When Does Adaptive LoRA Rank Allocation Help? Signal Reliability and Resource Accounting

Anonymous authors
Paper under double-blind review

Abstract

Adaptive LoRA redistributes a limited rank budget, yet final accuracy alone does not isolate the benefit of rank placement. Allocation scores depend on existing ranks, while rank transfers can change parameter cost and model state. We analyze the scores and test transfers at fixed active parameter count. For a score based on loss progress, a sign change shared by all modules can reverse their rankings even when the ordering of score magnitudes persists. We separate the immediate loss change caused by a transfer from the loss change during continued training. Experiments cover eight GLUE tasks and quantized Llama commonsense workloads. On commonsense, Spearman correlations between consecutive measurements average 0.944 for score magnitudes and -0.155 for signed scores. At selected RTE checkpoints, online allocation activates 16.4% more parameters than uniform at equal total rank. At a shared RTE checkpoint, one of three module pairs yields lower final loss after moving rank to the higher-scored module, but a less favorable loss change over 20 updates than the reverse transfer. Synthetic controls show that concentrating rank resolves local capacity shortages, while repeated adjustment helps after demand shifts.

1 Introduction

Low-rank adaptation (LoRA) reduces the cost of fine-tuning by learning compact updates to a frozen pretrained model (Hu et al., 2022). Its effectiveness is consistent with evidence of low intrinsic dimensionality in many adaptation problems (Aghajanyan et al., 2021), but choosing the rank still determines how much capacity each projection receives. Uniform allocation makes this choice simple by assigning the same rank throughout the model. Adaptive allocation instead uses module-importance scores to distribute capacity unevenly, aiming to accommodate differences in task-specific demand within a limited budget (Zhang et al., 2023b; Ding et al., 2023; He et al., 2025).

Realizing this promise requires both *signal reliability*—identifying modules that benefit from more rank—and *resource accounting*—establishing what the allocation actually costs. Importance scores are measured under an existing allocation: gradient-based scores, for example, use the loss's sensitivity to parameters as a proxy for capacity demand. Yet a module's assigned rank can itself affect its gradient magnitude. In a study of Group Relative Policy Optimization (GRPO), random rank assignments reproduce the gradient amplification in higher-rank modules seen with importance-based assignments (Sawant, 2026). The random control shows that assigned capacity itself can make a module appear important. Allocating further rank on that basis can reinforce the initial assignment while neglecting demand elsewhere. Signal reliability therefore concerns whether a score identifies useful changes in capacity, beyond describing the current allocation.

Even an informative score leaves a second question: does the resulting improvement come from better placement of capacity, or from a change in the resources used? Total rank sums the active ranks across modules, but each rank unit has a parameter cost determined by the projection dimensions (Hu et al., 2022). In DeBERTa, one unit costs 1,536 parameters in an attention projection and 3,840 in a feed-forward projection; transferring it from attention to feed-forward adds 2,304 parameters without changing total rank. Training history matters as well: AdaLoRA starts with a

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.

Table 1: Key observations from Tables 5 and 6 and Section 4.3. The RTE transfer example uses the first of three module pairs in Table 6. Positive loss differences favor moving rank to the higher-scored module; units are nats/example. Correlations and parameter counts are run means. Bold highlights the main comparisons.

Comparison	Observation	Interpretation
Online score, commonsense	Correlation between consecutive measurements: magnitudes **0.944**; signed scores $\mathbf{-0.155}$	Sign changes obscure a persistent ordering of score magnitudes.
Active parameters, RTE	Uniform: 995,328; online: 1,158,144; both total rank 432	Online activates **16.4% more parameters** at equal total rank.
RTE rank transfer (example)	Final loss difference $\mathbf{+0.017684}$; difference in loss reduction over 20 updates $\mathbf{-0.005055}$	Lower final loss, but less favorable learning after the transfer.

larger budget before reducing it to the target (Zhang et al., 2023b). These capacity differences are one source of performance variation. Changes to optimization are another: GoRA combines allocation with initialization and scaling (He et al., 2025), while LoRA+ changes factor learning rates at fixed rank (Hayou et al., 2024). Resource accounting and component comparisons are thus needed to distinguish the effect of rank placement from additional capacity or altered optimization.

