}$	$70.1_{\pm 0.0}$	$62.9_{\pm 0.1}$
	Full LoRA	$46.4_{\pm 0.2}$	$77.8_{\pm 0.4}$	$73.1_{\pm 0.0}$	$54.6_{\pm 0.3}$	$77.9_{\pm 0.1}$	$46.2_{\pm 0.4}$	$70.9_{\pm 0.1}$	**63.9**$_{\pm 0.0}$
L3.1-8B	$\hat{F}$	$55.4_{\pm 0.1}$	$83.9_{\pm 0.1}$	$78.4_{\pm 0.0}$	$63.5_{\pm 0.0}$	$81.8_{\pm 0.1}$	$50.3_{\pm 0.2}$	$74.4_{\pm 0.1}$	$69.7_{\pm 0.0}$
	$\hat{S}$	$55.8_{\pm 0.2}$	$83.8_{\pm 0.1}$	$78.2_{\pm 0.0}$	$63.2_{\pm 0.1}$	$81.8_{\pm 0.1}$	$50.4_{\pm 0.3}$	$74.5_{\pm 0.1}$	$69.6_{\pm 0.0}$
	Random	$54.8_{\pm 0.1}$	$83.5_{\pm 0.1}$	$78.0_{\pm 0.0}$	$63.4_{\pm 0.1}$	$80.9_{\pm 0.1}$	$48.3_{\pm 0.1}$	$74.5_{\pm 0.1}$	$69.1_{\pm 0.0}$
	Full LoRA	$55.9_{\pm 0.5}$	$83.9_{\pm 0.1}$	$78.5_{\pm 0.1}$	$63.4_{\pm 0.2}$	$81.1_{\pm 0.2}$	$52.4_{\pm 0.7}$	$74.4_{\pm 0.2}$	**69.9**$_{\pm 0.1}$

ARC-C = ARC-Challenge (acc_norm), HellaS = HellaSwag (acc_norm), TQA = TruthfulQA MC2, Wino = WinoGrande, PIQA (acc_norm).
Q2.5 = Qwen2.5, L3.2 = Llama-3.2, L3.1 = Llama-3.1.

B.2 OOD Generalization: Full Results

All models trained with GRPO, evaluated OOD with n=64 generations per problem at temperature 0.7. Solid lines show pass@k; dashed lines show maj@k (AMC panels only). 7B MATH-trained models omit Random, which did not learn at this scale. We show results for AIME 2024 (Jia, 2024), AIME 2025 (math ai, 2025) and AMC 2023 (Mathematical Association of America, 2023).

At 3B, $\hat{F}$ leads on AMC at every k, and Full LoRA falls below the base model from pass@8 onward on AIME 2025, consistent with sparse masks providing implicit regularization. At 7B, circuits match Full LoRA across all three benchmarks, with $\hat{S}$ slightly ahead on AIME 2025.

B.2.1 Cross-Task Transfer

MATH→GSM8K transfer is robust at both scales. GSM8K→MATH is asymmetric: at 3B, Full LoRA catastrophically degrades while $\hat{S}$ is the only method to improve over base in both directions.

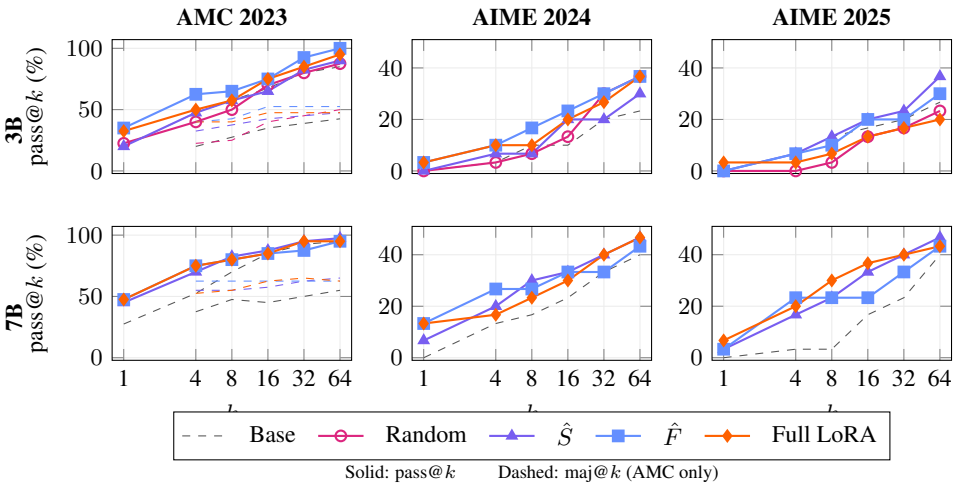

Figure 5: **MATH-trained pass@k and maj@k** (Qwen2.5-3B/7B). $\hat{F}$ at 50K reaches 100% pass@64 on AMC (3B) and 95% (7B). On AIME 2025, Full LoRA degrades below base at 3B while $\hat{S}$ reaches 36.7% pass@64; at 7B all methods improve over base. AIME maj@k is near-zero for all methods; AMC maj@64 peaks at 52.5% ($\hat{F}$, 3B) and 65% ($\hat{S}$, 7B).

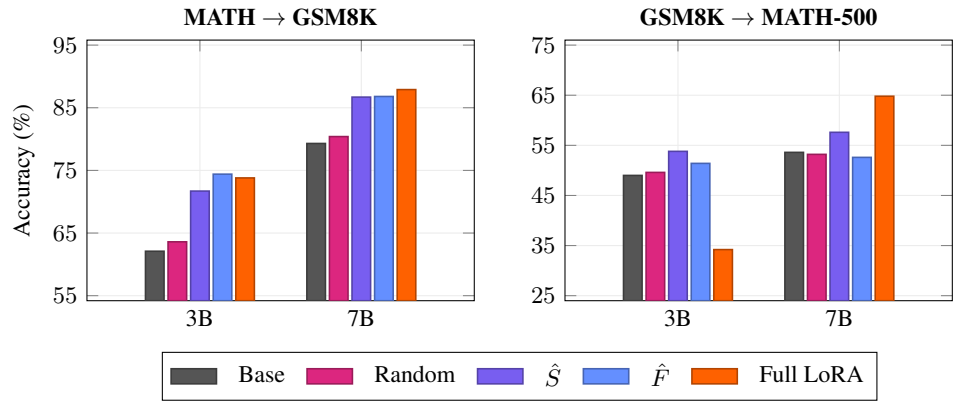

Figure 6: **Cross-task greedy transfer** (Qwen2.5-3B/7B). Gaps indicate methods not evaluated at that scale. MATH→GSM8K: circuits match Full LoRA to within 1.1pp. GSM8K→MATH-500: at 3B, Full LoRA degrades to 34.2% (−14.8pp below base) while $\hat{S}$ at 10K improves to 53.8%; at 7B, Full LoRA transfers well (64.8%, +11.2pp).

B.2.2 GSM8K-Trained Models

At 3B, Full LoRA degrades OOD, with AMC pass@1 dropping to 10% , while $\hat{S}$ at 10K preserves and improves OOD capability. At 7B, Full LoRA does not degrade and dominates on AIME, consistent with the cross-task pattern (Fig. 6): sparse placement helps most when the model is capacity-constrained or the training task is narrow.

Why do circuits generalize? Circuits discovered on different math datasets (GSM8K, MATH, NuminaMath) share 17–37% of their top-k elements on Qwen2.5-1.5B, hundreds of times the chance overlap (Table 15). This suggests circuit discovery identifies a core subnetwork the model consistently wants to update for a given task family, constraining updates to shared directions and providing natural regularization that full LoRA lacks.

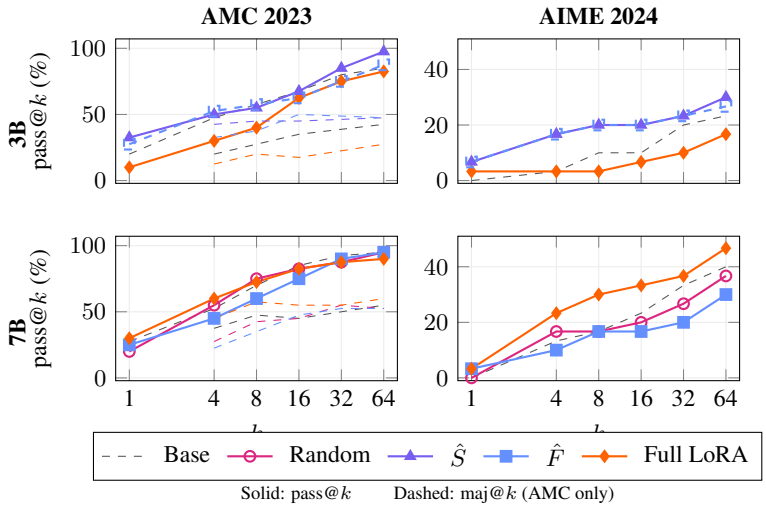

Figure 7: **GSM8K-trained pass@k and maj@k** (Qwen2.5-3B/7B). At 3B, Full LoRA degrades below base on AMC while $\hat{S}$ at 10K reaches 97.5% pass@64. At 7B, Full LoRA transfers well and dominates on AIME 2024.

C Ablations

Contents

Training A+B vs. B-Only . . . . . C.1, p. 23
Monte Carlo Convergence . . . . . C.2, p. 23
Sensitivity to A Initialization . . . . . C.3, p. 23
Circuit Discovery Cost . . . . . C.4, p. 24
Stability Across Seeds . . . . . C.5, p. 25
Synthetic Experiment Details . . . . . C.6, p. 25
Budget Selection . . . . . C.7, p. 25
Random Placement under GRPO . . . . . C.8, p. 26

C.1 Training A+B vs. B-Only

Our method freezes $\mathbf{A}$ and selects sparse elements of $\mathbf{B}$ to train. A natural question is whether jointly training sparse elements of both $\mathbf{A}$ and $\mathbf{B}$ at the same total budget would perform better. We compare the two strategies on Qwen3-0.6B across budgets from 1K to 500K parameters, using both perplexity evaluation (5 seeds) and GSM8K downstream accuracy as well as Qwen2.5-1.5B on instruction tuning.

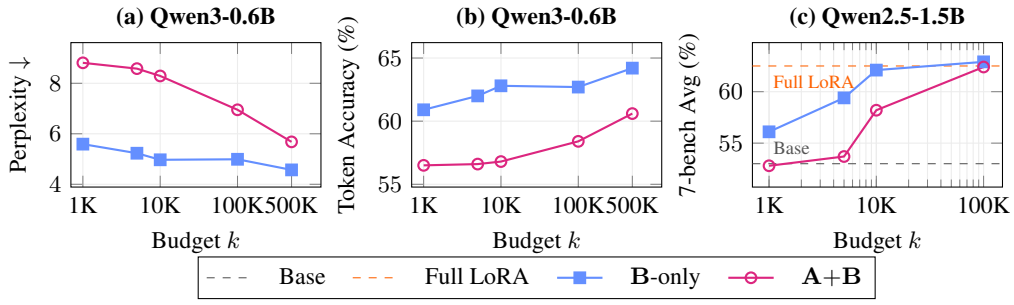

Figure 8: **B-only vs. A+B scaling curves** at matched parameter budget. (a, b) Qwen3-0.6B perplexity and token accuracy; (c) Qwen2.5-1.5B 7-benchmark SFT average. B-only (blue) dominates A+B (magenta) at every budget, with the gap largest at small k. Dashed lines in (c) show base model and full LoRA references.

B-only dominates A+B at every budget. At k=1K the gap is 3.2 points in perplexity; even at k=500K it remains substantial which manifests in the downstream accuracy on GSM8K.

This validates our design choice of fixing $\mathbf{A}$ and selecting within $\mathbf{B}$. Our finding is consistent with LoRA-FA (Zhang et al., 2024a), which showed that freezing $\mathbf{A}$ after initialization preserves or improves performance in the full-rank setting.

C.2 Monte Carlo Convergence

Table 6 reports top-10K element overlap with the N=100 reference circuit across three Qwen2.5 model sizes on GSM8K. Random 10K-element subsets overlap by $< 0.1\%$ in expectation. Convergence is consistent across models and improves with scale. We observe the same pattern on MATH (Qwen2.5-1.5B).

C.3 Sensitivity to A Initialization

We test whether the discovered circuit depends on the specific random initialization of $\mathbf{A}$. Starting from the same base model, we perturb $\mathbf{A}' = \mathbf{A} + \varepsilon\|\mathbf{A}\|_F \cdot \mathbf{\Delta}/\|\mathbf{\Delta}\|_F$ (where $\mathbf{\Delta} \sim \mathcal{N}(0, \mathbf{I})$) and re-run circuit discovery. Table 7 reports top-10K overlap with the unperturbed circuit.

Table 6: **Monte Carlo convergence.** Top-10K overlap (%) with the N=100 circuit (GSM8K).

