RIFT: SPARSE REPRESENTATION FINE-TUNING WITH RANDOM SUPPORT

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning aims to adapt pretrained models with few trainable parameters, but achieving strong performance under very small budgets remains challenging. Editing inter-layer representations offers a way to reuse pretrained features and adjust downstream computation while preserving the original weights. However, choosing which parameters to train remains a central design decision under such tight budgets. Here we show that randomly selected channel interactions can provide effective adaptation in this space. We introduce Randomized Intrinsic Fine-Tuning (RIFT), which applies a zero-initialized, entry-sparse linear edit to inter-layer representations. RIFT samples a fixed support uniformly from all channel pairs and trains only the selected coefficients. It requires no task-dependent support calibration, and its support can be reconstructed from a seed under a fixed sampling protocol. RIFT can achieve competitive adaptation across commonsense reasoning, language understanding, and mathematical reasoning while training fewer than one hundred-thousandth as many parameters as the backbone, far below the budgets of the evaluated rank-one adapters. Our ablations characterize RIFT's sensitivity to support selection, parameter budget, and intervention placement, and motivate mechanistic studies of how sparse representation edits alter pretrained model behavior.

1 INTRODUCTION

Parameter-efficient fine-tuning (PEFT) adapts pretrained models by optimizing a small subset of parameters or adding compact trainable modules (Houlsby et al., 2019; Hu et al., 2022). For a fixed set of adapted matrices, low-rank updates provide rank-level budget control, with even rank one requiring parameters proportional to matrix dimensions. Entry-sparse updates permit finer control by training individual coefficients, but require a choice of support. Some methods learn this support or construct it using task-dependent gradients or Fisher information (Guo et al., 2021; Sung et al., 2021; Song et al., 2023). Fixed random structure has also been used in weight-space adaptation (Kopiczko et al., 2024; Lingam et al., 2024), leaving open which parameterizations and budgets make task-dependent selection useful.

Representation Fine-Tuning (ReFT) offers another adaptation interface: learning interventions on hidden states while preserving the pretrained weights (Wu et al., 2024b). It provides explicit control over where and when interventions are applied, creating opportunities to study how representation changes affect model behavior and potentially improve interpretability. Low-rank interventions and coordinatewise edits can achieve strong accuracy–parameter trade-offs (Wu et al., 2024b;a). These properties motivate studying entry-sparse edits within this framework: can they provide finer budget control with support sampled before training, without task-specific calibration? Such a method would remove the need for labeled target-task examples to construct the support and for choosing the size of a separate calibration set.

We introduce **Randomized Intrinsic Fine-Tuning (RIFT)**, which learns a zero-initialized, entry-sparse linear edit of inter-layer representations. RIFT samples a fixed support uniformly from all entries of the edit matrix and trains only the selected coefficients. Each coefficient reads one channel and contributes a state-dependent edit to a selected channel, allowing both same-channel rescaling and cross-channel mixing. Each selected entry contributes one trainable scalar. At our main budgets,

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

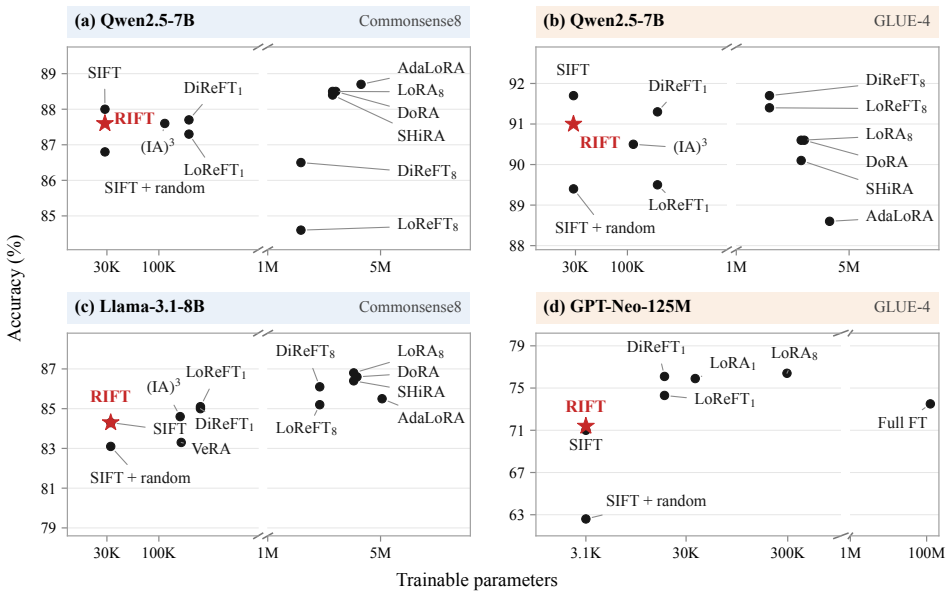

Figure 1: Accuracy versus trainable parameters on Qwen2.5-7B: (a) Commonsense8 and (b) GLUE-4; Llama-3.1-8B: (c) Commonsense8; and GPT-Neo-125M: (d) GLUE-4. RIFT (red stars) approaches SIFT at small budgets on the 7–8B backbones. **Upper-left configurations are preferable, combining higher accuracy with fewer parameters.** Parameter axes use broken logarithmic scales; subscripts denote adapter rank.

the edit is computed sparsely; its support is reconstructible from a seed under a fixed sampling protocol.

Competitive adaptation at small parameter budgets. Our main Qwen2.5-7B configuration trains 28.6K intervention parameters, below the budgets of the evaluated rank-one LoRA and ReFT configurations (Table 1). Figure 1 illustrates the accuracy–parameter trade-off on Commonsense8 and GLUE-4, including Llama-3.1-8B and GPT-Neo-125M. RIFT is competitive at small budgets on the tested larger decoders, but the trade-off is not uniform: encoder results show larger gaps to calibrated alternatives, and RIFT trails larger adapters on Llama GSM8K.

Our contributions are threefold:

1. **Calibration-free sparse representation adaptation.** RIFT provides single-coefficient budget control, unrestricted channel-pair eligibility, and seeded support reconstruction without task-dependent calibration.
2. **Competitive adaptation at exceptionally small budgets.** RIFT achieves competitive performance across three task families on Qwen2.5-7B with far fewer parameters than the evaluated rank-one adapters, compact learned-value storage, and modest peak-memory savings.
3. **Evidence for the effectiveness of random support.** Across the tested backbones, RIFT outperforms random-support SIFT on Commonsense8 and GLUE-4. On the evaluated decoder benchmarks, it also exceeds the saliency-probed RIFT variant under the reported settings. Our ablations characterize sensitivity to parameter budget, intervention placement, and support draws, establishing practical conditions for effective adaptation with random representation edits.

2 Related Work

Parameter-efficient fine-tuning. Adapters insert trainable bottleneck modules between frozen layers (Houlsby et al., 2019), whereas LoRA represents weight updates with low-rank factors (Hu et al., 2022). LoRA variants improve rank allocation, optimization, or initialization (Zhang et al.,

2023; Liu et al., 2024; Meng et al., 2024), but retain rank-level budget granularity whose minimum increment scales with model width. Fixed random structure underlies both VeRA, which learns scaling vectors over shared random low-rank projections (Kopiczko et al., 2024), and RandLoRA, which combines fixed random low-rank bases to produce full-rank updates (Albert et al., 2025). Both demonstrate that learned coefficients can exploit random structure, but they induce dense weight-space updates. RIFT instead learns individual interactions on a fixed random sparse support in representation space, enabling single-entry budget control.

Sparse fine-tuning and support selection. Sparse updates provide entry-level budget control but must determine which entries to train. Existing methods learn the support jointly (Guo et al., 2021), select weights using Fisher or gradient saliency (Sung et al., 2021; Song et al., 2023), impose fixed structural patterns (Ansell et al., 2022; Bhardwaj et al., 2024), or learn sparse coefficients in an alternative basis, as in SVFT (Lingam et al., 2024). Evidence for task-dependent selection is mixed: SIFT finds gradient-selected weight coordinates superior to random support (Song et al., 2023), whereas SVFT finds random support stronger than saliency-based selection in its singular-vector basis, with a banded pattern performing best (Lingam et al., 2024). These findings suggest that selection utility depends on the sparse parameterization. RIFT instead places entry sparsity on interactions between inter-layer representation coordinates and samples the support uniformly, eliminating the task-dependent selection stage.

Representation fine-tuning. Representation Fine-tuning (ReFT) adapts hidden states rather than model weights, often using substantially fewer parameters than LoRA (Wu et al., 2024b). LoReFT learns an orthogonal low-rank subspace, whereas DiReFT removes the orthogonality constraint; both therefore provide rank-level budget granularity Wu et al. (2024b). Coordinatewise methods instead learn per-dimension edits: RED applies scales and biases to hidden states (Wu et al., 2024a), while $(\mathrm{IA})^3$ rescales attention and feed-forward activations (Liu et al., 2022). Their diagonal form cannot mix coordinates, and their parameter cost scales with representation width. More recently, JoLA jointly learns which attention heads to edit, whether to use additive or multiplicative interventions, and the edit parameters themselves (Lai et al., 2025). RIFT instead fixes a random, entry-sparse support over all channel pairs before training, enabling coordinate mixing with single-entry budget control and no learned localization.

Inter-layer representations. Transformer blocks communicate through model-width inter-layer states. Their features can occupy non-axis-aligned, superposed directions (Elhage et al., 2021; 2022; Bricken et al., 2023; Templeton et al., 2024), but architectural effects and massive activations can privilege individual coordinates (Sun et al., 2024). Distributed representations motivate testing whether randomly sampled channel interactions can support effective adaptation, even though differences across channels may favor some supports over others.