To disentangle these effects, we analyze allocation scores and resource costs on DeBERTa-v3-base with eight GLUE tasks (He et al., 2023; Wang et al., 2019) and Llama-3.1-8B with 4-bit QLoRA on eight commonsense tasks (Grattafiori et al., 2024; Dettmers et al., 2023). A 60-run study crosses GoRA allocation with its initialization/scaling package (He et al., 2025). At a shared checkpoint, we also transfer rank along or against the score ordering while holding rank and active parameter count fixed. Losses measured immediately after each transfer and after continued training separate the initial perturbation from subsequent learning.

We compare uniform ranks, ranks chosen once before training (one-shot), and ranks revised during training (online). Their mean commonsense accuracies differ by only 0.28 percentage points, but the score measurements and rank transfers reveal differences beyond accuracy (Table 1). Changes in the sign of loss progress reverse module rankings even when the ordering of score magnitudes persists. In one RTE module pair, moving rank to the higher-scored module yields lower final loss, yet continued training favors the reverse transfer. Synthetic controls explain when redistribution helps: uniform allocation can leave some modules short of capacity despite a sufficient total budget, and demand shifts can make a useful initial allocation obsolete.

Our contributions are:

- **A shared sign change can reverse module rankings.** We derive an exact Spearman identity for the learning-progress score. It explains why correlations between successive measurements can be negative even when score-magnitude rankings persist, separating changes in global loss from changes in relative module scores.
- **Equal total rank does not guarantee equal parameter cost.** We quantify active parameters, cumulative rank-steps, and preallocated storage, then identify transfers that preserve both total rank and active parameter count. These transfers allow us to test score-directed decisions at fixed parameter cost.
- **Final loss and subsequent learning do not always favor the same transfer direction.** We separate the immediate effect of a rank transfer from the loss change during continued training. One of three RTE module pairs exhibits this disagreement over 20 updates. Synthetic controls further distinguish resolving a local rank shortage from tracking a shift in demand.

2 Related work

Allocation signals and their reliability. Allocation scores draw on different notions of sensitivity. AdaLoRA ranks components using smoothed sensitivity and uncertainty estimates (Zhang

et al., 2023b); GoRA uses a gradient snapshot to allocate module budgets before fine-tuning (He et al., 2025). ShapLoRA estimates importance through Shapley-inspired sensitivity (Zhao et al., 2026), and RSRA probes representation sensitivity (Liu et al., 2026). A score can also depend on the capacity already assigned: Sawant (2026) shows that random rank assignments reproduce rank-dependent gradient amplification under GRPO. We examine a different source of variation in supervised fine-tuning, where a shared loss-progress sign reverses temporal score orderings.

Allocation timing and resource accounting. DyLoRA trains a range of usable ranks (Valipour et al., 2023). AdaLoRA, SoRA, and dynamic-rank DoRA prune a larger parameterization (Zhang et al., 2023b; Ding et al., 2023; Mao et al., 2024), whereas IncreLoRA and AROMA add components during optimization (Zhang et al., 2023a; Sheng et al., 2025). PARA allocates rank through a global spectral threshold after optimization (Kumaravelu et al., 2026); CCM-LoRA selects directions per input (Rafiuddin & Abdullah, 2026). As Table 7 in Appendix A summarizes, the same terminal rank can represent different training and inference costs. GoRA explicitly converts a parameter budget into module ranks using projection dimensions (He et al., 2025). Our accounting builds on this distinction between rank and parameter count and extends the comparison to active capacity, stored factors, and the training trajectory.

Optimization effects beyond allocation. Optimization can change without changing the allocation: LoRA+ adjusts factor learning-rate ratios (Hayou et al., 2024), while weight-decomposed DoRA separates magnitude and direction (Liu et al., 2024). MoRA changes the update parameterization (Jiang et al., 2024), and QLoRA quantizes the base model (Dettmers et al., 2023). GoRA combines allocation with gradient-based initialization and scaling (He et al., 2025). Our component study separates these GoRA choices; checkpoint interventions then measure the immediate and subsequent effects of changing rank at fixed parameter cost.

3 Analytical framework

We test whether moving rank toward higher-scored modules improves learning without increasing the active parameter count. The cost calculation builds on LoRA's factorization (Hu et al., 2022), AdaLoRA's total-rank budget (Zhang et al., 2023b), and GoRA's dimension-aware parameter accounting (He et al., 2025). We then analyze the score and distinguish the immediate effect of a transfer from learning afterward.