	Qwen2.5-1.5B		Qwen2.5-3B		Qwen2.5-7B	
N	$\hat{S}$	$\hat{F}$	$\hat{S}$	$\hat{F}$	$\hat{S}$	$\hat{F}$
10	94.8	94.8	92.3	90.4	96.8	95.9
25	96.6	96.4	95.4	94.1	98.4	98.0
50	98.2	97.9	97.6	97.0	99.1	98.8
100	100.0	100.0	100.0	100.0	100.0	100.0

Table 7: **Sensitivity to A perturbation.** Top-10K overlap (%) with the unperturbed (ε=0) circuit across three model sizes (GSM8K, N=50 batches).

ε	1.5B	3B	7B
0.01	98.9	99.3	99.4
0.05	96.1	96.7	96.8
0.1	92.4	93.5	93.7
0.25	82.3	84.0	84.0
0.5	63.3	70.4	68.2
1.0	42.3	53.8	49.0
2.0	27.3	42.0	42.3

At small perturbations ($\varepsilon \leq 0.1$), over 92% of the top-10K elements are preserved across all model sizes. This confirms that our circuit reflects the base model's structure rather than the random $\mathbf{A}$ initialization. Degradation at larger ε is expected: $\mathbf{A}$ scales the gradient signal reaching $\mathbf{B}$, so large changes to $\mathbf{A}$ change which $\mathbf{B}$ elements receive the strongest gradients.

C.4 Circuit Discovery Cost

We benchmark the cost of circuit discovery (N=50 forward-backward passes, batch size 1, max length 1024, bfloat16) across Qwen2.5 model sizes on a single NVIDIA RTX PRO 6000 GPU (96 GB). No optimizer state is required. Only the accumulated gradient tensor (float32) is stored on CPU.

Table 8: **Circuit discovery cost** (N=50, single GPU). Discovery time excludes model loading. Peak VRAM includes model weights, LoRA adapters, activations, and gradients. Grad tensor is CPU memory for accumulated scores. The last column is discovery FLOPs as a percentage of a full SFT / GRPO run.

Model	Params	LoRA-B	Time (s)	Peak VRAM	Grad Tensor	% SFT / GRPO
Qwen2.5-0.5B	494M	9.7M	4.9	2.6 GB	39 MB	0.49 / 0.18
Qwen2.5-1.5B	1.5B	20.6M	5.5	5.5 GB	83 MB	0.49 / 0.18
Qwen2.5-3B	3.1B	33.0M	7.3	9.7 GB	132 MB	0.49 / 0.18
Qwen2.5-7B	7.6B	44.5M	8.5	19.6 GB	178 MB	0.49 / 0.18

Discovery time scales sub-linearly with model size: 7B takes only 1.7× longer than 0.5B (8.5s vs. 4.9s), since kernel launch overhead dominates at smaller scales.

Circuit discovery is amortizable: once computed, the same circuit can be reused across training runs with different hyperparameters, learning rates, or training durations.

Practical implications. (1) Circuit discovery is amortizable: once computed, the same circuit can be reused across training runs with different hyperparameters, learning rates, or training durations. (2) No optimizer state is needed, so peak VRAM is comparable to inference with a backward pass. (3) The accumulated gradient tensor is small (39–178 MB on CPU), making discovery feasible even on memory-constrained hardware where training would require offloading or sharding.

C.5 Stability Across Seeds

Circuit placement is stable across $\mathbf{A}$ initializations: layer concentration and module preferences are consistent within each architecture (Table 9).

Table 9: **Circuit placement stability across A-seeds** (top 10K, Alpaca). Mean ± std over 4 random $\mathbf{A}$ seeds and both scoring functions (8 runs). Layer and module preferences are consistent regardless of the random projection.

	Layer concentration		Top modules (% of circuit)		
Model	**Early (0–4)**	**std**	**1st**	**2nd**	**3rd**
Qwen2.5-3B	71.7%	±3.3	V (35.3)	Gate (23.5)	Up (16.1)
Qwen2.5-7B	98.9%	±0.2	V (53.9)	O (29.5)	Gate (8.3)
Llama-3.2-3B	97.4%	±0.6	Down (91.5)	V (6.6)	Up (0.9)
Llama-3.1-8B	96.8%	±1.8	V (51.3)	Down (46.7)	Up (1.3)

C.6 Synthetic Experiment Details

We use a 2-layer MLP with ReLU: $\mathbf{y} = \mathbf{W}_2\,\mathrm{ReLU}(\mathbf{W}_1\mathbf{x})$, where $\mathbf{W}_1 \in \mathbb{R}^{64\times128}$ and $\mathbf{W}_2 \in \mathbb{R}^{32\times64}$. We adapt $\mathbf{W}_1$ with a rank-16 LoRA: $\mathbf{W}_1^{\text{eff}} = \mathbf{W}_1 + \mathbf{BA}$, where $\mathbf{A} \in \mathbb{R}^{16\times128}$ is frozen (Kaiming initialization) and $\mathbf{B} \in \mathbb{R}^{64\times16}$ is initialized to zero. The total adapter has $|\mathbf{B}| = 1{,}024$ parameters. A mask $\mathbf{m} \in \{0,1\}^{64\times16}$ selects k elements for training. Full LoRA trains both $\mathbf{A}$ and $\mathbf{B}$ jointly.

The two panels use different targets to model the SFT/RL gradient structure. **SFT-like:** the target network differs from the base by a rank-2 additive residual on $\mathbf{W}_1$. Since the residual is dense and full-rank in the hidden dimension, gradients on $\mathbf{B}$ are distributed across all entries, so any parameter subset captures useful signal. **RL-like:** the target is a near-sparse $\mathbf{B}^*$ where only 5% of entries are large (0.3 std) and the rest are near-zero (0.01 std). Gradients concentrate on the few active entries, modeling the concentrated gradient structure we observe under GRPO (§3.3).

We run 100 forward-backward passes at initialization (batch size 128) and select the top-k elements of $\mathbf{B}$ by $|\nabla_{\mathbf{B}}\mathcal{L}|$, the same procedure as for LLMs (§3.2).

For SFT-like target, we use Adam, batch size 128, learning rate 5×10^{-2}, 5,000 steps, clean MSE gradients. For RL-like target, we use SGD with additive Gaussian noise ($\sigma = 0.005$) and gradient clipping (norm ≤ 1.0), batch size 4, learning rate 3×10^{-3}, 15,000 steps. The noise and small batch model the high-variance gradients characteristic of GRPO, where the reward signal is binary and most generations produce near-zero advantage.

We report MSE on a held-out test set (4,096 examples), normalized by the baseline MSE (no adaptation, $\mathbf{B} = 0$). All results are averaged over 10 seeds.

C.7 Budget Selection

Fig. 9 shows how performance scales with the parameter budget k under SFT for two model sizes.

VLM SFT budget sensitivity. At k=5K, $\hat{S}$ matches full LoRA while $\hat{F}$ and random fall short. The circuit methods are stable across the 5K–10K budget range; we report the 10K budget in the main paper.

GRPO budget scaling. Table 11 reports circuit accuracy under GRPO across budgets from 1K to 10K on both GSM8K and MATH-500 for Qwen2.5-1.5B. Unlike SFT, where random placement works at all budgets (Fig. 9), under GRPO random fails uniformly while circuit methods scale smoothly with budget.

On GSM8K, $\hat{F}$ outperforms $\hat{S}$ at 1K but the gap closes at 5K and reverses at 10K. On MATH-500, $\hat{S}$ leads at every budget. The scoring function ranking is task-dependent at extreme sparsity, consistent with the hypothesis that $\hat{S}$ (directional consistency) is more robust on harder tasks with more diffuse

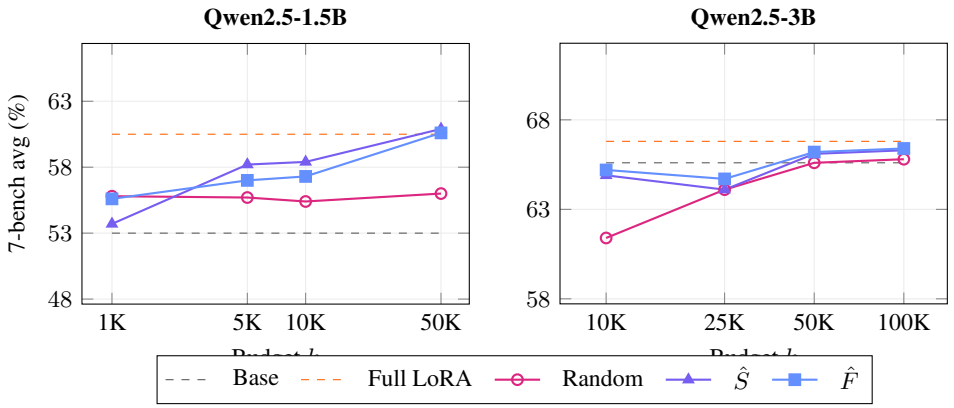

Figure 9: **Budget sweep under Alpaca SFT** (7-benchmark avg). Dashed lines mark base and full-LoRA references. On 1.5B, random placement is flat across budgets (55–56%) due to persistent MMLU collapse, while both circuits scale steadily and match full LoRA by k=50K. On 3B, circuits lead random by 3–4pp at k=10K; all methods converge toward full LoRA by k=100K.

Table 10: **VLM SFT on MathVista** (Qwen2.5-VL-7B). Circuit methods ($\hat{S}$, $\hat{F}$) across parameter budgets. Reference: Base = 62.5, Full LoRA = 66.9.

k	$\hat{S}$	$\hat{F}$	**Random**
5K	**66.9**	66.3	66.2
10K	**67.8**	66.8	66.4

gradient signal. Random placement fails uniformly at all budgets on both tasks, confirming that the collapse is structural.

We also show the 1K budget tier on MATH-500 for Qwen2.5-3B. $\hat{F}$ at just 1,000 trainable parameters matches full LoRA, the most extreme sparsity result in this paper, and shows that circuit discovery scales across model sizes.

SFT budget scaling. Table 12 scores over different rank and budget values for models trained on Bespoke-Stratos and tested on GPQADiamond. The selected circuits improve over the base model in nearly every configuration, and both gain from the larger budget at r=32.

C.8 Random Placement under GRPO

Tables 3 and 11 show that random placement fails under GRPO. Since sparse runs use a larger α than full LoRA, a natural question is whether random placement would succeed at a different budget or a different α. It does not.

First, random placement never leaves base level: across a $64\times$ range of α, and even when the budget is varied from 1K to 10K. Raising α by $128\times$ does not compensate for random placement, so the failure is structural rather than a scaling artifact. The same holds at 3B scale as well.

Second, the circuits do need the larger α. At $\alpha = 32$ they too sit near base level, and accuracy climbs monotonically until it saturates around $\alpha = 32r = 1024$. The scaling is therefore necessary for sparse training but not sufficient: it restores the update size that sparsity removes (§E.2), and applying the identical scaling to random placement changes nothing. What separates the two is which entries are trained.

Table 11: **GRPO budget scaling across models and tasks.** Circuit methods improve consistently with budget, while random placement fails. At extreme sparsity (k=1K), $\hat{F}$ can match full LoRA on larger models.

Model	Task	k	$\hat{S}$	$\hat{F}$	Random
Qwen2.5-1.5B	GSM8K	1K	43.2	**47.5**	9.5
		5K	62.2	**62.8**	9.8
	MATH-500	1K	**32.8**	29.2	18.0
		5K	**37.0**	35.0	18.4
Qwen2.5-3B	MATH-500	1K	58.4	**62.4**	51.2
Full LoRA (1.5B)			GSM8K: 63.7		MATH: 43.5
Full LoRA (3B)			MATH: 63.5		

Table 12: **Budget for SFT on Bespoke-Stratos (Reasoning); Accuracy on GPQADiamond.** Subscripts give the share of full LoRA's gain over the base model that the method recovers. r32 and r128 refers to LoRA rank 32 and 128.

Model	Base	Random	$\hat{S}$	$\hat{F}$	LoRA	k
Qwen2.5-7B-it r32	31.6	$37.4_{116\%}$	$35.6_{80\%}$	$33.3_{34\%}$	36.6	5K
Qwen2.5-7B-it r32	31.6	$37.4_{116\%}$	$\mathbf{38.9}_{146\%}$	$\mathbf{38.9}_{146\%}$	36.6	50K
Qwen2.5-7B-it r128	31.6	$41.1_{50\%}$	$43.3_{61\%}$	$31.1_{-3\%}$	**50.7**	5K
Qwen2.5-7B-it r128	31.6	$36.5_{26\%}$	$40.6_{47\%}$	$35.4_{20\%}$	50.7	50K

Table 13: **Random placement fails under GRPO at every** α (Qwen2.5-1.5B, GSM8K, fixed k=10K budget; base model 9.5%). Holding the budget fixed isolates scaling from placement: circuit accuracy rises with α and saturates near $\alpha = 32r$, while random placement stays at base level throughout.