3 METHOD

3.1 RANDOM SPARSE REPRESENTATION EDITS

We use row vectors throughout. Let $\boldsymbol{h}_{\ell,t} \in \mathbb{R}^{1\times d}$ be the state passed from Transformer block ℓ to the next block at position t. For selected layers $\ell \in \mathbb{L}$ and input-dependent token positions $t \in \mathbb{T}(x)$, RIFT applies

$$\widetilde{\boldsymbol{h}}_{\ell,t} = \boldsymbol{h}_{\ell,t}(\boldsymbol{I} + \boldsymbol{C}_\ell) = \boldsymbol{h}_{\ell,t} + \boldsymbol{h}_{\ell,t}\boldsymbol{C}_\ell, \tag{1}$$

where $\boldsymbol{C}_\ell \in \mathbb{R}^{d\times d}$ is an entry-sparse edit (Figure 2). The same operator is shared across the selected positions at a layer. In our pre-normalized decoders, this inter-layer state is the residual stream. An entry $C_{\ell,rc}$ reads channel r and writes to channel c:

$$[\boldsymbol{h}_{\ell,t}\boldsymbol{C}_\ell]_c = \sum_{r=1}^{d} h_{\ell,t,r} C_{\ell,rc}. \tag{2}$$

Thus each coefficient produces an input-dependent correction.

For each selected layer, we sample a support $\mathbb{S}_\ell \subseteq \{1,\dots,d\}^2$ of size k_ℓ uniformly without replacement from all d^2 entries, including the diagonal, with no per-row or per-column allocation

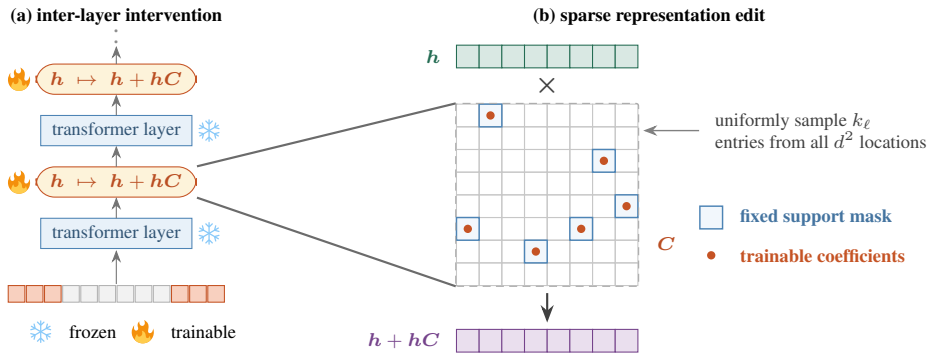

Figure 2: RIFT learns sparse representation edits on fixed random support. **(a)** Edits intervene between frozen Transformer blocks at selected token positions (orange). **(b)** Each edit maps $\boldsymbol{h}$ to $\boldsymbol{h} + \boldsymbol{hC}$, training only coefficients on a support sampled uniformly from all d^2 channel pairs before training, without task-dependent calibration.

constraints. The support remains fixed during training. We initialize its coefficients to zero and train only these coefficients, keeping all pretrained parameters frozen; the intervention therefore initially preserves the backbone's forward computation.

Density controls the realized budget. We specify a support density $\rho_\ell \in (0, 1]$, the requested fraction of eligible entries retained. The implementation truncates the computed product $\rho_\ell d^2$ to an integer, with a minimum of one entry, to obtain k_ℓ. The exact number of trainable intervention parameters is

$$P_{\mathrm{RIFT}} = \sum_{\ell \in \mathbb{L}} |\mathbb{S}_\ell| = \sum_{\ell \in \mathbb{L}} k_\ell. \tag{3}$$

We report realized counts, with one scalar per selected entry. These counts refer to the intervention; any separately trained task head is additional. Notation is collected in Appendix A.1.

Comparison with sparse weight updates. For an adapted pretrained weight matrix $\boldsymbol{W}_{\mathrm{pt}} \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{out}}}$, SIFT learns an entry-sparse increment (Song et al., 2023):

$$\boldsymbol{W}_{\mathrm{ft}} = \boldsymbol{W}_{\mathrm{pt}} + \Delta \boldsymbol{W}, \qquad \Delta W_{rc} = 0 \quad \text{for } (r, c) \notin \mathbb{S}^{\boldsymbol{W}}_{\mathrm{SIFT}}. \tag{4}$$

Given a budget of m entries for this matrix, SIFT fixes the support using the largest absolute entries of an initial task gradient:

$$\boldsymbol{G} = \nabla_{\boldsymbol{W}} \mathcal{L}_{\mathrm{cal}}(\boldsymbol{W}_{\mathrm{pt}}), \qquad \mathbb{S}^{\boldsymbol{W}}_{\mathrm{SIFT}} = \mathrm{Top}_m(\{|G_{rc}|\}_{r,c}), \tag{5}$$

where $\mathcal{L}_{\mathrm{cal}}$ is the loss on labeled calibration examples and Top_m returns coordinate pairs. The support remains fixed while its selected increments are trained. Our random-support SIFT baseline retains this weight-space parameterization but samples the support uniformly instead.

Both methods provide entry-level budget control. RIFT instead places its coefficients in $\boldsymbol{C}_\ell$ and edits the state passed between blocks; random selection removes the need for labeled calibration examples and a calibration-set-size hyperparameter.

3.2 Sparse computation and support reconstruction

Enumerate the support as $\mathbb{S}_\ell = \{(r_{\ell,j}, c_{\ell,j})\}_{j=1}^{k_\ell}$ and store the corresponding values in $\boldsymbol{\theta}_\ell \in \mathbb{R}^{k_\ell}$. Using $\boldsymbol{e}_i$ for a standard basis column vector, the edit is

$$\boldsymbol{C}_\ell = \sum_{j=1}^{k_\ell} \theta_{\ell,j} \boldsymbol{e}_{r_{\ell,j}} \boldsymbol{e}_{c_{\ell,j}}^\top. \tag{6}$$

Its application can be evaluated directly as

$$\boldsymbol{h}_{\ell,t} \boldsymbol{C}_\ell = \sum_{j=1}^{k_\ell} h_{\ell,t,r_{\ell,j}} \theta_{\ell,j} \boldsymbol{e}_{c_{\ell,j}}^\top. \tag{7}$$

At the main reported budgets, the implementation gathers read coordinates, multiplies by their coefficients, and scatter-adds into write coordinates, without a dense edit matrix. For $k_\ell > d$, larger-budget ablations use a dense matrix product for the same linear map; floating-point accumulation order may differ.

Coefficients and active indices require $\mathcal{O}(k_\ell)$ storage, and the sparse correction uses $\mathcal{O}(k_\ell)$ multiply–accumulate operations per intervened state. These counts exclude width-d state buffers, initialization or copying, and frozen-backbone computation; they are not end-to-end runtime bounds.

Support reconstruction requires the same seed, sampler, layer ordering, dimensions, support sizes, and coefficient ordering. The reference checkpoint format nevertheless stores the realized indices alongside the coefficients to preserve the exact support across software environments. Reported values-only storage excludes those indices and checkpoint metadata; seed-only storage is a possible alternative, not the distributed format.

4 Experimental Set-Up

Models and evaluation metrics. Our main evaluation uses Qwen2.5-7B (Qwen Team, 2025). We evaluate commonsense reasoning on BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy and ARC-Challenge (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018), following the Commonsense8 protocol of Hu et al. (2023). We additionally evaluate natural-language understanding on MNLI, QNLI, RTE, MRPC, and CoLA from GLUE (Wang et al., 2019), and mathematical reasoning on GSM8K (Cobbe et al., 2021).

We evaluate additional backbones to characterize the scope of random-support adaptation: Llama-3.1-8B, Qwen2.5-0.5B and Qwen2.5-1.5B (Qwen Team, 2025), GPT-Neo-125M, and RoBERTa-base and RoBERTa-large (Liu et al., 2019).

Commonsense8 reports mean accuracy over eight tasks; GLUE-4 averages MNLI, QNLI, RTE, and MRPC, with CoLA reported separately. GSM8K uses numeric exact match and counts unparseable outputs as errors. Learning rates and structural hyperparameters are reported in Appendix A.2.

Compute environment. Training used a pooled GPU cluster. Hardware-dependent efficiency measurements used dedicated reruns on the same NVIDIA H100 80 GB HBM3 GPU, with matched software, precision, batch size, and sequence length.

Baselines. We compare RIFT with weight- and activation-space PEFT baselines, including LoRA (Hu et al., 2022), AdaLoRA (Zhang et al., 2023), SHiRA (Bhardwaj et al., 2024), $(\mathrm{IA})^3$ (Liu et al., 2022), and VeRA (Kopiczko et al., 2024). Representation-space baselines include LoReFT and DiReFT (Wu et al., 2024b). SIFT (Song et al., 2023) is the closest sparse weight-space baseline: it uses target-task gradients to select a fixed set of trainable weight coordinates, whereas RIFT uniformly samples fixed routes between inter-layer representation coordinates.

RIFT configuration and parameter budgets. We intervene at the output of every Transformer block using the intervention framework of Wu et al. (2024a), with support sampled from all channel pairs as in Section 3.1. Unless stated otherwise, interventions apply to the first and last seven non-padding input tokens.

We control the RIFT parameter budget through the support density at each intervened layer and report the realized number of trainable coefficients. Our primary Qwen2.5-7B configuration trains $28{,}644$ coefficients (28.6K) across all 28 Transformer blocks.[1] The RoBERTa-base and RoBERTa-large configurations train $3{,}072$ and $8{,}208$ coefficients, respectively. Budget ablation varies the support density while holding intervention layers, token positions, and optimization settings fixed. Training configuration details are in Appendix A.2.