3.1 Rank definitions and resource costs

For module i, let $W_i \in \mathbb{R}^{d_{\text{out},i} \times d_{\text{in},i}}$ and let $r_i(t)$ be the number of active factor components at training step t. For $r_i(t) > 0$, ordinary LoRA uses (Hu et al., 2022)

$$\Delta W_i(t) = \frac{\alpha}{r_i(t)} B_i[:, 1 : r_i(t)] A_i[1 : r_i(t), :]. \tag{1}$$

Module rank and total rank. Module rank $r_i(t)$ counts active factor pairs, each consisting of a column of B_i and a row of A_i. Total rank is the sum of these counts across the adapted modules; at the terminal step T, we denote it by B_T, following the budget interpretation in AdaLoRA (Zhang et al., 2023b). Increasing one module's rank and decreasing another's by one redistributes a rank unit while preserving total rank. At rank zero, a module uses only its frozen base weight. With per-rank parameter cost $c_i = d_{\text{in},i} + d_{\text{out},i}$, the resource measures are

$$B_T = \sum_i r_i(T), \qquad P_{\text{active}}(t) = \sum_i c_i r_i(t), \tag{2}$$

$$B_{\text{step}} = \sum_{t=1}^{T} \sum_i r_i(t), \qquad P_{\text{inst}} = \sum_i c_i R_i, \tag{3}$$

where B_T is terminal total rank, $P_{\text{active}}(t)$ counts active parameters, B_{step} accumulates rank over training, and P_{inst} counts stored factor parameters at allocated tensor capacity R_i. Classification heads are counted separately. AdaLoRA adds one trainable scalar per rank component (Zhang et al.,

2023b), giving active count $\sum_i (c_i+1)r_i$. Rank bookkeeping and optimizer state are separate storage costs.

For DeBERTa, transferring one rank unit from attention (1,536 parameters per rank) to feed-forward (3,840) adds 2,304 parameters while preserving B_T. Main comparisons match terminal total rank and report actual parameter counts. Interventions instead transfer rank between equal-cost modules, preserving both total rank and active parameters. A common checkpoint and identical continuation batches match training history. Rank-steps describe the capacity trajectory; runtime also depends on geometry, sequence lengths, base-model computation, and measurement work.

3.2 Allocation policies and component comparisons

Uniform allocation provides the equal-rank reference. One-shot and online policies distinguish assigning capacity once from revisiting it during training. AdaLoRA adds budget shrinkage; the GoRA component study separates allocation from initialization/scaling.

Uniform and one-shot allocation. Our one-shot baseline uses the weight–gradient sensitivity underlying importance-based allocation (Zhang et al., 2023b; He et al., 2025). It accumulates gradients over an initial training-data panel and scores DeBERTa modules by $s_i = \sum_{ab} |W_{i,ab}\bar{g}_{i,ab}|$. Min–max normalization followed by capped integer allocation assigns ranks once. The quantized decoder uses $\sum_{p\in\{A_i,B_i\}} \|\bar{g}_p\|_1$, since its base weights do not provide floating-point gradients. Both paths retain ordinary LoRA initialization. The GoRA component study instead uses dimension-aware mean sensitivity (He et al., 2025) (Appendix C).

Online learning-progress allocation. We construct a learning-progress score from gradient magnitudes and the observed change in loss. This score lets us study how a loss change shared by all modules affects their ranking; AdaLoRA and GoRA remain separate comparison methods. Let k index allocation events at steps t_k, separated by $K = 100$ updates, and let m be the number of adapted modules. We measure calibration loss ℓ_k on a small training-data subset and obtain factor gradients $g_{p,k}$ from the pending training update. The instantaneous score and its exponential moving average (EMA) are

$$q_{i,k} = \max\left\{\eta_k \sum_{p\in\{A_i,B_i\}} \|g_{p,k}\|_F^2, 0\right\}, \qquad w_{i,k} = \frac{q_{i,k}}{\sum_j q_{j,k}}, \tag{4}$$

$$G_{i,k} = \frac{(\ell_{k-1} - \ell_k) w_{i,k}}{r_i(t_k^-)K}, \qquad z_{i,k} = 0.9 z_{i,k-1} + 0.1 G_{i,k}. \tag{5}$$

Here η_k is the current learning rate, and t_k^- denotes the state just before allocation. Gradients include decoder accumulation and come from the training update, not the calibration subset. We set $w_i = 1/m$ when $\sum_i q_i \le 10^{-8}$, initialize $z_{i,0} = 0$, and set the first loss difference to zero. Negative progress is retained in G; min–max normalization makes allocation weights nonnegative. The score distributes global loss progress through current gradients; it does not directly measure the benefit of adding rank.