	LoRA α						
Placement	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**
$\hat{S}$	10.2	23.7	37.4	54.5	60.8	64.2	64.7
$\hat{F}$	11.6	25.2	40.8	57.3	62.1	65.9	65.3
Random	9.4	9.1	9.7	9.9	9.4	9.5	9.7

D Additional Circuit Analysis

Contents

Circuit Anatomy: Full Results . . . . . D.1, p. 28
Per-Layer Score Concentration . . . . . D.2, p. 29
Cross-Dataset Circuit Overlap . . . . . D.3, p. 29
Scoring Functions: Formal Details . . . . . D.4, p. 30
Cumulative Update Alignment . . . . . D.5, p. 31
Gradient Structure: Full Results . . . . . D.6, p. 32
Sensitivity to the Scoring Objective . . . . . D.7, p. 32
Knockout Sweep Validation . . . . . D.8, p. 34
Module-Type Dissociation: Readers and Writers . . . . . D.9, p. 35
Gradient Sign Consistency – SVD Alignment.
Cross-Architecture Validation: Llama-3.2-3B . . . . . D.10, p. 36
Weight Norm and Behavior Change Dynamics . . . . . D.11, p. 38

D.1 Circuit Anatomy: Full Results

Circuits for Qwen and Llama models. Table 4 shows the placement density for the $\hat{S}$ circuit for the Qwen model. Here we provide the placements for the other circuits for other model families.

Table 14 shows the concentration of $\hat{F}$ for the Qwen models. The results mimic those of the $\hat{S}$ placements (Table 4). $\hat{F}$ circuits show the same early-layer concentration as $\hat{S}$, except on 3B MATH, where the budget spreads across layers 0–23. Fig. 10 shows the placement of the circuits for Llama-3.2-3B-Instruct on GSM8K. The circuit is more distributed partly due to the larger budget, but like the Qwen models still favors the V/O modules.

Table 14: $\hat{F}$**-circuit composition** (top 10K entries, % of circuit). "Param" shows each module type's share of all B parameters. V and O take most of the circuit at 7B, where O in particular concentrates; at 3B the budget spreads across V, Gate and Up. Gate and Up are consistently underrepresented relative to their parameter share.

		% of circuit (top 10K)			
Module	**Param %**	**3B MATH**	**3B GSM8K**	**7B MATH**	**7B GSM8K**
Q	7.1	3.6	0.4	0.9	0.2
K	0.9	3.1	6.4	3.5	0.9
V	0.9	17.6	37.1	34.9	30.6
O	7.1	8.9	1.3	40.4	54.0
Gate	38.4	30.3	30.9	10.7	5.4
Up	38.4	23.5	16.4	9.3	8.9
Down	7.1	12.9	7.5	0.3	0.1

Vision-Language Models. We see a similar preference for V and O modules. Most of the budget is now allocated to the vision encoder, concentrated in the earlier layers. Random placement is heavily allocated towards the language model simply because it has more parameters (Fig. 11).

The VLM result demonstrates that circuit discovery can identify which component of a multi-modal architecture is the training bottleneck. For a visual math task, the language model already has strong mathematical reasoning; the gradient signal concentrates in the vision encoder, which must learn to extract task-relevant information from images. A practitioner choosing modules to fine-tune would need domain expertise to reach this conclusion; the circuit scores reveal it automatically.

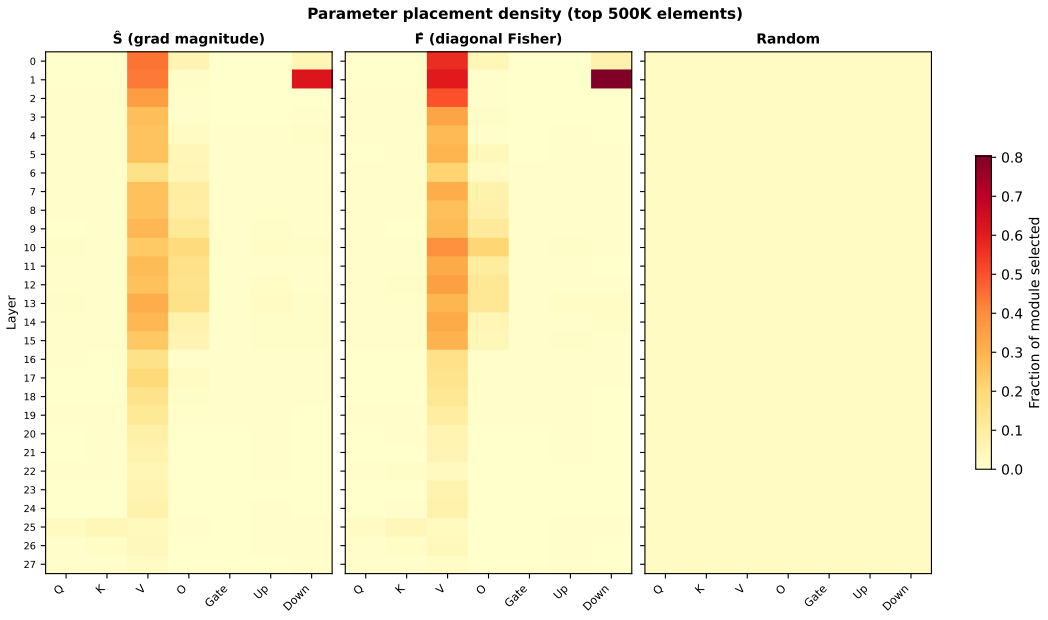

Figure 10: **Llama-3.2-3B-Instruct, GSM8K** (top 500K). At the larger budget (2% of $\mathbf{B}$), the circuit is more distributed but still favors V/O in the first half of layers. The instruct model shows similar module preferences to base Qwen models.

D.2 Per-Layer Score Concentration

The heatmaps in §D.1 show which modules receive budget; Fig. 12 shows how concentrated that budget is across depth. Panel (a) confirms that even within a single layer, gradient energy is far from uniformly distributed. The top 0.1% of elements capture 40–80$\times$ more energy than a random subset of the same size. Panel (b) shows that global top-k selection allocates budget non-uniformly across layers: early layers (depth < 0.2) receive a disproportionate share. Both patterns are consistent across 3B and 7B, and both scoring functions ($\hat{S}$, $\hat{F}$) produce similar layer profiles.

D.3 Cross-Dataset Circuit Overlap

We measure top-k element overlap between $\hat{S}$ circuits discovered on three math datasets (GSM8K, MATH, NuminaMath) using Qwen2.5-1.5B. Two random k-element subsets of the 20.6M entries of $\mathbf{B}$ overlap by $k/20.6$M in expectation. Table 15 shows that circuits share 17–37% of elements across datasets, hundreds of times above chance, indicating that circuit discovery identifies a stable task-general subnetwork rather than dataset-specific artifacts.

Table 15: **Cross-dataset circuit overlap** (Qwen2.5-1.5B, $\hat{S}$). Fraction of top-k elements shared between circuits discovered on different math datasets. Chance overlap is $k/20.6$M.

k	GSM8K–MATH	GSM8K–Numina	MATH–Numina	Chance
1K	21.1%	22.1%	16.7%	0.005%
5K	29.7%	29.1%	28.2%	0.024%
10K	29.6%	29.8%	29.6%	0.049%
50K	29.5%	30.9%	32.1%	0.24%
500K	35.6%	35.8%	37.3%	2.4%

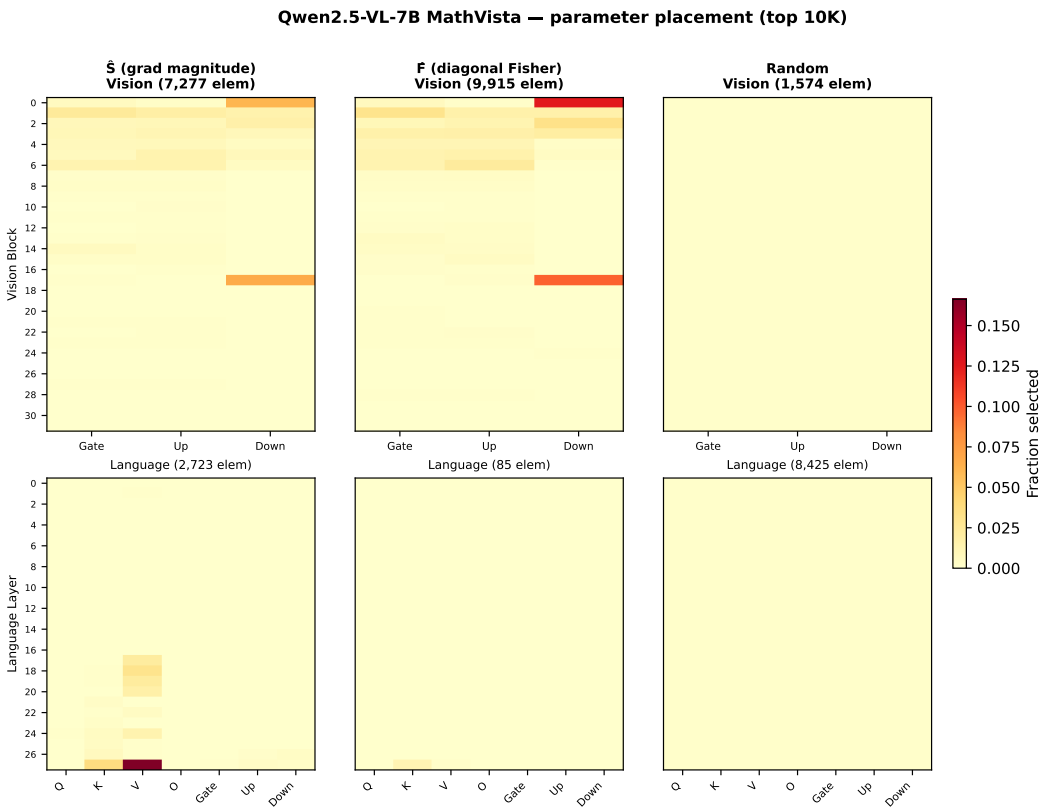

Figure 11: **Qwen2.5-VL-7B, MathVista** (top 10K). Circuit discovery automatically routes budget to the vision encoder: $\hat{S}$ allocates 73% to vision (vs. 16% for random), and $\hat{F}$ allocates 99%. Within vision, both methods concentrate on early blocks (0–6) and block 16. The few language elements selected by $\hat{S}$ go to V/O in late layers (16–19, 26), the same module preference as LLM-only circuits. Random placement inverts this, allocating 84% to language simply because it has more parameters.

D.4 Scoring Functions: Formal Details

We provide formal definitions and convergence properties of the two circuit scoring functions introduced in §3.2.

Definition 1. The **Fisher score** for element (i, j) of $\mathbf{B}$ is:

$$\hat{F}_{ij} = \frac{1}{N} \sum_{n=1}^{N} g_{n,ij}^2.$$

As a sample mean of i.i.d. non-negative random variables, $\hat{F}_{ij} \to F_{ij} = \mathbb{E}[g_{ij}^2]$ almost surely by the strong law of large numbers, with $\mathrm{Var}(\hat{F}_{ij}) = O(1/N)$ by Chebyshev's inequality.

Definition 2. The **gradient magnitude score** for element (i, j) of $\mathbf{B}$ is:

$$\hat{S}_{ij} = \left| \frac{1}{N} \sum_{n=1}^{N} g_{n,ij} \right|.$$

As a sample mean passed through the absolute value, $\hat{S}_{ij} \to |\mu_{ij}|$ where $\mu_{ij} = \mathbb{E}[g_{ij}]$, with convergence at rate $O(1/\sqrt{N})$ by the CLT.

Relationship. The two scores are connected by the bias-variance decomposition:

$$\mathbb{E}[g_{ij}^2] = \mu_{ij}^2 + \sigma_{ij}^2.$$

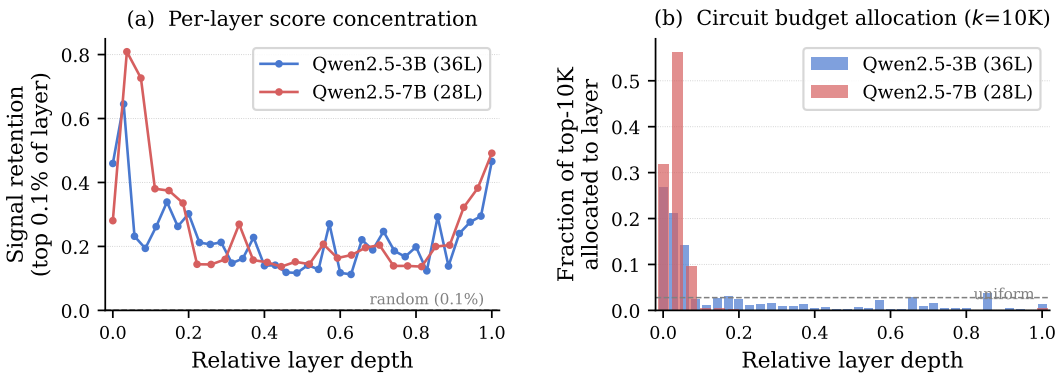

Figure 12: **Per-layer circuit analysis** for Qwen2.5-3B (36L) and Qwen2.5-7B (28L) on GSM8K with budget k=10K. **(a)** Signal retention: fraction of each layer's gradient energy captured by the top 0.1% of elements. Both models show strong concentration in early layers ($\sim$40$-$80$\times$ above the random baseline), confirming that circuit scores are far from uniform. **(b)** Circuit budget allocation: fraction of the global top-k budget assigned to each layer. The budget clusters heavily in the first $\sim$20% of layers, with a secondary peak near the output.