[1] The configured density is $1024/3584^2$. Floating-point evaluation followed by integer truncation produces 1,023 selected entries per block. SIFT uses 28,672 coefficients, so the two budgets differ by approximately 0.1%.

Table 1: RIFT achieves competitive performance across commonsense reasoning, language understanding, and mathematical reasoning on Qwen2.5-7B, trailing DiReFT (r=8) by only 0.7 percentage points on GLUE with less than 2% of its trainable parameters.

	Qwen2.5-7B					Llama-3.1-8B			
				GSM8K				GSM8K	
Method	Params	CS8	GLUE-4	Acc.	Unp.	Params	CS8	Acc.	Unp.
Frozen (zero-shot)	0	75.0	77.5	59.9	27.5	0	2.5	3.1	85.0
Full fine-tuning	7.62B	88.7	—	76.9	0.4	8.03B	85.0	65.4	0.1
Rank-limited budgets									
AdaLoRA	3.79M	**88.7**	88.6	**78.0**	0.3	5.11M	85.5	63.0	0.3
DoRA (r=8)	2.64M	88.5	90.6	76.3	0.5	3.57M	86.6	63.0	0.3
LoRA (r=8)	2.52M	88.5	90.6	76.9	0.5	3.41M	**86.8**	**63.2**	0.2
SHiRA	2.52M	88.4	90.1	76.1	0.5	3.41M	86.4	63.0	0.1
LoReFT (r=8)	1.61M	84.6	91.4	71.3†	1.4	2.10M	85.2	57.9	0.8
DiReFT (r=8)	1.61M	86.5	**91.7**	58.4	1.9	2.10M	86.1	57.7	0.6
LoRA (r=1)	315.4K	88.2	91.2	77.1	0.3	426K	86.1	58.4	0.6
LoReFT (r=1)	**200.7K**	87.3	89.5	77.0	0.2	**262.2K**	85.1	54.5	0.7
DiReFT (r=1)	**200.7K**	87.7	91.3	76.9	0.6	**262.2K**	85.0	55.5	1.1
Sub-rank-1 budgets									
VeRA	118.3K	85.0	89.1	77.5	0.2	167.9K	83.3	**57.6**	0.9
$(IA)^3$	114.7K	87.6	90.5	**77.8**	0.3	163.8K	**84.6**	55.5	0.8
SIFT	**28.7K**	**88.0**	**91.7**	77.6	0.2	**32.8K**	84.3	49.9	1.1
Sub-rank-1 budgets, calibration-free									
SIFT (random)	28.7K	86.8	89.4	**77.8**	0.3	**32.8K**	83.1	47.8	1.7
RIFT (ours)	**28.6K**	**87.6**	**91.0**	77.0	0.2	**32.8K**	**84.3**	**49.0**	1.0

CS8 and GLUE-4 are unweighted task means; GSM8K reports accuracy and unparseable-output rate (Unp.), all in percent.

5 RESULTS

5.1 ACCURACY AT SMALL ADAPTER BUDGETS

At the evaluated sparse budget on Qwen2.5-7B, RIFT remains within 0.6 percentage points of rank-one LoRA across Commonsense8, GLUE-4, and GSM8K while using approximately one-eleventh as many adapter parameters (Table 1). On Commonsense8, RIFT reaches 87.6% on Qwen2.5-7B and 84.3% on Llama-3.1-8B, with higher mean accuracy than random-support SIFT in the evaluated runs at approximately matched budgets. Its performance is also close to gradient-selected SIFT on these two backbones.

On Llama GSM8K, however, RIFT reaches 49.0%, versus 58.4% for rank-one LoRA and 63.2% for rank-eight LoRA at larger budgets. Detailed Qwen and Llama results appear in Appendices A.5 and A.4.

5.2 VARIATION ACROSS BACKBONES

At approximately matched sparse budgets, RIFT exceeds random-support SIFT on all four backbones (Table 2). Its gap to gradient-selected SIFT on Qwen2.5 is 6.4 points at 0.5B, 0.8 at 1.5B, and 0.4 at 7B; on Llama-3.1-8B, the gap is 0.1 point. RIFT thus approaches the calibrated baseline on the two largest decoder models tested, while the 0.5B result remains a limitation.

On the smaller backbones, RIFT shows larger gaps to the strongest baselines at the reported budgets (Appendix A.6). On RoBERTa-base and RoBERTa-large, RIFT scores below calibrated SIFT on the reported GLUE tasks (Appendix A.8); on GPT-Neo-125M, it trails larger adapters (Appendix A.7). Model size, architecture, pretraining, and evaluation differ across these comparisons, so they do not isolate a causal effect of scale.

Table 2: At approximately matched parameter budgets, RIFT outperforms random-support SIFT across all four backbones and approaches gradient-selected SIFT on Qwen2.5-7B and Llama-3.1-8B. Within Qwen2.5, the gap to gradient-selected SIFT narrows substantially as model size increases.

	Qwen2.5	smaller ⟶ larger		Llama-3.1
Model size	0.5B	1.5B	7B	8B
Nominal budget	6.1K	14.3K	28.6K	32.8K
SIFT, gradient support	63.2	78.9	88.0	84.3
SIFT, random support	55.8	73.6	86.8	83.1
RIFT, random support (ours)	56.8	78.1	87.6	84.3

Table 3: RIFT achieves the smallest learned-value storage footprint and lowest peak reserved training memory among the evaluated methods on Qwen2.5-7B.

Method	Params	Storage (MB)	Peak VRAM (GB)
AdaLoRA	3.79M	7.27	23.5
SHiRA	2.52M	4.83	23.6
DiReFT (r=8)	1.61M	3.09	24.0
LoReFT (r=8)	1.61M	3.09	24.0
LoRA (r=1)	315.4K	0.64	23.5
DiReFT (r=1)	200.7K	0.41	23.9
LoReFT (r=1)	200.7K	0.41	24.0
VeRA	118.3K	0.26	23.1
$(IA)^3$	114.7K	0.24	22.8
SIFT	28.7K	0.07	23.6
RIFT (ours)	28.6K	**0.06**	**22.6**

Storage is the serialized learned-value payload, excluding support indices and other checkpoint metadata; it is not the complete adapter size. VRAM excludes the CUDA context. All methods use the same measurement settings; details are in Table 20.

5.3 Adapter Storage and Training Memory

Table 3 reports learned-value payload storage and peak reserved VRAM on Qwen2.5-7B at bfloat16 precision, batch size 4, and sequence length 512 on an H100 80 GB GPU. RIFT's value payload is 0.06 MB at 28.6K parameters, close to SIFT's 0.07 MB. These are not complete adapter checkpoint sizes: stored support indices and other checkpoint metadata are excluded. Peak reserved VRAM spans 22.6–24.0 GB, indicating modest training-memory savings under these conditions.

RIFT also avoids task-dependent support calibration. SIFT selects 28,672 coefficients from 411M fp32 gradient entries (1.53 GB). RIFT needs no labeled-data probe; support reconstruction is possible under a fixed sampling protocol (Appendix A.10).

6 Ablation Studies

6.1 Task-dependent support selection

We compare random RIFT with support selected by activation–gradient saliency $|\boldsymbol{H}^\top \boldsymbol{A}|$, alongside SIFT and SIFT + random in weight space. Budgets are approximately matched within each backbone. Appendix A.3 gives the formulation and protocol.

Random RIFT exceeds the saliency-probed variant on both Commonsense8 backbones at the reported settings (Table 4). On Qwen GLUE-4, it reaches 91.0 versus 87.4 with the gain concentrated in RTE. SIFT instead exceeds its random counterpart on all three comparisons. These results demonstrate effective calibration-free representation adaptation.

6.2 Intervention placement

On Qwen2.5-7B, reducing intervention positions from first-and-last-seven to first-and-last-one lowers Commonsense8 accuracy from 87.6% to 86.3%; using only the final token yields 83.4% (Ta-

Table 4: Random-support RIFT outperforms its saliency-probed variant across the reported decoder benchmarks.

	Qwen2.5-7B			Llama-3.1-8B	
Method	Params	Commonsense8	GLUE-4	Params	Commonsense8
SIFT	28.7K	**88.0**	**91.7**	32.8K	**84.3**
SIFT + random	28.7K	86.8	89.4	32.8K	83.1
RIFT + probe	28.7K	85.4	87.4	32.8K	76.9
RIFT (random)	28.6K	**87.6**	**91.0**	32.8K	**84.3**

Table 5: RIFT benefits from intervening at more token positions, while the best-performing layer band is early on Qwen2.5-7B and early-middle on RoBERTa-base.

		Qwen2.5-7B / Commonsense8			RoBERTa-base / GLUE		
Ablation	Setting	$\|L\|$	# pos.	Acc.	$\|L\|$	# pos.	GLUE-4
Default	All layers, f7+l7	28	14	**87.6**	12	14	<u>74.8</u>
Positions	f3+l3	28	6	87.0	12	6	74.6
	f1+l1	28	2	86.3	12	2	73.7
	Last token / `[CLS]`	28	1	83.4	12	1	73.4
Layers	Early	7	14	**87.6**	3	14	73.0
	Early-mid	7	14	87.0	3	14	**76.0**
	Late-mid	7	14	84.4	3	14	73.5
	Late	7	14	80.3	3	14	70.5

ble 5). Concentrating the approximately matched budget in the first seven blocks preserves the displayed all-layer score, whereas the last seven yield 80.3%. The best tested band differs on RoBERTa-base, where early-middle layers score 76.0% versus 74.8% for all layers. Placement effects therefore depend on the backbone and task suite.