For target total rank B, a capped floor-and-remainder rule distributes $B - mr_{\min}$ units above the floor $r_{\min} = 1$ (Appendix B). Rank reallocation stops for the final 15% of training. GLUE resamples its calibration subset at each event; the decoder keeps a fixed subset. Equation 5 therefore includes loss variation from resampling on GLUE.

AdaLoRA and the component study. AdaLoRA begins with larger SVD-like factors and prunes importance-ranked components toward its target (Zhang et al., 2023b). We use the PEFT implementation and retain only target-rank evaluations for its headline GLUE score. The component study is a 2×2 design: uniform versus GoRA allocation, crossed with ordinary versus GoRA initialization/scaling. The sum-sensitivity one-shot baseline is a fifth arm. Initialization and scaling are treated as one joint component.

3.3 Diagnosing signal reliability

We ask how concentrated the scores are and whether their ordering persists across measurements. Signed scores retain positive and negative values; score magnitudes are their absolute values. We analyze both, then test rank transfers in Section 4.5.

For instantaneous scores, set $p_{i,k} = \max(G_{i,k}, 0) / \sum_j \max(G_{j,k}, 0)$. We report normalized entropy $H_k = -\sum_i p_{i,k} \log p_{i,k} / \log m$ and the coefficient of variation $\mathrm{CV}_k = \mathrm{SD}_{\mathrm{pop}}(G_{:,k}) / |\mathrm{mean}(G_{:,k})|$. Undefined events are omitted, not assigned zero. Entropy measures concentration of positive scores; CV measures relative dispersion of the signed scores.

The Spearman correlation $\rho(G_{:,k}, G_{:,k+1})$ compares module rankings at consecutive measurements. The correlation $\rho(z_{:,k}, G_{:,k+1})$ instead tests whether the smoothed allocation score predicts the next instantaneous score. These correlations describe score behavior over time; rank transfers test whether following the scores improves learning. We average valid event statistics within each run and then across runs, giving each run equal weight.

4 Evaluating allocation signals and resource costs

We first compare complete allocation policies, then test individual rank transfers. Task scores and resource records establish what differs between policies; score measurements and training from a shared checkpoint examine their allocation decisions.

4.1 Experimental setup

The encoder experiments adapt 72 projections in DeBERTa-v3-base (He et al., 2023): query, key, value, attention output, and both feed-forward projections per layer. Terminal rank 432 corresponds to uniform rank six. GLUE metrics are F1 for MRPC/QQP, Matthews correlation for CoLA, Pearson correlation for STS-B, and accuracy otherwise; MNLI uses the matched split (Wang et al., 2019). We retain these task-specific metrics without averaging them.

The 60-run component study uses RTE, CoLA, STS-B, and MNLI with a rank-four reference. Allocation-enabled arms realize total ranks 208–271 after dimension-aware rounding and clipping; uniform and initialization-only use 288.

The decoder experiments adapt 224 projections in Llama-3.1-8B (Grattafiori et al., 2024) with NF4 QLoRA (Dettmers et al., 2023), targeting total rank 3,584. Commonsense training uses the first 30,000 merged instruction examples. Evaluation uses up to 300 examples per task across eight tasks: the sample-weighted mean covers 2,399 examples, with Winogrande scored by conditional likelihood. GSM8K uses 500 fixed test examples (Cobbe et al., 2021). Task definitions and scoring appear in Appendix A.2.

Table 8 summarizes optimization. GLUE uses the best scheduled validation metric, with AdaLoRA candidates restricted to target-rank events. Commonsense uses the end-of-training checkpoint. Appendix A specifies data selection, repetitions, and checkpoint rules, including GSM8K selection; Appendix E gives resource records.