Fisher equals the squared mean gradient plus the gradient variance. An element can have high Fisher but low $\hat{S}$ if its gradients are large but oscillate in sign; high $\hat{S}$ requires directional consistency across examples.

Why small N suffices. For ranking stability, what matters is separation between top-ranked and median scores. Empirically, the score distribution is heavy-tailed: top elements have scores 100–1000$\times$ above the median. This large gap means even noisy estimates at small N produce stable rankings. We verify in §C.2 that N=50 yields $\geq$97% overlap with the N=100 reference circuit across all model sizes.

D.5 Cumulative Update Alignment

Since $\mathbf{B} = 0$ at initialization, the trained $\mathbf{B}$ weights are the total displacement $\Delta\mathbf{B}$. For a given circuit mask (top-k elements by score) and a size-matched random mask, we measure what fraction of $\|\Delta\mathbf{B}\|^2$ falls on each subset. The ratio Circuit/Random tells us whether training concentrates weight changes onto the elements that circuit discovery identified. A ratio above 1 means circuit elements move more than expected; below 1 means they move less.

Table 16 reports the ratio of per-element update energy (mean squared displacement over training) for circuit elements vs. random elements at k=10K.

Table 16: **Cumulative update alignment** (k=10,000, circuit/random energy ratio). Under SFT, circuit elements move less than random ($< 1\times$); under GRPO, they move more ($> 1\times$). The reversal holds across models, scoring methods, and tasks.

		GSM8K		MATH-500	
Model	**Circuit**	**SFT**	**GRPO**	**SFT**	**GRPO**
Qwen2.5-1.5B	$\hat{S}$	0.41$\times$	**2.04**$\times$	0.41$\times$	**1.38**$\times$
Qwen2.5-1.5B	$\hat{F}$	0.40$\times$	**2.05**$\times$	0.46$\times$	1.35$\times$
Qwen2.5-3B	$\hat{S}$	0.24$\times$	**1.98**$\times$	0.39$\times$	**1.60**$\times$
Qwen2.5-3B	$\hat{F}$	0.26$\times$	**1.62**$\times$	0.44$\times$	1.53$\times$

Under SFT, coherent gradients spread updates broadly, so any subset accumulates comparable displacement and circuit elements are not preferentially updated (0.24–0.41$\times$). Under GRPO, incoherent gradients produce meaningful cumulative displacement only at elements with consistent

gradient direction, exactly what $\hat{S}$ and $\hat{F}$ identify. Circuit elements move 1.4–2.1$\times$ more than random, confirming that training concentrates learning onto the discovered circuit.

Weight-statistic scores are unreliable. The Wanda scoring function, a product of pretrained weight magnitude and mean gradient (§F.1.1), is inconsistent across architectures (Table 34). Weight-magnitude methods do not reliably identify the elements onto which GRPO concentrates updates, further motivating gradient-based scoring.

D.6 Gradient Structure: Full Results

Table 1 reports gradient structure on MATH across three Qwen2.5 scales. Table 17 provides the measurements on GSM8K, where the same contrast holds, most sharply at 1.5B and 3B.

Table 17: **Gradient structure under SFT vs. GRPO** (matched gradient steps at initialization). Three metrics across Qwen2.5 base and instruct models on GSM8K. The contrast is sharpest at 1.5B and 3B; at 7B and on the instruct model the two regimes are more similar.

	Eff. Gradient Rank		Cosine Similarity		Net / Step Norm	
Model	**SFT**	**GRPO**	**SFT**	**GRPO**	**SFT**	**GRPO**
Qwen2.5-1.5B	2.3	24.4	0.87	0.08	0.93	0.34
Qwen2.5-3B	4.0	6.6	0.78	−0.04	0.88	0.22
Qwen2.5-7B	1.5	2.7	0.96	0.41	0.97	0.78
Qwen2.5-7B-Instruct	12.6	5.9	0.54	0.27	0.73	0.52

Per-step versus accumulated gradient norm. A small Net / Step Norm can mean two different things: the steps were small, or the steps were large and cancelled. Table 18 separates them on MATH. GRPO's net update is 1.4–6.6$\times$ smaller than SFT's at every scale, while its per-step norm is comparable or larger. At 1.5B and 3B the two even move in opposite directions: GRPO takes larger steps and moves the weights less. The net update shrinks far faster than the steps do, so the steps must be cancelling.

Table 18: **Per-step versus accumulated gradient norm** (Qwen2.5, MATH, matched gradient steps at initialization). $\overline{\|g_t\|}$ is the average gradient length; $\|\bar{g}\|$ is the length of the average gradient. Their ratio is the Net / Step Norm column of Table 1.

	$\overline{\|g_t\|}$ (per-step)			$\|\bar{g}\|$ (accumulated)		
Model	**SFT**	**GRPO**	**GRPO/SFT**	**SFT**	**GRPO**	**SFT/GRPO**
Qwen2.5-1.5B	2.70	6.24	2.31$\times$	2.42	1.77	1.37$\times$
Qwen2.5-3B	1.88	2.84	1.51$\times$	1.51	0.63	2.39$\times$
Qwen2.5-7B	2.22	1.73	0.78$\times$	2.06	0.31	6.61$\times$

D.7 Sensitivity to the Scoring Objective

Circuit discovery scores $\mathbf{B}$ with the supervised cross-entropy gradient even when the downstream task is trained with RL (§E.5). This is what makes discovery cheap: it needs no rollouts and no reward evaluations. It also raises two questions, which we answer here on Qwen2.5-1.5B with GSM8K. Does the gradient structure we report depend on GRPO's particular advantage estimator? And would scoring with an RL objective select different entries of $\mathbf{B}$?

The structure does not depend on the advantage estimator. GRPO centers rewards on the mean of the sampled group. Table 19 replaces that with a leave-one-out baseline, $\mathrm{adv}_k = r_k - \mathrm{mean}(r_{j\neq k})$, the estimator used by RLOO. We measure gradient structure at initialization and do not train with it. The resulting gradient is close to the GRPO gradient and nothing like the SFT gradient: effective rank

22.2 against 25.3 for GRPO and 2.2 for SFT, and consecutive-step cosine 0.13 against 0.11 and 0.88. The cancellation described in §3.3 is therefore not an artifact of group-relative centering. We did not measure an actor-critic method with a learned value function, so we make no claim about PPO.

Table 19: **Gradient structure under a leave-one-out baseline** (Qwen2.5-1.5B, GSM8K, 80 gradient steps at initialization, 8 generations per prompt). The three regimes use disjoint prompt slices of the same split, so the SFT and GRPO columns are an independent resampling of the Qwen2.5-1.5B row of Table 17 and differ from it slightly.

Metric	SFT	GRPO	Leave-one-out
Effective gradient rank	2.2	25.3	22.2
Top-1 singular value explained	89%	23%	26%
Cosine similarity	0.88	0.11	0.13
Per-step norm $\overline{\|g_t\|}$	4.70	7.37	8.67
Net / Step Norm	0.94	0.34	0.38

Different objectives select the same modules, not the same entries. Table 20 scores the same model with cross-entropy and with two RL objectives and compares the resulting top-10K circuits. Element-level agreement is low: a cross-entropy circuit shares a small minority of its entries with an RL-scored circuit, though that is still hundreds of times above chance, so the circuits are far from independent without being the same set. Module-level agreement is much closer: every objective places most of the budget on V and gate (Table 21). DAPO is omitted because it is not an independent comparison. At initialization the policy ratio is exactly 1, so DAPO's clipped weight reduces to GRPO's and the two circuits coincide by construction.

Table 20: **Circuit overlap across scoring objectives** (Qwen2.5-1.5B, GSM8K, k=10K, 200 prompts, 8 generations). Two independent runs. Chance overlap is $10\text{K}/20.6\text{M} = 0.048\%$.

	Overlap (%)		Fold above chance	
Objective pair	Run 1	Run 2	Run 1	Run 2
GRPO ∩ Dr.GRPO	30.1	32.7	622×	674×
CE ∩ Dr.GRPO	13.9	16.2	287×	335×
CE ∩ GRPO	11.4	12.9	235×	266×

Table 21: **Module split of the circuit by scoring objective** (Qwen2.5-1.5B, GSM8K, k=10K). Percentage of the selected entries falling in each projection. Under every objective V takes the largest share of the budget and gate the second largest.

Objective	V	K	Q	O	Gate	Up	Down
CE	49.9	6.6	1.4	11.5	15.6	13.4	1.6
GRPO	56.5	6.2	4.6	4.5	16.1	11.1	1.0
Dr.GRPO	53.8	8.1	2.7	11.3	13.2	10.3	0.5

We read the low element-level overlap as multiplicity rather than instability. The same pattern appears when only the random seed changes: the individual entries move, but the modules and layers do not (§C.5); circuits discovered on different datasets likewise share 17–37% of their entries (§D.3). The claim the paper rests on is existential, that informed placement works where random placement fails, and several distinct informed sets satisfying it is consistent with that claim.

One measurement does not follow this pattern. Depth allocation shifts with the objective: 81.1% of the cross-entropy circuit falls in the first 40% of layers, against 59.0% for the advantage-weighted circuit. Of what we measured, where in depth the budget lands is the least stable across objectives.

D.8 KNOCKOUT SWEEP VALIDATION

The training experiments (§4.3) demonstrate that circuit-placed parameters learn effectively while random-placed parameters do not. Knockout sweeps provide independent, training-free confirmation: we zero out the top fraction of trained **B** entries by score, merge the residual LoRA into the base model, and evaluate accuracy. If circuits identify functionally critical parameters, circuit-ordered knockout should degrade accuracy faster than removing entries at random.

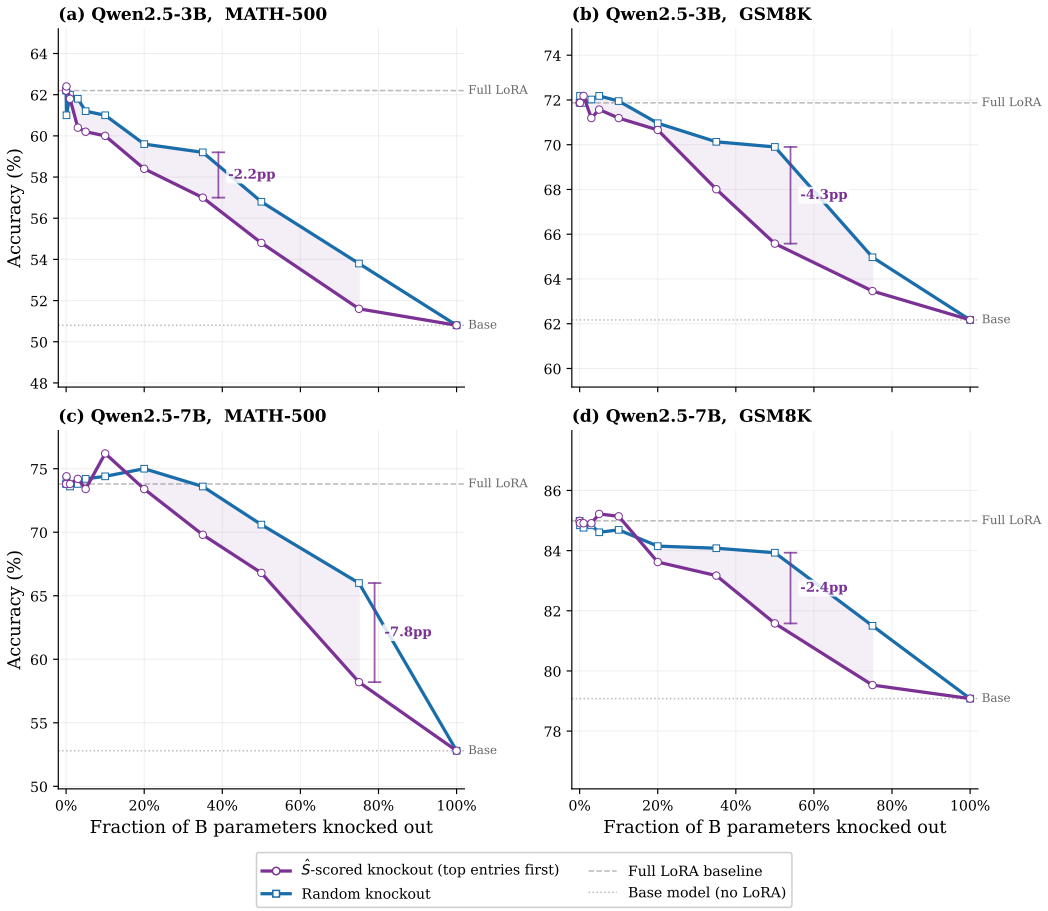

Figure 13: $\hat{S}$ **knockout sweep** (3B/7B × MATH/GSM8K). $\hat{S}$-ranked entries degrade accuracy faster than random, but with smaller gaps than $\hat{F}$ (Fig. 3): the peak is −7.8pp (7B MATH at 75%) vs. −12.2pp for $\hat{F}$.