6.3 Parameter budget

Figure 3 examines budget sensitivity using a mean over BoolQ, HellaSwag, and WinoGrande on Qwen-2.5-7B and Llama-3.1-8B. These scores are not directly comparable to the eight-task Commonsense8 leaderboard. For SIFT, the advantage of gradient-selected over random support decreases from 9.9 to 0.8 points between 0.9K and 229K parameters on Qwen, and from 9.7 points at 1K to approximately zero at 131K on Llama. This suggests that selecting individual weight coordinates matters most under severe budget constraints; increasing the budget reduces the observed penalty for random selection.

RIFT's behavior at the smallest budgets differs across backbones. On Qwen, it remains close to calibrated SIFT at 0.9K parameters, whereas on Llama it trails by 8.4 points at 1K and approaches SIFT as the budget grows. Random representation-space support therefore does not eliminate budget sensitivity: the budget needed for competitive performance depends on the backbone under the tested settings.

6.4 Sensitivity to support draws

We evaluate whether RIFT's performance depends on a favorable random support using ten independently sampled supports per task on Qwen2.5-7B and Llama-3.1-8B with Commonsense8, and RoBERTa-base with GLUE-4. We fix the training seed at 42 and retain each configuration's hyperparameters.

Across support draws, benchmark-level standard deviations are 0.15 and 0.24 percentage points on Qwen and Llama Commonsense8, respectively, and 0.44 points on RoBERTa GLUE-4 (Table 6). Task-level standard deviations range from 0.09 to 1.16 points. Thus the reported performance is not confined to one favorable support draw.

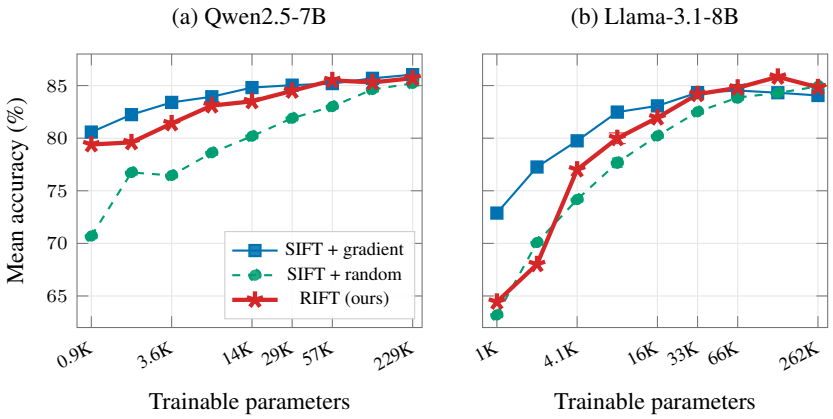

Figure 3: RIFT rapidly closes the gap to gradient-selected SIFT as the parameter budget increases, reaching comparable performance on both backbones without task-dependent support calibration.

Table 6: RIFT shows little variation in benchmark accuracy across random support draws when the training seed is held fixed.

Backbone	Benchmark	Mean (%)	SD (pp)
Qwen2.5-7B	Commonsense8	87.63	0.15
Llama-3.1-8B	Commonsense8	84.20	0.24
RoBERTa-base	GLUE-4	74.78	0.44

7 Discussion

Random support as an adaptation choice. RIFT demonstrates that randomly selected representation edits can achieve competitive adaptation with far fewer trainable parameters than the evaluated rank-one adapters, without task-dependent support calibration. Pretrained representations may offer many channel interactions through which sparse edits can adjust downstream computation. In smaller models, useful interactions may be fewer or more concentrated, making random selection less effective and task-dependent saliency more valuable. This remains a hypothesis: we have not directly measured the distribution of useful interactions, and differences in optimization may also contribute to the observed gaps.

Limitations and future directions. RIFT trails calibrated SIFT on the tested encoders and larger adapters on Llama mathematical reasoning. These comparisons do not isolate model scale from architecture, pretraining, and task differences. Our evaluation also leaves larger backbones and other modalities unexplored. Future work could extend RIFT to larger language models and vision transformers, and investigate the mechanisms through which random representation edits enable adaptation, including how selected channel interactions affect downstream computation and evolve during training. Evaluating distribution shifts and end-to-end runtime would further clarify its practical applicability.

AI USE STATEMENT

In this work, we used generative AI tools to assist with literature searches, brainstorming diagnostic analyses, checking mathematical formulations, code development and debugging, tracking experimental sweeps, interpreting experimental summaries, and drafting and editing manuscript text. We reviewed the AI-assisted work and checked mathematical expressions and numerical summaries against the underlying analyses and experimental records. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

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, 2025. URL https://arxiv.org/abs/2502.00987.

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 404–413. PMLR, 2018. URL https://proceedings.mlr.press/v80/balles18a.html.

Kartikeya Bhardwaj, Nilesh P Pandey, Sweta Priyadarshi, Viswanath Ganapathy, Shreya Kadambi, Rafael Esteves, Shubhankar Borse, Paul Whatmough, Risheek Garrepalli, Mart Van Baalen, et al. Sparse high rank adapters. *Advances in Neural Information Processing Systems*, 37:13685–13715, 2024.

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

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E. Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *North American Chapter of the Association for Computational Linguistics (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 preprint*, 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, 2021. URL https://arxiv.org/abs/2110.14168.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. LLM-Adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

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

Wen Lai, Alexander Fraser, and Ivan Titov. Joint localization and activation editing for low-resource fine-tuning, 2025. URL https://arxiv.org/abs/2502.01179.

Vijay Lingam, Atula Tejaswi, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Alex Dimakis, Eunsol Choi, Aleksandar Bojchevski, and Sujay Sanghavi. Svft: Parameter-efficient fine-tuning with singular vectors, 2024. URL https://arxiv.org/abs/2405.19597.

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

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

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.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Qwen Team. Qwen2.5 technical report. *arXiv preprint*, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Weixi Song, Zuchao Li, Lefei Zhang, Hai Zhao, and Bo Du. Sparse is enough in fine-tuning pre-trained large language models. *arXiv preprint arXiv:2312.11875*, 2023.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. In *Conference on Language Modeling (COLM)*, 2024.

Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jermyn, Shan Carter, Christopher Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019. URL https://arxiv.org/abs/1804.07461.

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Zhu JianHao, Cenyuan Zhang, Xiaoqing Zheng, and Xuan-Jing Huang. Advancing parameter efficiency in fine-tuning via representation editing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13445–13464, 2024a.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation finetuning for language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

A APPENDIX

A.1 NOTATION

RIFT is specified by three objects: the inter-layer states on which it acts, a fixed random support defining the available coordinate routes, and the coefficients learned on those routes.

Model and intervention. We use row vectors throughout.

Symbol	Meaning
$\boldsymbol{h}_{\ell,t} \in \mathbb{R}^{1\times d}$	inter-layer hidden state at layer ℓ and token position t
d	width of the inter-layer state
$\mathbb{L}$	set of intervened layers
$\mathbb{T}(x)$	token positions selected for input x; by default, the first and last seven non-padding input tokens
$\boldsymbol{I}_d$	$d \times d$ identity matrix
$\mathbf{e}_r$	r-th standard basis vector in $\mathbb{R}^d$

For each $\ell \in \mathbb{L}$, RIFT applies

$$\Phi_\ell(\boldsymbol{h}_{\ell,t}) = \boldsymbol{h}_{\ell,t}(\boldsymbol{I}_d + \boldsymbol{C}_\ell) = \boldsymbol{h}_{\ell,t} + \boldsymbol{h}_{\ell,t}\boldsymbol{C}_\ell, \qquad t \in \mathbb{T}(x). \tag{8}$$

Here $\boldsymbol{h}_{\ell,t}$ is the state passed from one Transformer block to the next; in our pre-normalized decoder models, it is the residual stream.

Sparse routes and trainable coefficients.

Symbol	Meaning
$\mathcal{I} = \{(r,c) : 1 \le r, c \le d\}$	candidate set of all read–write coordinate pairs
$N = d^2$	number of candidate routes per layer
$\mathbb{S}_\ell \subset \mathcal{I}$	fixed support sampled for layer ℓ
$k_\ell = \lvert\mathbb{S}_\ell\rvert$	number of trainable routes at layer ℓ
$\boldsymbol{\Omega}_\ell \in \{0,1\}^{d\times d}$	conceptual binary mask encoding $\mathbb{S}_\ell$, with $\lVert\boldsymbol{\Omega}_\ell\rVert_0 = k_\ell$
$\boldsymbol{\theta}_\ell \in \mathbb{R}^{k_\ell}$	trainable coefficients associated with the routes in $\mathbb{S}_\ell$
$\boldsymbol{C}_\ell \in \mathbb{R}^{d\times d}$	sparse coordinate-mixing operator determined by $(\mathbb{S}_\ell, \boldsymbol{\theta}_\ell)$
r	read coordinate of a route (r, c)
c	write coordinate of a route (r, c)
$P_{\mathrm{RIFT}} = \sum_{\ell\in\mathbb{L}} k_\ell$	total number of trainable RIFT parameters

If the j-th route in $\mathbb{S}_\ell$ is $(r_{\ell,j}, c_{\ell,j})$, then

$$\boldsymbol{C}_\ell = \sum_{j=1}^{k_\ell} \theta_{\ell,j}\, \mathbf{e}_{r_{\ell,j}} \mathbf{e}_{c_{\ell,j}}^\top, \qquad \boldsymbol{h}\boldsymbol{C}_\ell = \sum_{j=1}^{k_\ell} \theta_{\ell,j} h_{r_{\ell,j}}\, \mathbf{e}_{c_{\ell,j}}^\top. \tag{9}$$

Thus, route (r, c) reads coordinate h_r and writes the input-dependent contribution $\theta_{\ell,r,c}h_r$ to coordinate c. The coefficients are initialized to zero, so RIFT initially preserves the frozen model exactly.