4.2 Task performance across allocation policies

Encoder and component comparisons. Table 2 reports the main GLUE results. The one-shot baseline improves SST-2 by 0.11 points and lowers CoLA by 1.27. Online allocation improves MRPC by 0.09 points and MNLI by 0.16, while lowering CoLA by 1.10. AdaLoRA has the highest mean on RTE, STS-B, QNLI, and MNLI. Its training trajectory uses more active rank than the other policies, as quantified below.

The component results in Table 3 separate changes in allocation from changes in initialization/scaling. With uniform rank fixed, initialization/scaling lowers RTE by 1.32 points and produces smaller changes on the other three tasks. The combined arm is 1.81 points below uniform on RTE and 0.01 above it on MNLI. The allocation-only arm achieves 90.21 on MNLI while using fewer active parameters than uniform rank four, illustrating a different quality–capacity trade-off. The one-shot arm improves CoLA by 0.17 points, with a standard deviation of 1.90.

Table 2: GLUE validation results at target total rank 432. All metrics are multiplied by 100; entries are means ± population standard deviations over three runs. Bold marks the highest displayed mean per task, including ties. Textual differences use unrounded values.

Task	Uniform	One-shot	Online	AdaLoRA
RTE	87.24±0.90	87.00±0.78	87.00±0.78	**87.36**±0.00
MRPC	93.53±0.25	93.52±0.22	**93.61**±0.10	92.95±0.15
CoLA	**70.92**±0.30	69.66±0.22	69.83±0.25	69.94±0.68
STS-B	90.77±0.60	90.90±0.18	90.91±0.34	**91.38**±0.08
SST-2	95.68±0.24	**95.80**±0.27	95.49±0.19	95.57±0.11
QNLI	94.23±0.04	94.34±0.13	94.19±0.12	**94.39**±0.15
MNLI	90.26±0.11	90.34±0.11	90.42±0.07	**90.56**±0.05
QQP	**88.76**±0.15	**88.76**±0.20	88.62±0.18	88.74±0.02

Table 3: Allocation × initialization/scaling study, plus the sum-sensitivity one-shot baseline. Metrics are multiplied by 100 (mean ± population standard deviation, three runs). Allocation and combined arms realize total ranks 208–271; the others use 288. Bold marks the highest displayed mean per task.

Task	Uniform	One-shot	Allocation	Init./scaling	Combined
RTE	**87.85**±0.74	87.12±0.34	87.36±0.29	86.52±0.74	86.04±0.85
CoLA	70.68±0.28	**70.85**±1.90	69.66±0.16	70.36±0.34	70.30±0.54
STS-B	**90.94**±0.40	90.93±0.04	90.70±0.08	90.88±0.05	90.90±0.10
MNLI	90.13±0.05	90.15±0.06	**90.21**±0.12	90.01±0.17	90.14±0.12

The decoder results are in Table 4. Under the specified commonsense protocol, uniform, one-shot, and online allocation score 74.66, 74.59, and 74.38, respectively; AdaLoRA scores 72.04. Their conditional-likelihood Winogrande scores are also close. On GSM8K, online and uniform allocation achieve best scheduled accuracies of 65.40 and 64.80, respectively.

Appendix D reports all 46 method-minus-uniform paired intervals: 24 main GLUE, 16 component, and six decoder comparisons. The intervals summarize variation across matched training runs and reinforce the task-dependent pattern of the results.

4.3 Resource use at matched total rank

The DeBERTa implementation preallocates rank-12 factors for *all* ordinary LoRA policies, including inactive slices. On RTE this is 1,992,194 trainable parameters with the classifier. AdaLoRA stores 1,993,058, differing by its 864 trainable singular-value scalars. Stored parameter counts are therefore nearly identical across these implementations.

Active parameters tell a different story. Uniform rank six uses 995,328 LoRA parameters; the RTE one-shot and online allocations use means of 1,153,536 and 1,158,144 despite having the same total rank. At the decoder scale, uniform rank 16 activates 41.94 million of 83.89 million stored LoRA parameters. The one-shot allocation changes the active count to roughly 48 million by assigning rank to differently sized modules. These differences follow directly from Equation 2.

Training history introduces a further difference. AdaLoRA's main GLUE trajectories consume approximately $1.79\times$ the rank-steps of uniform LoRA. On RTE, the totals are 1.01 million for uniform and one-shot, 1.04 million for online, and 1.81 million for AdaLoRA. Online's excess includes its initial rank-eight trajectory; calibration passes add measurement work. Recorded RTE training times average 637, 643, 648, and 788 seconds, respectively, under the original execution conditions. These timings include in-loop calibration.