$\hat{S}$ knockout (Fig. 13) shows the same qualitative pattern with uniformly smaller gaps on 3B GSM8K. This ranking matches the training performance comparison (Table 3) and reflects the scores' statistical properties. $\hat{S}$ captures directional consistency ($|\mathbb{E}[g]|$), while $\hat{F}$ captures sensitivity ($\mathbb{E}[g^2] = \mu^2 + \sigma^2$). Elements with large but sign-varying gradients score high on $\hat{F}$ but low on $\hat{S}$. The knockout results confirm that these high-variance elements are functionally important, explaining why $\hat{F}$ identifies more concentrated circuits.

Removing cross-module scale destroys the score. Global top-k compares raw gradient statistics across modules, so a module with larger-scale gradients contributes more entries. To test whether that scale is a confound or the signal, we z-score each tensor before taking the global top-k, which removes cross-module scale but leaves the ranking within each module intact, then knock out the selection this produces. On both Qwen2.5-7B and Qwen2.5-3B on MATH, the normalized criterion is indistinguishable from random at every fraction, while knocking out the raw $\hat{F}$ selection returns

Table 22: **Knockout sweep summary** across four configurations. LoRA effect is the accuracy span between the full adapter and the base model. Peak gap is the maximum difference between circuit-ordered and random knockout at any fraction. $\hat{F}$ consistently produces larger gaps than $\hat{S}$, and the gap scales with LoRA effect size.

		Full LoRA	Base	LoRA	Peak gap (pp)	
Model	**Task**	**(%)**	**(%)**	**effect**	$\hat{F}$	$\hat{S}$
Qwen2.5-3B	MATH	62.8	49.4	13.4	−7.2	−2.2
Qwen2.5-3B	GSM8K	72.0	62.2	9.8	−5.0	−4.3
Qwen2.5-7B	MATH	74.4	52.8	21.6	−12.2	−7.8
Qwen2.5-7B	GSM8K	85.2	79.5	5.7	−2.5	−2.4

the model to base accuracy. Cross-module scale is therefore part of what the score measures, not an artifact of it.

D.9 Module-Type Dissociation: Readers and Writers

The circuit heatmaps (§D.1) show which modules are selected; this section asks why. Two analyses, gradient sign consistency and SVD alignment, reveal that selected parameters fall into two functional classes with distinct structural signatures: **attention readers** (V, K) and **residual writers** (O, Down), following the work of (Elhage et al., 2021).

D.9.1 Gradient Sign Consistency

For each element of $\mathbf{B}$, we compute the fraction of training examples (N=100) on which the gradient sign agrees with the majority sign, then average across all elements of a given module type. Higher consistency means the loss surface pushes that element in a stable direction, a prerequisite for accumulating meaningful displacement under noisy RL gradients (§3.3).

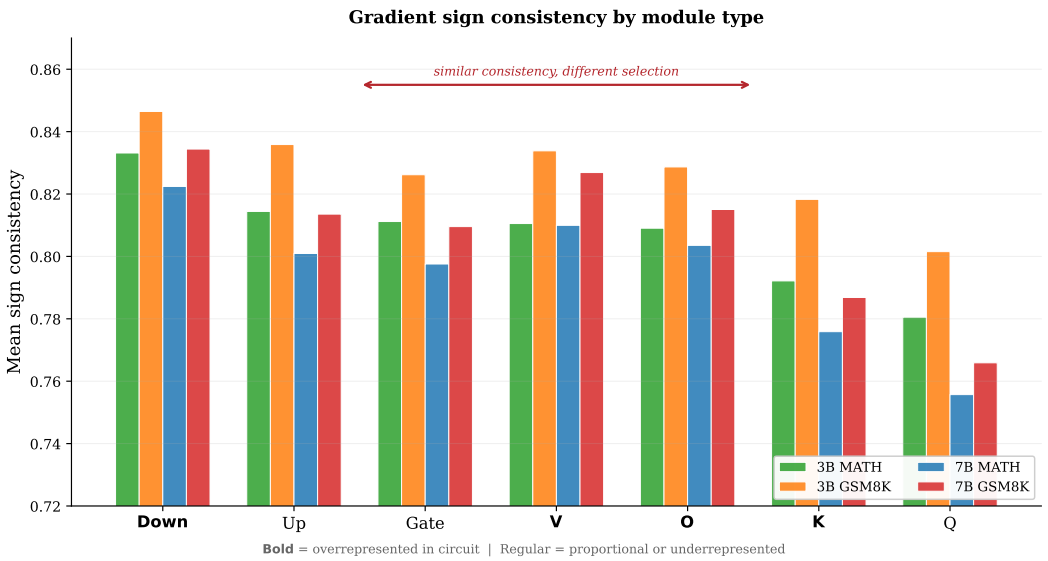

Figure 14: **Gradient sign consistency by module type** (3B/7B × MATH/GSM8K). Modules ordered by descending consistency; bold labels indicate modules overrepresented in circuits. Down is consistently highest (0.82−0.85). Gate and Up achieve consistency comparable to the selected modules V and O, yet are underrepresented in circuits. Consistency is necessary but not sufficient for selection.

Fig. 14 shows a stable hierarchy, Down $>$ {V, Up, Gate, O} $>$ K $>$ Q, with the middle group tightly clustered. The ordering holds across model scale and task; only the absolute level shifts. Gate and Up are as consistent as the selected modules V and O yet are rarely selected, and Q, the least consistent, is never overrepresented. Consistency is therefore necessary but not sufficient for selection.

D.9.2 SVD Alignment

Sign consistency measures how reliably each module's gradient points in one direction; SVD alignment measures where that direction lies relative to the pretrained weight structure. For each trained LoRA adapter $\Delta\mathbf{W} = \mathbf{BA}$, we compute the SVD of the base weight $\mathbf{W}_0 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ and measure two quantities:

- **Left alignment**: how much of $\Delta\mathbf{W}$'s column space overlaps with the top-r left singular vectors of $\mathbf{W}_0$. High left alignment means the adapter modifies the same output directions that the pretrained weight already emphasizes.
- **Spectral ratio**: the fraction of $\|\Delta\mathbf{W}\|_F^2$ captured by the top-r singular values of $\Delta\mathbf{W}$. A high spectral ratio means the update is spectrally concentrated and effectively low-rank.

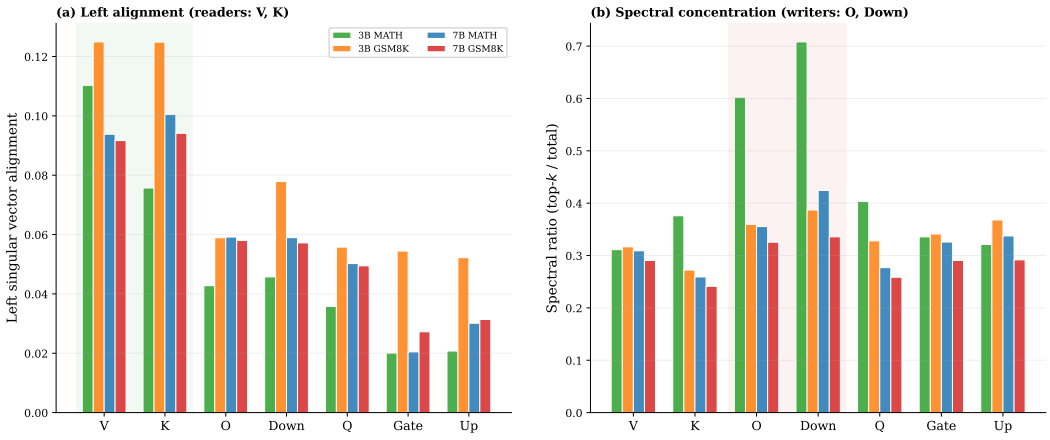

Figure 15: **SVD alignment by module type** (3B/7B $\times$ MATH/GSM8K). (a) Left singular vector alignment: V and K dominate, identifying them as attention readers whose updates reinforce existing information-selection directions. (b) Spectral concentration: O and Down dominate, identifying them as residual writers whose updates are low-rank. Shaded bands highlight the dominant pair in each panel.

Fig. 15 splits the modules into **attention readers** (V, K), whose updates align with the pretrained weight's dominant directions, and **residual writers** (O, Down), whose updates are low-rank. V and K sit at the input of attention, so an aligned update adjusts attention inside the subspace the model already uses. O and Down write back into the residual stream, so a concentrated update steers the output along a few directions. Gate and Up modify activations inside the MLP; they show neither signature, and neither does Q.

A stable gradient direction is what lets an element accumulate displacement under RL, and Down, V, Up, Gate and O all have one. Among those, the elements selected are the ones that also sit at the residual stream, reading from it or writing to it. Gate and Up pass the first filter and fail the second, because their updates change an internal MLP activation rather than what enters or leaves the stream.

D.10 Cross-Architecture Validation: Llama-3.2-3B

The analyses in §D.9 use Qwen2.5 models exclusively. To test whether the Readers/Writers dissociation is architecture-specific or reflects a general property of transformer fine-tuning under RL, we replicate all three measurements on Llama-3.2-3B-Instruct trained with GRPO on MATH.

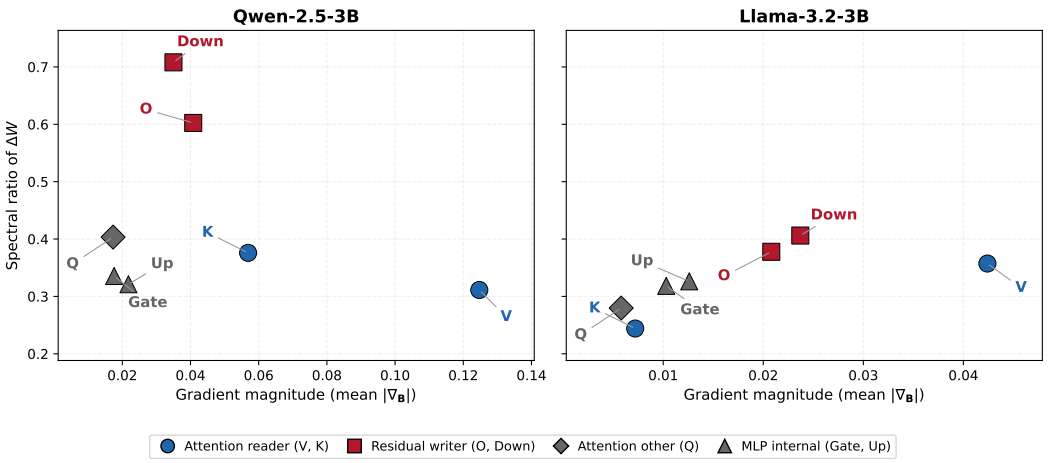

Figure 16: **Cross-architecture dissociation.** Each point is one module type, positioned by its gradient magnitude (mean $|\nabla_{\mathbf{B}}|$, x-axis) and spectral concentration of ΔW (y-axis). In both Qwen-2.5-3B and Llama-3.2-3B, residual writers (O, Down; red squares) cluster at high spectral ratio while attention readers (V, K; blue circles) receive the strongest gradient signal. The spatial separation of these two functional roles is preserved across model families.

Table 23: **Module-level analysis: Llama-3.2-3B-Instruct on MATH** (GRPO, 28 layers). Three complementary measurements per module type, averaged across all layers. The Readers/Writers dissociation transfers from Qwen to Llama: V and K show the highest left alignment (attention readers), O and Down show the highest spectral concentration (residual writers), and Gate/Up show high sign consistency but neither structural signature, the same anomaly observed in Qwen (Fig. 14).