Each $\mathbb{S}_\ell$ is sampled uniformly without replacement from $\mathcal{I}$, with the supports determined by the run seed under the fixed sampling protocol. The support remains fixed throughout training; only $\boldsymbol{\theta}_\ell$ is optimized. Conceptually, support construction determines $\boldsymbol{\Omega}_\ell$, whereas downstream training learns $\boldsymbol{\theta}_\ell$.

At the main budgets, the active read coordinates are gathered, multiplied by their coefficients, and scattered to the write coordinates without a dense edit matrix. For $k_\ell > d$, the reference implementation instead uses the dense branch described in Section 3.2. When every layer uses budget k, $P_{\mathrm{RIFT}} = \lvert\mathbb{L}\rvert k$.

A.2 Training Configuration

Table 7: Shared optimization settings differ across backbone and benchmark families; GPT-Neo uses generation-based evaluation.

	Qwen2.5-7B / Llama-3.1-8B		GPT-Neo-125M	RoBERTa-base/large
Setting	CS8 / GSM8K	GLUE	GLUE	GLUE
Model type	Decoder	Decoder	Decoder	Encoder
Optimizer	AdamW	AdamW	AdamW	AdamW
Schedule	Linear decay	Linear decay	Linear decay	Linear decay
Warmup ratio	0.06	0.06	0.06	0.06
Weight decay	0.0	0.0	0.0	0.0
Batch size	4	4	32	32
Epochs	3	5	10	10
Max sequence length	512	512	512	512
Precision	bfloat16	bfloat16	float32	float32
Default seeds	42–44	42–44	42–44	42–44
Evaluation	Generate and match	Generate and match	Generate and match	Classification head
Representation positions	f7+l7	f7+l7	f7+l7	f7+l7

These are the default adaptation settings; ablations and full fine-tuning exceptions are specified separately. Qwen full fine-tuning on GSM8K uses two epochs. RoBERTa RTE and CoLA include additional seeds 45–47 where available. GPT-Neo uses its causal language-model head with generation and answer matching, with no separately trained classification head. RoBERTa uses classification heads whose parameters are additional to the adapter counts. The position row applies to representation interventions; GPT-Neo CoLA reruns use the `cls` position setting.

Table 8: Learning rate per method, backbone and benchmark: the single hyperparameter tuned per method. Each value is the rate maximising that method's mean over the benchmark's task set, restricted to cells with at least three seeds. "—" marks an unreported combination. †marks a rate at the edge of its searched grid, i.e. an optimum the search may not have bracketed. *marks a cell with a single seed.

	Qwen2.5-7B			Llama-3.1-8B		Encoders	
Method	CS8	GLUE	GSM8K	CS8	GSM8K	RoBERTa-b	RoBERTa-l
Full fine-tuning	1e-5*	—	1e-5	1e-5*	1e-5*	1e-5	1e-5
AdaLoRA	1e-3	3e-5	3e-4	3e-4	1e-3	1e-3	1e-3
DoRA (r=8)	3e-4	1e-5	3e-4	1e-4	3e-4	3e-4	3e-4
LoRA (r=8)	3e-4	1e-5	1e-4	1e-4	3e-4	3e-4	3e-4
SHiRA (r=8)	3e-4	3e-5	3e-4	3e-4	1e-3	1e-3	1e-3
LoRA (r=1)	1e-4	1e-4	3e-4	1e-4	3e-4	1e-3	3e-4
LoReFT (r=8)	3e-3	1e-4	3e-3	3e-4	1e-3	—	—
DiReFT (r=8)	3e-3	3e-4	3e-3	1e-3	1e-3	—	—
LoReFT (r=1)	3e-3	3e-3	9e-4	3e-4	3e-3	1e-3	3e-4
DiReFT (r=1)	3e-3	3e-3	3e-3	3e-4	1e-3	3e-4	3e-4
VeRA (r=64)	3e-4	3e-4	3e-4	1e-3	3e-3	1e-2	3e-3
$(IA)^3$	3e-3	3e-3	3e-4	3e-3	3e-3	3e-3	3e-3
SIFT	3e-4	3e-4	3e-4	3e-4	1e-3	1e-3	1e-3
SIFT + random	1e-2	1e-3†	1e-4†	1e-2	1e-3	3e-3†	3e-3†
RIFT (ours)	3e-2	1e-2	3e-2	3e-2	1e-1	1e-2	1e-2

Table 9: Method-specific hyperparameters and trainable parameter counts. Reparameterisation baselines target the query and value projections following Hu et al. (2022); counts therefore exclude the key and output projections and the MLP. RoBERTa counts exclude the classification head, which is trained for every method including head-only and is 0.59M on RoBERTa-base and 1.05M on RoBERTa-large. GPT-Neo has no separately trained classification head.

Method	Hyperparameters	Qwen2.5-7B	RoBERTa-b	RoBERTa-l
Full fine-tuning	—	7.62B	124.1M	354.3M
AdaLoRA	r_{init}=12, r_{target}=8	3,785,376	442,656	1,180,224
DoRA (r=8)	α=16, dropout 0	2,637,824	313,344	835,584
LoRA (r=8)	α=16, dropout 0	2,523,136	294,912	786,432
SHiRA (r=8)	—	2,523,136	294,912	786,432
LoRA (r=1)	α=2, dropout 0	315,392	36,864	98,304
LoReFT (r=8)	rank 8, all layers	1,605,856	—	—
DiReFT (r=8)	rank 8, all layers	1,605,856	—	—
LoReFT (r=1)	rank 1, all layers	200,732	18,444	49,176
DiReFT (r=1)	rank 1, all layers	200,732	18,444	49,176
VeRA	r=64, shared projections	118,272	19,968	52,224
$(\text{IA})^3$	—	114,688	18,432	49,152
SIFT	top-k $\lvert\partial L/\partial W\rvert$, 512/module (Qwen)	28,672	3,072	8,208
SIFT + random	uniform, 512/module (Qwen)	28,672	3,072	8,208
RIFT (ours)	k per layer, `block_output`	28,644	3,072	8,208

Table 10: Support construction for the sparse methods. Among the three configurations below, only SIFT requires a calibration pass; both random-support arms draw their support before seeing any data.

	SIFT + gradient	SIFT + random	RIFT (ours)
Space	Weight	Weight	Representation
Selection	top-k $\lvert\partial L/\partial W\rvert$	Uniform over W	Uniform over $d \times d$
Target	`q_proj`, `v_proj`	`q_proj`, `v_proj`	`block_output`
Labelled data required	Yes	No	No
Calibration examples	32	—	—
Reference checkpoint	Stored indices	Stored indices	Stored indices
Seed reconstruction	No	Possible	Possible
Diagonal excluded	No	No	No

A.3 SALIENCY-PROBED REPRESENTATION SUPPORT

The RIFT + probe ablation retains RIFT's sparse linear intervention but constructs its support from labeled training calibration data. For calibration example x_i with task loss $\mathcal{L}_i$, collect the inter-layer states $\boldsymbol{h}_{\ell,t}(x_i)$ and their loss gradients at the selected positions $t \in \mathbb{T}(x_i)$. Stack these as rows of $\boldsymbol{H}_\ell, \boldsymbol{A}_\ell \in \mathbb{R}^{n\times d}$, respectively, where $n = \sum_i |\mathbb{T}(x_i)|$ counts example–position pairs. With $\mathcal{L}_{\text{cal}} = \sum_i \mathcal{L}_i$ and zero edit coefficients, the initial edit gradient is

$$\boldsymbol{P}_\ell = \left.\frac{\partial \mathcal{L}_{\text{cal}}}{\partial \boldsymbol{C}_\ell}\right|_{\boldsymbol{C}=\boldsymbol{0}} = \boldsymbol{H}_\ell^\top \boldsymbol{A}_\ell. \tag{10}$$

The implementation concatenates the collected rows without channel or per-position normalization. Signed contributions are summed before taking absolute values.

The reported probe suppresses diagonal scores and selects the k_ℓ largest scores:

$$Q_{\ell,rc} = \begin{cases} |P_{\ell,rc}|, & r \neq c, \\ 0, & r = c, \end{cases} \qquad \mathbb{S}_\ell^{\text{probe}} = \text{Top}_{k_\ell}(\boldsymbol{Q}_\ell), \tag{11}$$

where Top_{k_ℓ} returns coordinate pairs. Zeroing diagonal scores does not strictly exclude them in zero-valued ties. Random-support RIFT instead samples uniformly from all d^2 channel pairs. The selected support remains fixed, and its trainable coefficients are initialized to zero.

Decoder protocol. Table 4 uses ordinary hidden-state saliency probes calibrated on 32 labeled training examples per task, processed one example at a time in evaluation mode without parameter updates. These are not composable multi-task probes. Both RIFT variants edit every block output at the first and last seven prompt tokens. Qwen uses $d = 3584$ and 28 layers: the probe trains 1,024 coefficients per layer (28,672 total), while random RIFT realizes 1,023 per layer (28,644 total) through integer truncation. Llama uses $d = 4096$ and 32 layers, with 1,024 coefficients per layer (32,768 total) in both variants. All pretrained parameters remain frozen.

Training uses bfloat16, AdamW, batch size 4, maximum sequence length 512, a linear schedule with 6% warmup, and zero weight decay. Commonsense8 uses three epochs and Qwen GLUE-4 uses five; each result averages seeds 42–44. Commonsense8 uses learning rates 3×10^{-3} for the probe and 3×10^{-2} for random RIFT; both Qwen GLUE-4 variants use 10^{-2}. SIFT rates follow Table 8. The GLUE-4 gain is concentrated in RTE (91.1 for random versus 77.1 for the probe).