	Sign Consistency			SVD Alignment	
Module	**Mean**	**Frac >0.8**	**SNR**	**Left Align**	**Spec. Ratio**
V	0.780	0.482	0.354	**0.066**	0.357
K	0.763	0.425	0.313	**0.081**	0.244
Q	0.761	0.419	0.301	0.049	0.280
O	0.778	0.476	0.352	0.056	**0.378**
Gate	0.792	0.518	0.358	0.022	0.318
Up	0.785	0.498	0.353	0.024	0.326
Down	0.787	0.505	0.371	0.061	**0.406**

Fig. 16 visualizes the dissociation: writers occupy the upper region (high spectral ratio) while readers extend rightward (high gradient magnitude), with the same spatial layout in both architectures. Table 23 confirms that all three structural signatures transfer:

1. **Sign consistency**: The same Gate/Up anomaly appears. Gate and Up match or exceed V and O, yet V, not Gate or Up, receives the strongest gradient signal (Table 24). Consistency is necessary but not sufficient, exactly as in Qwen.
2. **Left alignment**: K and V have the highest alignment with the pretrained weight's dominant singular vectors, confirming their role as attention readers that refine existing information-selection directions.
3. **Spectral concentration**: Down and O produce the most concentrated (low-rank) updates, confirming their role as residual writers that steer output through narrow perturbations.

The dissociation is therefore not an artifact of the Qwen family: V still dominates gradient magnitude and K still shows the highest left alignment on Llama. The functional roles, reading from vs. writing to the residual stream, are determined by the transformer's computational graph, not by any one model's design choices.

Table 24: **Gradient magnitude by module type** (Llama-3.2-3B-Instruct, MATH, $\mathbf{B}$-matrix only). Mean $|\nabla_{\mathbf{B}}|$ averaged across all layers. V dominates ($3-4\times$ over Gate/Up), followed by O and Down, the same ordering as Qwen2.5, and consistent with the Readers/Writers interpretation: modules at the residual stream interface receive the strongest gradient signal under RL.

Module	$\mathbb{E}[\lvert\nabla_{\mathbf{B}}\rvert]$	Role
V	0.042	Attention reader
Down	0.024	Residual writer
O	0.021	Residual writer
Up	0.013	MLP internal
Gate	0.010	MLP internal
K	0.007	Attention reader
Q	0.006	—

D.11 Weight Norm and Behavior Change Dynamics

Here we continue the analysis on the dynamics between how selected neurons change and how much model predictions change, across training checkpoints (top) and across model layers (bottom), but extend it to Llama models and use KL instead of Jaccard. This was first discussed in §5.2. We repeat the setup below for reader convenience.

Setup. We focus on the dynamics for Qwen2.5-1.5B/3B/7B and Llama-3.2-3B/3.1-8B (GRPO on MATH) over 100 test examples from MATH, focused only on the processing of the prompt and problem definition, compared to the respective base models. We report $1-$ *top-25 overlap*, where overlap is the Jaccard overlap of the top-25 predicted tokens between the fine-tuned and base model, averaged over positions and examples. The *top* row x-axis reports the effective norm of the LoRA trainable weights, $\|\mathbf{B}_{\text{nz}}\|_2 \cdot \alpha/r$. Since $\mathbf{B} = 0$ at initialization, the non-zero $\mathbf{B}_{\text{nz}}$ weights are the total displacement. The y-axis captures some of the behavioral change, as reflected by the change in output token distributions. For top, this is the projection across the vocabulary across training checkpoints; for bottom, these values are across model layers (x-axis) using logit lens (Belrose et al., 2023), projecting residual hidden states to the vocabulary logits.

The two metrics we use to capture how much the model's behavior is changing are:

1. $1-$ *top-25 overlap*, where overlap is the Jaccard-style fraction of top-25 predicted tokens shared between each pair of models. Higher values indicate greater divergence. A value of 0 indicates identical behavior to the base model.
2. KL divergence between each pair of models.

Both metrics are averaged across positions and examples.

In Fig. 18, we replicate the analysis from Fig. 4, using KL divergence instead of top-25 overlap.

Table 25 shows the final-layer (final training checkpoint) output distribution metrics across a number of settings. This is over 100 test examples from MATH, focused only on the processing of the prompt and problem definition, for Qwen2.5-1.5B/3B/7B and Llama-3.2-3B/3.1-8B.

More Findings. Beyond the findings discussed in the main body of the paper, Table 25 suggests two further takeaways: (1) selected circuits find new solutions rather than replicating LoRA behavior; (2) random circuits do not diverge far from the base model.

Table 25: **Distributional divergence at final checkpoints (all families).** Models Qwen2.5-1.5B, -3B, -7B and Llama-3.2-3B, -3.1-8B are marked by Q- and L-, respectively. We report the $\mathrm{KL}(M_1 \| M_2)$ and $1-$Top-25 overlap at the final transformer layer between different model pairs (M_1, M_2). This is approximated from top-25 token distributions over 100 MATH-500 prompts and problem texts.

	KL divergence					**1 – Top-25 overlap**				
	Q-1.5B	Q-3B	Q-7B	L-3B	L-8B	Q-1.5B	Q-3B	Q-7B	L-3B	L-8B
Full-LoRA vs $\hat{F}$	0.161	0.154	0.180	0.019	0.244	0.185	0.153	0.150	0.117	0.274
Full-LoRA vs $\hat{S}$	0.186	0.257	0.166	0.021	0.290	0.204	0.185	0.142	0.121	0.277
Full-LoRA vs Random	0.122	0.033	0.273	0.006	0.337	0.142	0.069	0.192	0.030	0.312
Full-LoRA vs Pre-trained	0.125	0.047	0.290	0.007	0.341	0.144	0.085	0.197	0.033	0.314
Pre-trained vs $\hat{F}$	0.283	0.240	0.154	0.029	0.074	0.241	0.196	0.122	0.134	0.172
Pre-trained vs $\hat{S}$	0.324	0.339	0.145	0.031	0.075	0.263	0.226	0.132	0.136	0.180
Pre-trained vs Random	0.004	0.012	0.008	0.001	0.001	0.013	0.027	0.020	0.014	0.014

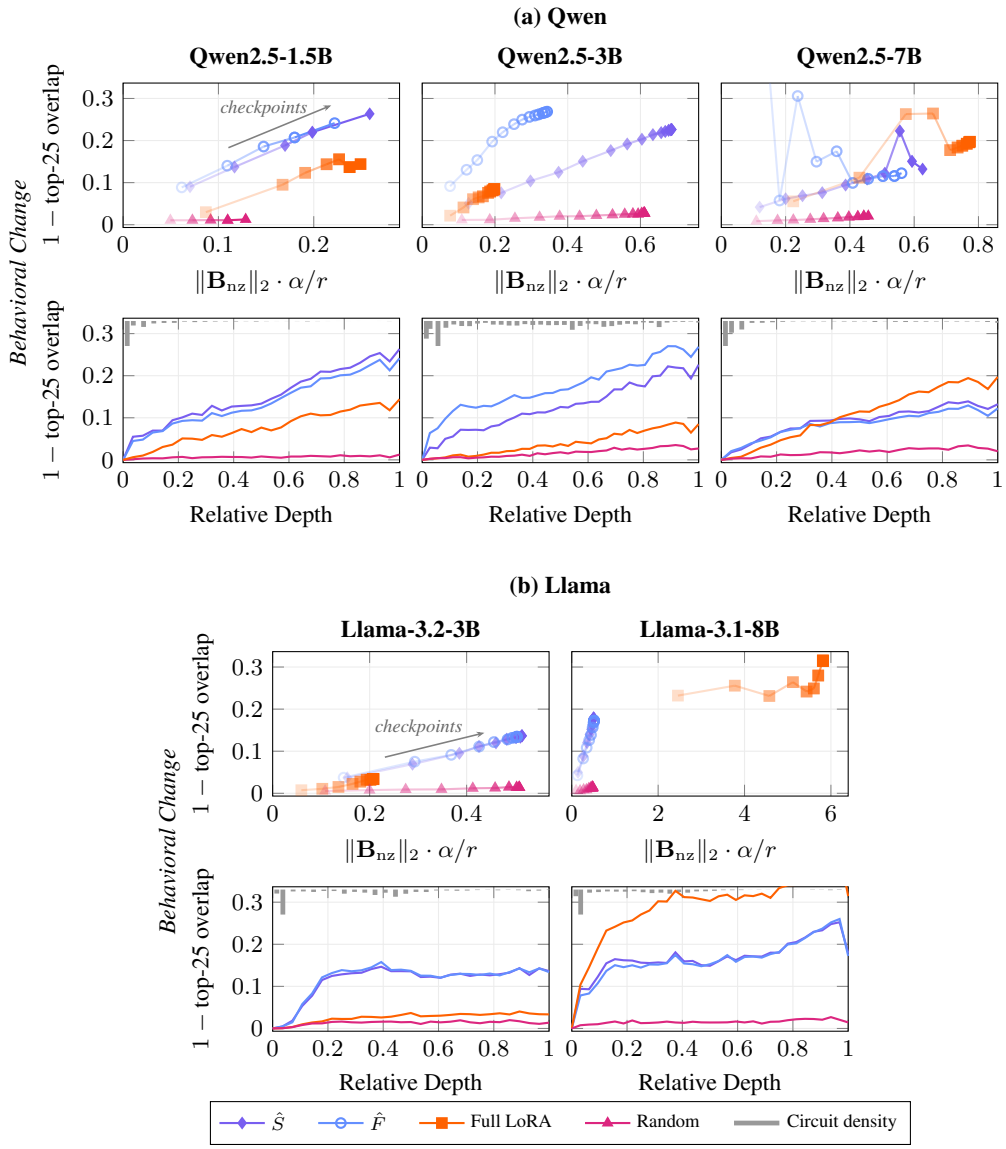

Figure 17: **Divergence vs. effective update norm** (a: Qwen2.5-1.5B/3B/7B, MATH-500, GRPO); (b: Llama-3.2-3B/Llama-3.1-8B, MATH-500, GRPO). For both rows, the y-axis is $1 - $ *top-25 token overlap*, measuring behavioral change via logit lens relative to the base model. Top: each trajectory traces training checkpoints, opacity increases with step. x-axis: $\|\mathbf{B}_{\text{nz}}\|_2 \cdot \alpha/r$, the scaled norm of non-zero $\mathbf{B}$ weights. Bottom: per-layer divergence at the final checkpoint. x-axis: relative layer depth. Gray bars report the relative circuit density.

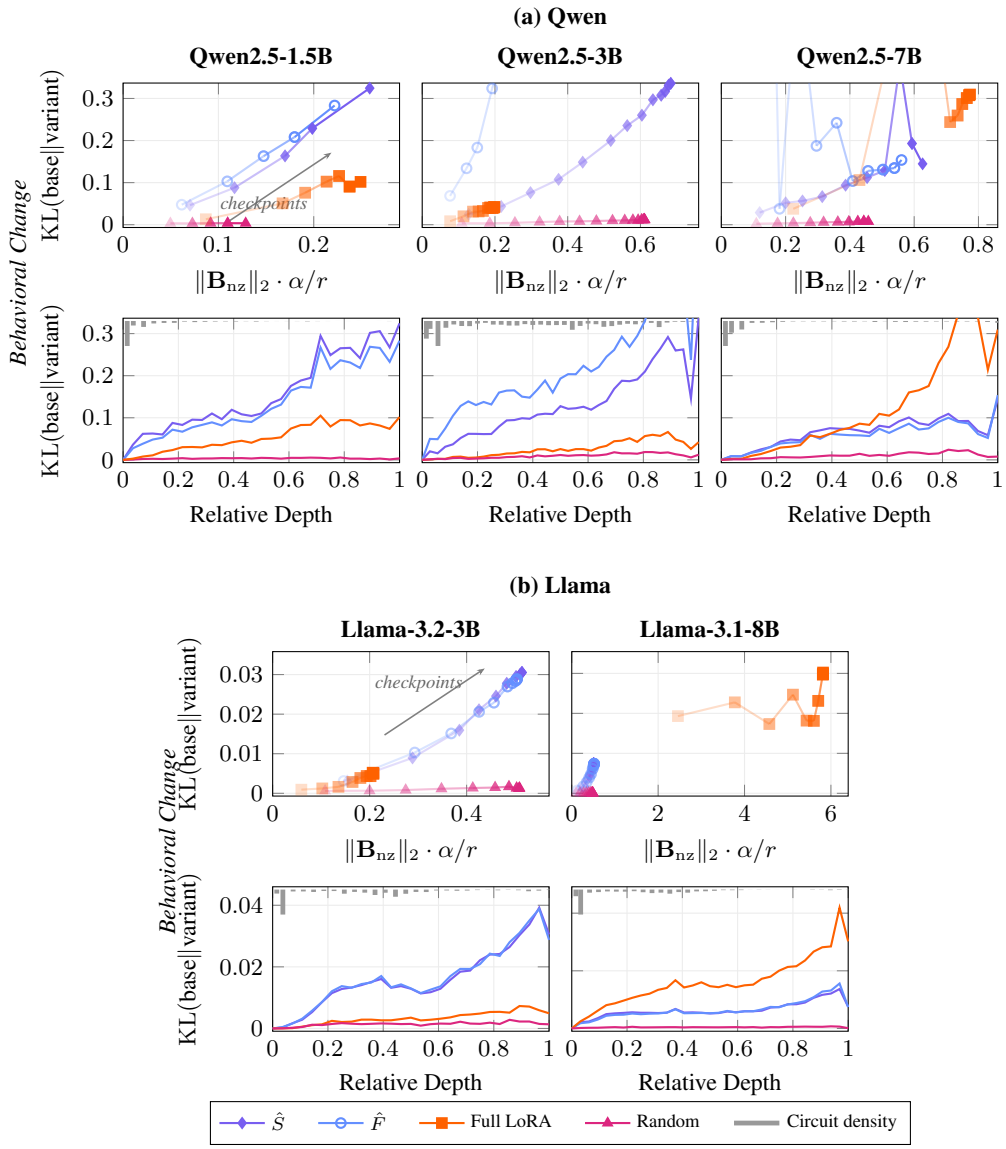

Figure 18: **Divergence vs. effective update norm** (a: Qwen2.5-1.5B/3B/7B, MATH-500, GRPO); (b: Llama-3.2-3B/Llama-3.1-8B, MATH-500, GRPO). For both rows, the y-axis is the KL divergence from the base model, measuring behavioral change via logit lens. Top: each trajectory traces training checkpoints, opacity increases with step. x-axis: $\|\mathbf{B}_{\text{nz}}\|_2 \cdot \alpha/r$, the scaled norm of non-zero $\mathbf{B}$ weights. Bottom: per-layer divergence at the final checkpoint. x-axis: relative layer depth. Gray bars report the relative circuit density.