The reported Commonsense8 probe search covers two rates on Qwen and one on Llama. At the shared rate 3×10^{-3}, probe/random means are 85.4/84.7 on Qwen and 76.9/77.8 on Llama; at 3×10^{-2}, Qwen's probe mean is 76.6. Thus the reported advantage does not establish the probe's fully tuned performance or isolate selection from learning rate and diagonal treatment.

A.4 LLAMA-3.1-8B COMMONSENSE8 RESULTS

Table 11: RIFT achieves Commonsense8 accuracy comparable to calibrated SIFT and exceeds random-support SIFT on Llama-3.1-8B, without task-dependent support calibration.

Method	Params	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
Frozen (zero-shot)	0	5.6	0.2	1.1	5.8	2.9	1.0	1.2	2.2	2.5
Rank-limited budgets										
AdaLoRA	5.11M	74.4	89.0	81.5	96.1	87.6	91.8	76.2	87.7	85.5
DoRA (r=8)	3.57M	75.8	89.9	81.9	96.6	88.3	**92.7**	**80.5**	86.8	86.6
LoRA (r=8)	3.41M	75.8	**90.3**	**82.2**	**96.7**	**88.4**	92.4	**80.5**	88.0	**86.8**
SHiRA (r=8)	3.41M	75.9	90.0	81.6	96.4	87.4	**92.7**	78.9	**88.2**	86.4
DiReFT (r=8)	2.10M	**76.1**	89.5	81.5	96.1	86.8	92.5	79.9	86.4	86.1
LoReFT (r=8)	2.10M	69.6	88.9	81.7	95.9	87.9	91.9	78.7	86.9	85.2
DiReFT (r=1)	**262.2K**	74.2	88.9	80.8	95.0	85.2	91.9	79.0	85.0	85.0
LoReFT (r=1)	**262.2K**	74.4	88.7	81.2	95.3	86.0	91.6	78.7	84.5	85.1
Sub-rank-1 budgets										
VeRA	167.9K	72.6	**88.5**	**81.1**	95.0	**85.8**	89.8	70.0	83.5	83.3
$(IA)^3$	163.8K	**74.8**	88.4	80.9	**95.8**	85.5	91.3	76.3	**84.3**	**84.6**
SIFT	**32.8K**	73.9	87.5	80.3	94.7	84.1	**91.9**	**78.1**	84.2	84.3
Sub-rank-1 budgets, ***calibration-free***										
SIFT (random)	**32.8K**	73.3	87.3	79.6	93.8	81.7	91.0	75.8	82.2	83.1
RIFT (ours)	**32.8K**	**74.3**	**88.5**	**79.7**	**94.0**	**84.0**	**91.5**	**78.3**	**83.9**	**84.3**

Accuracy (%) over three seeds; the frozen row uses one deterministic greedy decode. All rows use generation and answer matching. Avg. is the unweighted eight-task mean. Bold marks the best displayed score or smallest budget within each block.

A.5 QWEN-2.5-7B DETAILED RESULTS

Table 12: RIFT achieves competitive Commonsense8 accuracy on Qwen2.5-7B with far fewer trainable parameters than the evaluated rank-one adapters and no task-dependent support calibration.

Method	Params	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
Frozen (zero-shot)	0	67.4	85.3	66.3	78.1	58.1	91.5	80.5	72.6	75.0
Rank-limited budgets										
AdaLoRA	3.79M	74.4	91.0	**81.8**	**96.4**	**88.6**	96.1	88.4	92.6	**88.7**
DoRA (r=8)	2.64M	74.3	90.0	81.1	96.1	87.2†	**96.7**	**90.0**	92.5	88.5
LoRA (r=8)	2.52M	**74.9**	90.1	81.0†	96.0	86.8†	**96.7**	89.6	92.5	88.5
SHiRA (r=8)	2.52M	74.0	**91.5**	80.2	96.2	88.3	96.2	88.3	**92.7**	88.4
DiReFT (r=8)	1.61M	69.0	90.2	81.2	95.3	84.2	95.3	86.3†	90.7†	86.5
LoReFT (r=8)	1.61M	68.1	89.8	81.3†	95.0	83.6	95.5†	87.5	75.7	84.6
DoRA (r=1)	315.4K	73.6	91.1	82.0	96.3	87.5	95.8	88.0	92.1	88.3
LoRA (r=1)	315.4K	73.4	90.9	82.1	96.2	87.5	95.7	88.1	92.0	88.2
DiReFT (r=1)	**200.7K**	74.7	89.9	81.6	95.1	84.8	96.4	87.6	91.4	87.7
LoReFT (r=1)	**200.7K**	74.1	90.2	81.4	95.1	83.9	95.6	87.1	90.9	87.3
Sub-rank-1 budgets										
VeRA (r=64)	118.3K	70.3	88.9	79.9	93.9	82.4	95.1	85.4	84.0	85.0
$(IA)^3$	114.7K	72.7	90.1	81.0	**95.7**	85.9	95.9	**88.0**	91.2	87.6
SIFT	**28.7K**	**74.0**	**90.4**	**82.1**	**95.7**	**86.6**	**96.1**	87.9	**91.3**	**88.0**
Sub-rank-1 budgets, ***calibration-free***										
SIFT (random)	**28.7K**	73.4	88.7	80.4	94.5	83.1	95.7	88.1	90.7	86.8
RIFT (ours)	**28.6K**	**74.5**	**89.8**	**81.5**	**95.0**	**84.0**	**96.3**	**88.2**	**91.5**	**87.6**

Accuracy (%) over three seeds; the frozen row uses one deterministic greedy decode. Avg. is the unweighted eight-task mean. †At least one collapsed seed is excluded from that cell's mean. Bold marks the best displayed score or smallest budget within each block.

Table 13: RIFT approaches the GLUE-4 performance of larger adapters on Qwen2.5-7B with only 28.6K intervention parameters and no task-dependent support calibration.

Method	Params	MNLI	QNLI	RTE	MRPC	Avg.-4	CoLA
Frozen (zero-shot)	0	69.2	81.9	83.8	75.0	77.5	—
Rank-limited budgets							
AdaLoRA	3.79M	91.0	95.1	88.4	80.0	88.6	62.7
DoRA (r=8)	2.64M	**91.4**	95.6	89.3	85.9	90.5	66.6
LoRA (r=8)	2.52M	91.3	**95.7**	89.4	85.9	90.6	66.5
SHiRA (r=8)	2.52M	90.8	95.3	89.3	84.9	90.1	66.9
DiReFT (r=8)	1.61M	91.1	95.2	**91.7**	**88.9**	**91.7**	**70.0**
LoReFT (r=8)	1.61M	91.0	95.3	90.5	88.8	91.4	66.2
LoRA (r=1)	315.4K	90.7	95.3	90.5	88.2	91.2	66.8
DiReFT (r=1)	**200.7K**	90.6	95.2	90.7	88.6	91.3	68.4
LoReFT (r=1)	**200.7K**	90.4	95.2	90.6†	81.6	89.5	66.4
Sub-rank-1 budgets							
VeRA (r=64)	118.3K	**90.9**	95.3	88.4	81.9	89.1	64.2
$(IA)^3$	114.7K	90.3	94.8	89.3	87.7	90.5	65.7
SIFT	**28.7K**	**91.0**	**95.6**	**91.3**	**88.7**	**91.7**	**65.7**
Sub-rank-1 budgets, ***calibration-free***							
SIFT (random)	28.7K	**90.5**	**95.1**	89.0	82.9	89.4	65.5
RIFT (ours)	**28.6K**	90.1	94.7	**91.1**	**88.2**	**91.0**	**66.7**

Three-seed mean accuracy (%) on MNLI, QNLI, RTE, and MRPC; Avg.-4 is their unweighted mean. CoLA reports Matthews correlation times 100 and is excluded from Avg.-4. †At least one collapsed seed is excluded from the cell mean. Each method uses one learning rate selected by GLUE-4 mean; representation interventions use f7+l7. Bold marks the best displayed score or smallest budget within each block.

A.6 GLUE Comparison Across RoBERTa and GPT-NEO

Table 14: RIFT outperforms random-support SIFT on GLUE-4 but trails the leading PEFT baselines on RoBERTa.

	RoBERTa-base		RoBERTa-large		GPT-Neo-125M	
Method	Adapter params	GLUE-4	Adapter params	GLUE-4	Adapter params	GLUE-4
Head only (encoders)	0	63.0	0	61.6	–	–
Full fine-tuning	124M	86.7	354M	90.1	125.2M	73.5
Rank-one-or-higher budgets						
AdaLoRA	442.7K	80.3	1.18M	84.1	442.7K	73.4
DoRA (r=8)	313.3K	**85.3**	835.6K	**89.8**	313.3K	**76.5**
LoRA (r=8)	294.9K	**85.3**	786.4K	89.6	294.9K	76.4
SHiRA	294.9K	84.5	786.4K	89.0	294.9K	74.7
LoRA (r=1)	36.9K	84.7	98.3K	87.9	36.9K	75.9
LoReFT (r=1)	**18.4K**	83.0	**49.2K**	85.0	**18.4K**	74.3
DiReFT (r=1)	**18.4K**	79.5	**49.2K**	82.6	**18.4K**	76.1
Sub-rank-one budgets						
VeRA (r=64)	20.0K	**83.0**	52.2K	**87.3**	20.0K	**75.0**
$(IA)^3$	18.4K	82.1	49.2K	86.8	18.4K	74.8
SIFT	**3.1K**	81.5	**8.2K**	86.4	**3.1K**	71.0
Sub-rank-one budgets, calibration-free						
SIFT (random)	**3.1K**	73.9	**8.2K**	76.1	**3.1K**	62.6
RIFT (ours)	**3.1K**	**74.8**	**8.2K**	**79.1**	**3.1K**	**71.4**

A.7 GPT-NEO-125M GLUE RESULTS

Table 15: RIFT outperforms random-support SIFT on GPT-Neo-125M GLUE-4 at the same parameter budget, while trailing larger adapters.

Method	Params	MNLI	QNLI	RTE	MRPC	Avg.-4	CoLA
Full fine-tuning	125.2M	75.2	85.3	61.5	71.8	73.5	41.9
Rank-one-or-higher budget							
AdaLoRA	442.7K	**79.1**	85.6	58.1	70.8	73.4	24.1
DoRA (r=8)	313.3K	**79.1**	**86.7**	65.7	74.7	**76.5**	37.1
LoRA (r=8)	294.9K	**79.1**	86.6	65.3	74.4	76.4	**37.2**
SHiRA	294.9K	78.1	85.4	62.2	73.1	74.7	19.8
LoRA (r=1)	36.9K	78.0	86.3	63.9	75.4	75.9	36.3
VeRA (r=64)	20.0K	76.9	85.2	63.4	74.6	75.0	34.6
LoReFT (r=1)	**18.4K**	75.1	85.4	61.6	75.2	74.3	27.8
DiReFT (r=1)	**18.4K**	74.8	84.8	**68.6**	**76.1**	76.1	33.7
Below-rank-one budget							
$(IA)^3$	18.4K	**76.6**	**85.8**	**62.0**	**74.7**	**74.8**	**26.6**
SIFT	**3.1K**	72.0	83.5	56.0†	72.6	71.0	—§
Below-rank-one budget with random support							
SIFT + random	**3.1K**	59.4	70.2	52.4‡	68.4	62.6	-1.3‡
RIFT (ours)	**3.1K**	**71.6**	**82.4**	**60.3**	**71.5**	**71.4**	**20.5**†

Scores average three seeds before the reported failure filtering. MNLI, QNLI, RTE, and MRPC report accuracy (%); Avg.-4 is their unweighted mean. CoLA reports Matthews correlation times 100 and is excluded from Avg.-4. †Some seeds were excluded; ‡all seeds fell at or below chance plus five points, so the raw mean is shown and retained in Avg.-4. RIFT CoLA retains one of three seeds (unfiltered mean: 7.3). §SIFT CoLA is withheld because its runs used zero calibration examples and therefore did not implement gradient-selected support. This does not affect GLUE-4. Full fine-tuning and rank-one LoRA use learning-rate sweeps; other non-sparse baseline rates are transferred from RoBERTa-base. Bold marks the best displayed score or smallest budget within each block.

A.8 RoBERTa GLUE Results

Table 16: RIFT outperforms random-support SIFT on RoBERTa-base GLUE-4 but remains below calibrated SIFT and the leading larger-budget adapters.

Method	Params	MNLI	QNLI	RTE	MRPC	Avg.-4	CoLA
Head only	0	51.8	68.6	55.6†	76.0	63.0	11.5
Full fine-tuning	124M	87.4	93.0	76.7	89.6	86.7	61.4
Rank-one-or-higher budget							
AdaLoRA	442.7K	**87.1**	**92.9**	54.2‡	87.3	80.3	57.6
DoRA (r=8)	313.3K	86.9	92.8	**73.4**	88.1	**85.3**	60.3
LoRA (r=8)	294.9K	87.0	92.7	73.0	**88.7**	**85.3**	**61.0**
SHiRA	294.9K	86.7	92.3	71.8	87.3	84.5	58.9
LoRA (r=1)	36.9K	86.0	92.1	72.9	87.7	84.7	60.4
VeRA (r=64)	20.0K	83.7	91.0	69.9	87.3	83.0	59.3
LoReFT (r=1)	**18.4K**	83.7	91.0	69.3†	87.8	83.0	52.4†
DiReFT (r=1)	**18.4K**	83.7	90.3	58.1†	85.8	79.5	52.9
Below-rank-one budget							
$(IA)^3$	18.4K	**84.2**	**90.5**	68.8	84.7	**82.1**	**53.9**
SIFT	**3.1K**	81.5	89.0	**69.3**	**86.2**	81.5	52.9
Below-rank-one budget with random support							
SIFT + random	**3.1K**	76.9	85.3	56.0†	77.5	73.9	20.9
RIFT (ours)	**3.1K**	**77.8**	**86.2**	**57.1**	**78.0**	**74.8**	**37.9**

Accuracy (%) on development sets; Avg.-4 averages MNLI, QNLI, RTE, and MRPC. CoLA is Matthews correlation times 100 and is excluded from Avg.-4. Parameter counts exclude the trained task head (0.59M for RoBERTa-base; 1.05M for RoBERTa-large). Runs at or below chance plus five points are excluded: †some seeds were excluded; ‡all were, so the raw mean is shown and retained in Avg.-4. Bold marks the best displayed score or smallest budget within each block.

Table 17: RIFT outperforms random-support SIFT on RoBERTa-large GLUE-4 but trails calibrated SIFT and the leading larger-budget adapters.

Method	Params	MNLI	QNLI	RTE	MRPC	Avg.-4	CoLA
Head only	0	52.9	70.2	51.3‡	71.8	61.6	8.7†
Full fine-tuning	354M	90.0	94.6	85.7	90.2	90.1	66.2
Rank-one-or-higher budget							
AdaLoRA	1.18M	**90.5**	**94.8**	64.7	86.4	84.1	62.5
DoRA (r=8)	835.6K	90.2	94.4	**85.2**	**89.1**	**89.8**	**67.3**
LoRA (r=8)	786.4K	90.2	94.5	84.6	89.0	89.6	65.1
SHiRA	786.4K	90.1	94.2	83.0	88.7	89.0	63.6†
LoRA (r=1)	98.3K	90.2	94.4	80.1	86.8	87.9	62.6
VeRA (r=64)	52.2K	89.7	93.7	78.6	87.1	87.3	62.8
LoReFT (r=1)	**49.2K**	89.5	94.0	69.9	86.5	85.0	44.7†
DiReFT (r=1)	**49.2K**	89.5	93.9	65.2	81.6	82.6	55.8
Below-rank-one budget							
$(IA)^3$	49.2K	**89.5**	**93.4**	**76.8**	87.4	**86.8**	**59.7**
SIFT	**8.2K**	88.2	93.0	76.2†	**88.2**	86.4	59.5†
Below-rank-one budget with random support							
SIFT + random	**8.2K**	86.2	89.1	52.6‡	76.3	76.1	28.4
RIFT (ours)	**8.2K**	**87.1**	**91.5**	**58.1**†	**79.7**	**79.1**	**52.6**

Accuracy (%) on development sets; Avg.-4 averages MNLI, QNLI, RTE, and MRPC. CoLA is Matthews correlation times 100 and is excluded from Avg.-4. Parameter counts exclude the trained task head (0.59M for RoBERTa-base; 1.05M for RoBERTa-large). Runs at or below chance plus five points are excluded: †some seeds were excluded; ‡all were, so the raw mean is shown and retained in Avg.-4. Bold marks the best displayed score or smallest budget within each block.

A.9 Detailed Support Robustness

Table 18: RIFT maintains consistent benchmark accuracy across random support draws on three backbones, with the training seed and hyperparameters held fixed.

(a) Qwen2.5-7B: Commonsense8

	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
Mean	74.59	89.94	81.35	94.96	84.18	96.17	88.76	91.12	87.63
SD	0.31	0.16	0.47	0.09	0.66	0.16	0.51	0.58	0.15

(b) Llama-3.1-8B: Commonsense8

	BoolQ	PIQA	SIQA	HellaS.	WinoG.	ARC-e	ARC-c	OBQA	Avg.
Mean	74.01	88.32	79.46	94.09	84.27	91.55	77.38	84.48	84.20
SD	0.62	0.39	0.45	0.15	0.83	0.34	0.41	0.79	0.24

(c) RoBERTa-base: GLUE-4

	MNLI	QNLI	RTE	MRPC	Avg.
Mean	77.82	86.07	56.75	78.48	74.78
SD	0.47	0.27	1.16	0.69	0.44

A.10 Support Calibration Cost

Table 19: RIFT eliminates task-dependent support calibration at approximately matched parameter budgets on Qwen2.5-7B.

Method	Task probe?	Checkpoint support	Dense gradient buffer
SIFT	Yes	Stored indices	1.53 GB
RIFT	No	Stored indices	—

SIFT uses 28,672 coefficients and RIFT uses 28,644. The reported SIFT coordinate-pair payload is 0.44 MB, excluding serialization metadata. Random support is reconstructible from a seed under a fixed sampling protocol, but the reference adapter format stores realized indices. No seed-only checkpoint size is measured here.

A.11 PEAK VRAM RESULTS

Table 20: RIFT achieves modest reductions in peak training memory relative to the other evaluated PEFT methods on Qwen2.5-7B, despite much larger differences in trainable parameter counts.