E HYPERPARAMETERS

CONTENTS

Common Setup . . . . . E.1, p. 42
GRPO Training: Qwen2.5 Base Models . . . . . E.2, p. 42
GRPO Training: Llama Instruct Models . . . . . E.3, p. 43
Supervised Fine-Tuning . . . . . E.4, p. 43
Circuit Discovery . . . . . E.5, p. 45

E.1 COMMON SETUP

All experiments run on NVIDIA RTX PRO 6000 GPUs (96 GB each). Qwen2.5 base-models GRPO uses TRL (von Werra et al., 2020) with PEFT in bf16 across two GPUs via Accelerate. Llama instruct GRPO uses Unsloth (Daniel Han & team, 2023) with 4-bit quantization on a single GPU. SFT uses the HuggingFace SFTTrainer on a single GPU.

All main results (Tables 2 and 3) use greedy decoding: temperature 0.0, no nucleus sampling. We use vllm and `lm_eval_harness` (Gao et al., 2024a) for SFT.

E.2 GRPO TRAINING: QWEN2.5 BASE MODELS

Table 27 reports the configuration shared across all Qwen2.5 base-model GRPO runs (GSM8K and MATH-500). Where 7B differs from 1.5B/3B, we note the deviation. Table 26 reports the LoRA rank used for each model–task combination. The 1.5B runs and 3B GSM8K use rank 32; 3B MATH and the 7B runs use rank 8. In all cases, LoRA is applied to every linear layer in the transformer block (q, k, v, o, gate, up, down projections), with $\mathbf{A} \sim \mathcal{N}(0, 2/d)$ and $\mathbf{B} = \mathbf{0}$.

Table 26: **LoRA rank and α per GRPO configuration.** Sparse circuits scale $\alpha = 32 \times r$ to compensate for the reduced number of active parameters.

Model	Task	r	α (full)	α (sparse)
Qwen2.5-1.5B	GSM8K	32	32	1024
Qwen2.5-1.5B	MATH	32	32	1024
Qwen2.5-3B	GSM8K	32	32	1024
Qwen2.5-3B	MATH	8	32	256
Qwen2.5-7B	GSM8K	8	32	256
Qwen2.5-7B	MATH	8	32	256

Table 27 reports the remaining training configuration, shared across all Qwen2.5 base-model GRPO runs. All runs train for 2,000 steps.

Loss types. DAPO (Yu et al., 2026) removes the KL penalty ($\beta = 0$) and applies clipped importance weights. DR-GRPO (Liu et al., 2025) additionally length-normalizes the per-token log-probabilities to remove the bias toward shorter completions. We use DR-GRPO for 3B MATH, 7B GSM8K, and 7B MATH; while 1.5B and 3B GSM8K train with DAPO.

Alpha scaling for sparse circuits. When only k of the $\mathbf{B}$-matrix elements are trainable, the effective adapter norm shrinks proportionally. To compensate, we scale α to $32 \times r$ (Table 26). This scaling is applied uniformly to all sparse and random runs; full LoRA runs use $\alpha = 32$. Measured on trained Qwen2.5-1.5B adapters, the shrinkage is large: the raw update $\|\mathbf{BA}\|$ of a k=10K circuit is 20–29$\times$ smaller than dense full LoRA's (0.0046–0.0048 against 0.135 on GSM8K, 0.0065–0.0088 against 0.174 on MATH). Scaling α by r therefore returns the effective update $\|(\alpha/r)\,\mathbf{BA}\|$ to full LoRA's range rather than pushing it past. The same α is used for informed and random placement, so it cannot account for the difference between them. We ablate alpha scaling in §C.8.

Reward function. All mathematics GRPO runs use binary correctness rewards. For GSM8K, the model is prompted with "Put your final numerical answer after ####" and the answer is extracted

Table 27: **GRPO training hyperparameters (Qwen2.5 base models).** Where 7B differs, the deviation is noted.

Hyperparameter	1.5B / 3B	7B
Optimization		
Loss type	DAPO	DR-GRPO
KL coefficient (β)	0.0	0.0
Learning rate	1×10^{-6}	1×10^{-6}
LR scheduler	constant	constant
Warmup steps	25	25
Optimizer	AdamW	AdamW
Adam (β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)
Weight decay	0.0	0.0
Max gradient norm	0.1	0.1
Precision	bf16	bf16
GRPO generation		
Generations per prompt	4	4
Max completion length	1024	1024
Sampling temperature	1.0	1.0
Batching and schedule		
Per-device batch size	1	1
Gradient accumulation	4	4
Gradient checkpointing	yes	

and compared symbolically using `math_verify`. For MATH, the prompt requests the answer in `\boxed{}` format and the same `math_verify` library grades symbolic equivalence.

Checkpoint selection. We evaluate every saved checkpoint (every 200 steps on MATH, final checkpoint on GSM8K) on the corresponding test set with greedy decoding and report the best checkpoint per method. All methods within a comparison use the same evaluation protocol. Best and final checkpoints are close in practice: across the Qwen2.5 base-model GRPO runs with at least five evaluated checkpoints, the best checkpoint exceeds the final one by at most 1.8pp (median 0.35pp), for full LoRA and random placement as well as for circuits.

Training data. GSM8K uses the standard 7,473-example training split (Cobbe et al., 2021). MATH uses all 7,500 training examples from EleutherAI/hendrycks_math (7 subjects concatenated) (Hendrycks et al., 2021b).

E.3 GRPO Training: Llama Instruct Models

Llama instruct models (Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct) use BNPO loss (Xiao et al., 2026) throughout, with 4-bit quantized base weights via Unsloth (Daniel Han & team, 2023). All runs train for 2,000 steps with checkpoints saved every 200 steps. Table 28 reports the full configuration.

E.4 Supervised Fine-Tuning

All SFT experiments use the HuggingFace TRL SFTTrainer (von Werra et al., 2020) with AdamW optimizer, linear warmup over 20 steps, per-device batch size 1, max sequence length 1024. Training data is the cleaned version of Alpaca (Taori et al., 2023).

Default configuration. LoRA rank 32, $\alpha = 32$, applied to all linear layers. Learning rate 2×10^{-4}, no weight decay, 4 gradient accumulation steps (effective batch size 4), 1,000 training steps.

Per-model deviations. Table 29 reports deviations from the default. Larger models (Qwen2.5-7B, Llama-3.1-8B) and all Llama models required hyperparameter adjustments. For Llama-3.2-1B/3B,

Table 28: **GRPO training hyperparameters (Llama instruct models).**

Hyperparameter	3B	8B
LoRA configuration		
Rank (r)	32	32
α	32	32
Target modules	q, k, v, o, gate, up, down projections	
Quantization	4-bit (Unsloth)	
Optimization		
Loss type	BNPO	BNPO
KL coefficient (β)	0.002	0.002
Learning rate	5×10^{-6}	2×10^{-6}
LR scheduler	cosine	cosine
Warmup ratio	5%	5%
Optimizer	AdamW 8-bit	AdamW 8-bit
Adam (β_1, β_2)	(0.9, 0.95)	(0.9, 0.95)
Weight decay	0.1	0.1
Max gradient norm	0.1	0.1
GRPO generation		
Generations per prompt	4	4
Max completion (GSM8K)	1024	1024
Max completion (MATH)	—	2048
Sampling temperature	1.0	1.0
Batching		
Per-device batch size	4	4
Gradient accumulation	4	4

adding weight decay (λ=0.01) stabilized training. For Qwen2.5-7B and Llama-3.1-8B at 100K budgets, we reduced the learning rate to 5×10^{-5} and weight decay of 0.01.

Llama-3.1-8B full LoRA baseline uses the default learning rate (2×10^{-4}), no weight decay; the circuit and random conditions use the values shown. All other models use the same hyperparameters across all placement methods

Table 29: **Per-model SFT hyperparameter deviations.** All models use rank 32, α=32, warmup 20, max sequence length 1024. "—" indicates the default applies (lr=2×10^{-4}, wd=0, grad. accum. 4, 1,000 steps).

Model	lr	wd	grad. accum.	steps
Qwen2.5-1.5B	—	—	—	—
Qwen2.5-3B	—	—	—	—
Qwen2.5-7B	5×10^{-5}	0.01	—	—
Qwen3-8B	—	—	8	2,000
Llama-3.2-1B	—	0.01	—	—
Llama-3.2-3B	—	0.01	—	—
Llama-3.1-8B	5×10^{-5}	0.01	8	—

Alpha scaling. For sparse circuits, α is scaled from 32 to 1024 ($= 32 \times r$) to compensate for the reduced effective adapter norm when only $k \ll |\mathbf{B}|$ elements are trainable. This matches the GRPO alpha scaling (§E.2). Full LoRA and random baselines at the main-table budgets (Table 2) use $\alpha = 32$.

E.5 Circuit Discovery

Circuit discovery runs N forward-backward passes at LoRA initialization on the training distribution to accumulate gradient statistics for the $\mathbf{B}$ matrix. Gradients are computed with the standard supervised cross-entropy loss, regardless of whether the downstream task uses RL or SFT. This is computationally efficient, requiring only 50 forward-backward passes, and produces circuits that transfer across both training regimes. Table 30 reports the configuration.

Table 30: **Circuit discovery hyperparameters.** The same configuration is used for all models and tasks.

Parameter	Value
Forward-backward passes (N)	50
Batch size	1
Max sequence length	1024
LoRA initialization	$\mathbf{A} \sim \mathcal{N}(0, 2/d)$, $\mathbf{B} = \mathbf{0}$
Loss	Cross-entropy (supervised)
Method	$\mathbf{B}$-zero (accumulate $\nabla_{\mathbf{B}}\mathcal{L}$ with $\mathbf{B} = \mathbf{0}$)
Precision	bfloat16
Scoring functions	
$\hat{S}$ (gradient magnitude)	$1/N \left\lvert \sum_{n=1}^{N} \nabla_{\mathbf{B}}^{(n)} \right\rvert$
$\hat{F}$ (diagonal Fisher)	$1/N \sum_{n=1}^{N} \left(\nabla_{\mathbf{B}}^{(n)} \right)^2$
Mask application	
Selection	Top-k elements globally across all layers
Masking mechanism	Gradient hook: $\nabla \mapsto \nabla \odot \mathbf{m}$ (binary mask)
Frozen layers	Layers with zero selected elements have `requires_grad=False`

The discovery dataset matches the GRPO training distribution (GSM8K train or MATH train). A single circuit discovery pass takes $<$10 seconds on any model up to 7B (Table 8).

OOD evaluation (pass@k). AMC and AIME evaluations (§B.2) generate n=64 completions per problem at temperature 0.7 with top-$p = 0.95$ and max 4,096 tokens. Pass@k is computed as the fraction of problems where at least one of the first k samples is correct. Majority vote (maj@k) takes the most common answer among k samples.

Grading. GSM8K answers are extracted after the `####` delimiter. MATH-500 answers are extracted from `\boxed{}`. Both are verified using `math_verify` for symbolic equivalence (e.g., $\frac{1}{2}$ matches 0.5).

Knockout sweep. Starting from a fully trained LoRA (merged into float16), we zero the top fraction of $\mathbf{B}$ entries by score and evaluate accuracy. Fractions tested: $\{0.1\%, 1\%, 3\%, 5\%, 10\%, 20\%, 35\%, 50\%, 75\%\}$. Random knockout uses seed 42 for reproducibility.