Method	Params	Peak alloc (GB)	Peak maxlen (GB)	Peak resv (GB)	Mean (GB)	Peak dev (GB)
Frozen (no training)	0	16.4	17.8	16.9	14.2	17.4
RIFT (random)	28.6K	**22.1**	**27.0**	**22.6**	16.3	**23.2**
SIFT	28.7K	23.0	28.5	23.6	16.6	24.1
SIFT + random	28.7K	23.0	28.5	23.5	16.6	24.1
$(IA)^3$	114.7K	22.4	27.4	22.8	**15.9**	23.4
VeRA (r=64)	118.3K	22.6	27.8	23.1	16.2	23.7
DiReFT (r=1)	200.7K	23.5	28.4	23.9	17.6	24.6
LoReFT (r=1)	200.7K	23.5	28.4	24.0	17.6	24.6
LoRA (r=1, q,v)	315.4K	23.0	28.5	23.5	16.2	24.1
DiReFT (r=8)	1.61M	23.5	28.4	24.0	17.5	24.6
LoReFT (r=8)	1.61M	23.5	28.4	24.0	17.4	24.6
SHiRA (r=8, q,v)	2.52M	23.2	28.2	23.6	16.3	24.2
AdaLoRA	3.79M	23.1	28.5	23.5	17.1	24.1

Measurements use batch size 4 and maximum sequence length 512 on an NVIDIA H100 80 GB HBM3; all configurations are measured sequentially on the same GPU. *Peak alloc*: maximum memory allocated to live tensors, measured by `max_memory_allocated`. *Peak maxlen*: peak allocated memory in a separate measurement with inputs padded to the full sequence length. *Peak resv*: maximum memory reserved by the caching allocator, including allocated tensors and unused cached blocks, measured by `max_memory_reserved`. *Mean*: mean allocated memory sampled every 50 ms during the timed steps. *Peak dev*: peak whole-device memory usage measured through `cuda.mem_get_info`, including memory outside the allocator, such as the CUDA context. Bold marks the lowest value among methods undergoing adaptation; the frozen row is an inference-only reference.

A.12 GRADIENT NORMS ON RANDOM SUPPORTS AT INITIALIZATION

We compare the gradient ℓ_1 norm and squared ℓ_2 norm captured by random supports at initialization. These summarize local loss sensitivity on selected coordinates, not measured training progress.

Measurement protocol. We use Qwen2.5-7B on BoolQ, HellaSwag, and WinoGrande, with three disjoint batches of four training examples per task and data seed 42. We examine all 28 block-output intervention sites and the first and last seven prompt positions, yielding 56 positions per batch without repetitions. All edit coefficients are zero. The frozen backbone runs with dropout disabled; activations and backward gradients use bfloat16, followed by float64 products and accumulation for the statistics. The scalar loss F is mean causal language-model cross-entropy over valid completion tokens in the batch; its gradient preserves cancellation between examples.

Let $\boldsymbol{H}_\ell$ contain states at intervened positions and $\boldsymbol{A}_\ell$ the corresponding loss gradients with respect to edited states. At initialization,

$$\boldsymbol{G}_\ell = \left.\frac{\partial F}{\partial \boldsymbol{C}_\ell}\right|_{\boldsymbol{C}=\boldsymbol{0}} = \boldsymbol{H}_\ell^\top \boldsymbol{A}_\ell. \tag{12}$$

Flatten $\boldsymbol{G}_\ell$ to $\boldsymbol{g}_\ell \in \mathbb{R}^N$, with $N = 3584^2 = 12{,}845{,}056$, and express its support $\mathbb{S}_\ell$ in the same indexing. We regenerate ten supports using the training sampler, seeds 1001–1010, and layer-specific seed offsets. Each has $k = 1023$ entries per layer, totaling $K = 28{,}644$ coefficients. The sampler matches the recorded training-code version, but regenerated supports were not checked against saved original masks. Support seeds are reused across batches and tasks.

Norms quantify different aspects of local sensitivity. Let $\boldsymbol{g}_{\ell,\mathbb{S}_\ell}$ denote the gradient restricted to the selected coordinates. For $p \in \{1, 2\}$, define

$$Z_{\ell,p} = \|\boldsymbol{g}_\ell\|_p^p, \qquad X_{\ell,p} = \|\boldsymbol{g}_{\ell,\mathbb{S}_\ell}\|_p^p, \qquad r_{\ell,p} = \frac{Z_{\ell,p}^2}{\sum_{i=1}^N |g_{\ell,i}|^{2p}}. \tag{13}$$

Thus $X_{\ell,1} = \sum_{i\in\mathbb{S}_\ell} |g_{\ell,i}|$ is the captured ℓ_1 norm, whereas $X_{\ell,2} = \sum_{i\in\mathbb{S}_\ell} g_{\ell,i}^2$ is the captured *squared* ℓ_2 norm. Squaring gives larger coordinates more weight. The participation ratio $r_{\ell,p}$ quantifies concentration, not matrix rank, intrinsic dimension, or the number of useful coordinates.

For an ordinary gradient step, the first-order predicted loss reduction is $\eta \sum_\ell X_{\ell,2}$. For an idealized AdamW update with zero moment estimates, bias correction, and step size η, zero coefficients instead give

$$-\sum_\ell \sum_{i\in\mathbb{S}_\ell} g_{\ell,i}\Delta\theta_{\ell,i} = \eta \sum_\ell \sum_{i\in\mathbb{S}_\ell} \frac{g_{\ell,i}^2}{|g_{\ell,i}| + \epsilon} \approx \eta \sum_\ell X_{\ell,1}. \tag{14}$$

This motivates the ℓ_1 diagnostic through Adam's approximate sign normalization (Balles & Hennig, 2018). Omitting $\epsilon = 10^{-8}$ changes the unscaled sum by at most $K\epsilon$, giving relative error below 7.45×10^{-6} on the measured gradients. This bounds the approximation, not actual loss changes or finite-precision updates. Zero coefficients do not imply zero optimizer moments: the training runs use learning-rate warmup, during which moments can accumulate before coefficients change. The diagnostic therefore does not replay the first nonzero training update or characterize subsequent optimization.

Sampling prediction and aggregation. For a fixed gradient and a uniform size-k support drawn without replacement,

$$\mu_{\ell,p} = \mathbb{E}[X_{\ell,p}] = \frac{k}{N} Z_{\ell,p}, \qquad \frac{\mathrm{Var}(X_{\ell,p})}{\mu_{\ell,p}^2} = \frac{N-k}{k(N-1)} \left(\frac{N}{r_{\ell,p}} - 1\right). \tag{15}$$

Under independent layer draws, the predicted coefficient of variation (CV) for the whole-model total is $\sqrt{\sum_\ell \mathrm{Var}(X_{\ell,p})}/\sum_\ell \mu_{\ell,p}$. Distinct seeded streams implement this sampling model. For each task–batch pair, we sum $X_{\ell,p}$ over layers for each mask, then divide the sample standard deviation across ten masks by their mean. Summary CVs average the nine batch-level CVs equally; reused supports are not treated as 90 independent draws.

Captured ℓ_1 norms vary less across supports. Across the nine task–batch pairs, the median over layers of $r_{\ell,2}/N$ ranges from 1.06×10^{-4} to 1.27×10^{-4}, compared with 0.0436–0.0664 for $r_{\ell,1}/N$. The mean observed CV is 2.54% for captured ℓ_1 norms and 59.92% for squared ℓ_2 norms; predictions are 3.14% and 67.70%, respectively (Table 21). Per-batch maximum-to-minimum ratios range from 1.059 to 1.134 for ℓ_1 and 3.35 to 14.83 for squared ℓ_2. These observations establish a difference in relative variability, without requiring exact agreement between ten-draw CV estimates and their predictions.

Table 21: Random supports capture initial gradient ℓ_1 norms with substantially lower relative variability than squared ℓ_2 norms on Qwen2.5-7B.

		Squared ℓ_2 ($p = 2$)		ℓ_1 ($p = 1$)	
Task	Batch	Observed	Predicted	Observed	Predicted
BoolQ	1	58.80	69.32	3.34	3.29
BoolQ	2	51.42	65.97	2.78	3.32
BoolQ	3	71.48	66.28	3.89	3.31
HellaSwag	1	37.37	70.30	1.71	3.33
HellaSwag	2	49.61	72.09	1.99	3.33
HellaSwag	3	42.84	73.36	1.81	3.33
WinoGrande	1	72.24	63.80	2.54	2.77
WinoGrande	2	54.49	63.66	2.21	2.78
WinoGrande	3	101.07	64.50	2.63	2.80
Mean		59.92	67.70	2.54	3.14

Association with final accuracy remains unresolved. For each task and support seed, we average the whole-model statistics over three diagnostic batches and match them to the random-RIFT accuracy record by method, task, and support seed. We average norm statistics, not gradients. Training uses seed 42, learning rate 0.03, three epochs, a linear schedule with 6% warmup, and zero weight decay. Spearman correlations use average ranks for tied accuracies; two-sided tests enumerate all $10!$ permutations of accuracy labels (Table 22). None of the six unadjusted tests reaches $p < 0.05$.

Table 22: Across ten random support draws per task, neither captured initial gradient norm statistic shows a statistically significant association with final accuracy in the unadjusted Spearman tests.

Task	Quantity	Pearson r	Spearman ρ	p
BoolQ	Squared ℓ_2	-0.421	+0.006	0.994
BoolQ	ℓ_1	-0.059	+0.111	0.761
HellaSwag	Squared ℓ_2	-0.315	-0.146	0.685
HellaSwag	ℓ_1	-0.563	-0.518	0.128
WinoGrande	Squared ℓ_2	+0.189	+0.399	0.257
WinoGrande	ℓ_1	+0.400	+0.448	0.199

Interpretation is limited to initialization. The initial gradient quantity relevant to a zero-moment AdamW approximation has low relative variability across the tested random supports. This does not establish similar training trajectories or explain final accuracy. We do not measure subsequent optimizer moments, actual loss changes, other backbones, or weight-space support here. Stable captured norms also do not imply large or optimal fractions of the full gradient norm.