F Additional Experiments

Contents

Extended Scoring Function Comparison . . . . . F.1, p. 46
Additional Scoring Functions – Alternative Scoring Methods (GSM8K) – Row-Level Scoring (GSM8K) – IFEval – Consistency Summary.
Module-Restricted LoRA . . . . . F.2, p. 48
Additional Budget Sweeps . . . . . F.3, p. 49
Text-to-SQL . . . . . F.4, p. 49

This appendix reports experiments that supplement the main paper's evaluation. We compare additional scoring functions beyond $\hat{S}$ and $\hat{F}$, test element- vs. row-level selection granularity, restrict dense LoRA to the modules the scores favor, extend the instruction-following evaluation to larger budgets and additional benchmarks and models, and repeat the placement comparison under GRPO in a non-math domain.

F.1 Extended Scoring Function Comparison

The main paper focuses on two circuit scoring functions: gradient magnitude ($\hat{S}$) and diagonal Fisher ($\hat{F}$). Here we evaluate additional importance metrics under SFT across multiple model families and tasks, at both element-level and row-level selection granularities. All methods are computable at initialization from N=50 forward-backward passes in under 10 seconds (§C.4). We find that $\hat{S}$ and $\hat{F}$ are the most consistent across architectures; weight-statistic methods and row-level variants are unreliable. This motivates the main paper's focus on $\hat{F}$ and $\hat{S}$.

F.1.1 Additional Scoring Functions

All scoring functions operate on the LoRA $\mathbf{B}$ matrix at initialization ($\mathbf{B} = 0$). We distinguish two selection granularities:

Element-level methods select individual entries of $\mathbf{B}$:

- **Gradient magnitude** ($\hat{S}$): $\left|\frac{1}{N}\sum_{n=1}^{N} g_{n,ij}\right|$. Directional consistency of per-example gradients. Defined in §3.2.
- **Diagonal Fisher** ($\hat{F}$): $\frac{1}{N}\sum_{n=1}^{N} g_{n,ij}^2$. Gradient sensitivity (second moment). Defined in §3.2.
- **Weight magnitude** (Mag): $|[\mathbf{W}_0\mathbf{A}^\top]_{ij}|$. Selects $\mathbf{B}$ entries where the pretrained weight projected through $\mathbf{A}$ is largest. Requires no gradient computation.
- **Wanda**: $|[\mathbf{W}_0\mathbf{A}^\top]_{ij} \cdot \bar{g}_{ij}|$. Product of projected weight magnitude and mean gradient, adapted from the structured pruning method of Sun et al. (2024).

Row-level methods select entire rows of $\mathbf{B}$ (all r entries per hidden dimension train together):

- **Row Fisher / Row Magnitude / Row Wanda**: aggregate the element-level score across the rank dimension for each row, then select the top-k rows. This coarser granularity selects entire hidden dimensions rather than individual elements, making the selection independent of the randomly initialized $\mathbf{A}$ matrix. Row-level selection connects to structured pruning methods that operate on neurons or channels.

F.1.2 Alternative Scoring Methods (GSM8K)

Table 31 compares element-level scoring methods on GSM8K under both SFT and GRPO. All SFT rows use k=5K; the GRPO row uses k=10K.

Under SFT, $\hat{S}$ and $\hat{F}$ beat random on every model, while Mag and Wanda do not. Under GRPO the contrast sharpens: random is stuck at base (9.8%), Magnitude recovers only half the gap (34.2%), and Wanda reaches 44.8%, while $\hat{S}$ and $\hat{F}$ match full LoRA. This motivates the main paper's focus on gradient-based scores.

Table 31: **Element-level scoring on GSM8K** (accuracy, %). Gradient-based scores ($\hat{S}$, $\hat{F}$) consistently outperform random; weight-statistic methods (Mag, Wanda) are unreliable. Under GRPO the gap is extreme: random stays at base while circuits recover full LoRA.

Model	Regime	Base	Rand	$\hat{F}$	$\hat{S}$	Mag	Wanda	Full LoRA
Qwen2.5-1.5B	SFT	9.5	13.2	62.2	64.3	64.6	63.6	61.4
Qwen2.5-3B	SFT	62.1	62.9	66.3	67.5	66.9	62.5	**70.0**
Qwen3-4B	SFT	37.6	78.2	78.7	79.1	69.4	75.6	**81.4**
Qwen2.5-1.5B	GRPO	9.5	11.6	63.8	**65.2**	34.2	44.8	63.7

F.1.3 Row-Level Scoring (GSM8K)

Row-level methods select entire rows of $\mathbf{B}$ (r entries per hidden dimension), making selection independent of the random $\mathbf{A}$ initialization. Table 32 reports accuracy at three budgets under SFT and one budget under GRPO.

Table 32: **Row-level scoring on GSM8K** (accuracy, %). No single row method is reliable across models and regimes. Results below base are highlighted.

		Row Budget				
Model	**Method**	**1.6K**	**4.8K**	**16K**	**Base**	**Full LoRA**
SFT						
Qwen2.5-1.5B	Row Fisher	63.8	**68.2**	62.4	9.5	61.4
	Row Magnitude	63.2	67.6	65.7		
	Row Wanda	59.4	65.4	64.8		
Qwen2.5-3B	Row Fisher	59.1	64.3	66.8	62.1	70.0
	Row Magnitude	62.7	61.0	64.7		
	Row Wanda	66.9	59.4	65.3		
Qwen3-4B	Row Fisher	61.1	60.5	58.6	37.6	81.4
	Row Magnitude	**79.1**	76.8	62.3		
	Row Wanda	70.8	69.3	62.6		
GRPO ($k = 10$K)						
Qwen2.5-1.5B	Row Fisher		41.6		9.5	63.7
	Row Magnitude		33.1			
	Row Wanda		43.2			

Under SFT, row methods beat base on 1.5B but are erratic on 3B (several entries below base). Under GRPO, all row methods fall far short of element-level circuits (41–43% vs. 64%), confirming that the finer granularity of element selection is critical when gradients are noisy.

F.1.4 IFEval

Table 33 reports IFEval accuracy for Qwen2.5-1.5B under SFT at k=5K.

Table 33: **IFEval accuracy (%), Qwen2.5-1.5B** (SFT, k=5K). $\hat{F}$ recovers 71–77% of the full LoRA gain; random is indistinguishable from base.

Train Data	Base	Rand	$\hat{F}$	Mag	Wanda	Full LoRA
OpenHermes	17.0	16.8	**21.4**	21.4	18.5	22.7
Alpaca	17.0	16.3	**22.4**	20.9	18.3	24.6

F.1.5 CONSISTENCY SUMMARY

Table 34 tallies how often each scoring method outperforms random across all configurations in Tables 31 and 32.

Table 34: **Scoring method consistency across models and training regimes.** $\hat{S}$ and $\hat{F}$ are the only methods that never fail.

Method	Wins / Total	Failure Modes
Element-level		
$\hat{S}$	**4/4**	—
$\hat{F}$	**4/4**	—
Magnitude	3/4	Collapses under GRPO (34.2% vs. 9.8 rand)
Wanda	2/4	$\approx$ random on 3B SFT; partial under GRPO
Row-level		
Row Fisher	2/4	Below base on 3B SFT; 41.6% under GRPO
Row Magnitude	2/4	Below base on 3B SFT; 33.1% under GRPO
Row Wanda	2/4	Below base on 3B SFT; budget-sensitive

The pattern is clear: $\hat{S}$ and $\hat{F}$ are the only scoring methods that outperform random on every architecture and training regime tested. Weight-statistic methods (Magnitude, Wanda) work under SFT's stable gradients on some models but break under GRPO, where Magnitude collapses to 34.2%, worse than Wanda (44.8%) and far below circuits (64.2%). Row-level methods, which select entire hidden dimensions independent of $\mathbf{A}$, are inconsistent even under SFT (multiple entries below base on 3B) and fall far short under GRPO (33–43% vs. 64% for element-level $\hat{S}$).

This motivates the main paper's focus on $\hat{S}$ and $\hat{F}$: they are the only methods that reliably identify which parameters matter, regardless of whether the training signal is stable (SFT) or noisy (GRPO).

F.2 MODULE-RESTRICTED LORA

Row-level selection fails at a circuit-sized budget (§F.1.3); a coarser question is whether the modules the scores single out are useful targets on their own. Here we keep standard dense LoRA but apply it only to the two modules most overrepresented relative to their share of parameters (§5.1 and table 21): V and K for 1.5B and 3B, V and O for 7B. Every entry of those modules' $\mathbf{B}$ matrices is trainable, so this is neither a sparse method nor a substitute for circuits. It uses 8.7–12.5% of full LoRA's trainable parameters against a circuit's 0.02–0.33%, but it is far cheaper than adapting all modules.

Table 35: **LoRA restricted to the top-2 circuit modules** (GRPO accuracy, %). Dense LoRA on the two highest-budget modules, against dense LoRA on all seven. The last column is the trainable-parameter count as a fraction of full LoRA's ($\mathbf{A} + \mathbf{B}$ at the same rank).

Model	Task	Top-2	Module LoRA	Full LoRA	% params
Qwen2.5-1.5B	GSM8K	{V, K}	**64.7**	63.7	8.7
Qwen2.5-3B	GSM8K	{V, K}	**72.0**	71.5	8.9
Qwen2.5-7B	GSM8K	{V, O}	86.2	**87.4**	12.5
Qwen2.5-1.5B	MATH	{V, K}	41.6	**43.5**	8.7
Qwen2.5-3B	MATH	{V, K}	63.4	**63.5**	8.9
Qwen2.5-7B	MATH	{V, O}	73.8	**75.1**	12.5

Table 35 reports GRPO accuracy against full LoRA on all modules. Restricting LoRA to the top two modules matches full LoRA within about a point in every configuration, above it at 1.5B and 3B and slightly below it at 7B. The scores are therefore informative at module granularity as well as element granularity, and the two results bound the trade-off from both sides: at a matched 10K budget only element-level selection works (§F.1.3), while at 10% of the parameter count a module-level restriction already suffices.

F.3 Additional Budget Sweeps

Qwen2.5-1.5B (OpenHermes). Random placement scales slowly: 55.6% at k=1K, 58.3% at 10K, 61.6% at 100K, matching full LoRA only at k=500K, an order of magnitude above the circuit budgets.

Qwen3-8B (Alpaca). $\hat{F}$ leads at every budget tested: 70.9% at k=5K and 71.2% at k=10K, matching full LoRA (71.4%) at 0.02% of adapter parameters. Random (70.5%) and Row Fisher (70.4%) trail by 0.7pp at 10K.

Training dataset sensitivity. The 1.5B results above use OpenHermes. On larger models (3B, 7B), OpenHermes shows mixed effects and yields no net gain. MMLU improves but BoolQ and TruthfulQA degrade. We use Alpaca for 3B and larger in the main paper.

F.4 Text-to-SQL

Mathematical reasoning is where nearly all RLVR results, including those above, are measured. Text-to-SQL differs in two ways. The model must emit a query that parses and executes against a database schema, and the reward is shaped rather than binary, combining output format, successful execution, and a match between the rows returned and those of the reference query. We repeat the placement comparison there, following the setup of SQL-R1 (Ma et al., 2025): we train on their SynSQL data and evaluate execution accuracy on the held-out validation set of SynSQL. We choose the best validation accuracy checkpoint for validating on the dev sets of Spider (Yu et al., 2018) (1,034 queries) and BIRD (Li et al., 2024a) (1,534 queries). We run four Qwen2.5-7B variants (base, instruction-tuned, code-specialized, and both) at k=100K, 0.12% of full LoRA's trainable parameters.

Table 36: **Text-to-SQL under GRPO** (execution accuracy, %). Four Qwen2.5-7B variants trained on SynSQL with an execution reward at k=100K. Best per row in bold.

Model	Pretrained	Random	$\hat{S}$	$\hat{F}$	Full LoRA
Spider-dev					
Qwen2.5-7B	57.8	56.0	**69.2**	66.6	69.0
Qwen2.5-7B-Inst	64.9	66.7	71.7	71.5	**72.1**
Qwen2.5-Coder-7B	63.4	65.4	**70.7**	70.2	70.1
Qwen2.5-Coder-7B-Inst	73.3	74.7	77.9	77.7	**78.0**
BIRD-dev					
Qwen2.5-7B	28.9	30.1	34.9	35.2	**35.5**
Qwen2.5-7B-Inst	33.8	37.0	41.6	41.1	**42.0**
Qwen2.5-Coder-7B	29.1	33.7	**37.9**	37.2	37.7
Qwen2.5-Coder-7B-Inst	45.2	46.5	48.3	**48.9**	48.6

Informed placement beats random on every model and on both benchmarks. The margin is largest on the base model, where random placement on Spider ends below the pretrained model while $\hat{S}$ recovers full LoRA, and it narrows as the model moves closer to the domain. Circuits stay within half a point of full LoRA in every configuration and exceed it in four of the eight.