The minimum-budget boundary. At total rank 144, online-minus-uniform changes on RTE, MRPC, CoLA, and MNLI are -0.60, -0.05, -0.34, and -0.14 points. At rank 72, the 72 modules and floor $r_i \geq 1$ force $r_i = 1$ throughout training. Differences then arise from measurement/training paths rather than redistribution. Appendix F retains these runs as a degenerate-allocation control.

Table 4: Decoder accuracy (%) under the protocol in Appendix A.2. Commonsense entries are means ± population standard deviations over three runs; GSM8K reports best scheduled accuracy. The eight-task mean is weighted by example count and includes conditional-likelihood Winogrande. Bold marks the highest displayed mean per column.

Method	Eight-task mean	Winogrande	GSM8K
Uniform	**74.66**±0.31	59.11±1.37	64.80
One-shot	74.59±0.61	**59.56**±0.42	65.00
Online	74.38±0.68	59.22±1.29	**65.40**
AdaLoRA	72.04±0.29	54.33±0.00	62.20

Table 5: Online score diagnostics, averaged within runs and then across runs. Signed (ρ_G) and Magnitude ($\rho_{|G|}$) are Spearman correlations between consecutive measurements. Sign flips reports the fraction of valid adjacent pairs with opposite loss-progress signs; Next score correlates the current EMA with the next instantaneous score. Bold highlights the commonsense comparison.

Task	Entropy	CV	Signed	Magnitude	Sign flips	Next score
RTE	0.901	0.978	-0.016	0.545	0.479	0.172
MRPC	0.862	1.289	-0.261	0.540	0.731	0.042
CoLA	0.839	1.380	-0.233	0.592	0.692	0.000
STS-B	0.881	1.068	-0.173	0.677	0.635	0.048
SST-2	0.782	1.683	-0.203	0.738	0.653	-0.043
QNLI	0.822	1.613	-0.258	0.773	0.675	-0.028
MNLI	0.836	1.594	-0.207	0.793	0.644	-0.046
QQP	0.840	1.532	-0.259	0.767	0.678	-0.022
Commonsense	0.725	2.555	**-0.155**	**0.944**	0.583	-0.060

4.4 Score signs change while magnitude rankings persist

Table 5 compares correlations between consecutive measurements using signed scores and their magnitudes. On GLUE, the mean signed-score correlation is negative for every task, ranging from -0.261 to -0.016; mean magnitude correlations are 0.540–0.793. The contrast is stronger on commonsense, at -0.155 versus 0.944. Module rankings by score magnitude therefore persist even when the signed rankings reverse.

The difference follows from the score definition. Write $G_{i,k} = a_k v_{i,k}$, where $a_k = (\ell_{k-1} - \ell_k)/K$ and $v_{i,k} = w_{i,k}/r_i(t_k^-) \geq 0$. For nondegenerate event pairs, Spearman invariance gives

$$\rho(G_{:,k}, G_{:,k+1}) = \operatorname{sgn}(a_k a_{k+1})\rho(|G_{:,k}|, |G_{:,k+1}|). \tag{6}$$

The global factor changes sign in 48–73% of adjacent GLUE measurements and 58% of commonsense measurements. Equation 6 holds for every valid pair in the stored traces to numerical precision. A negative signed correlation can therefore coexist with a persistent ordering of normalized squared-gradient magnitudes per active rank. EMA smoothing integrates these reversals rather than removing the shared factor by construction.

On GLUE, resampling the calibration subset adds variation to the loss difference, alongside changes from training and rank allocation. Correlation with the next *signed* score also combines changes in magnitude rankings with changes in the shared sign. Appendix G reports event counts and individual traces.

4.5 Does moving rank to higher-scored modules help?

We test whether transfers aligned with the score ordering improve subsequent learning while holding total rank and active parameter count fixed. For the RTE intervention, we select three disjoint high/low-score pairs using the EMA score z from Equation 5, matching projection type and per-rank parameter cost. The aligned branch transfers one rank unit from the lower-scored to the higher-scored module; the anti-aligned branch reverses this transfer. Each added component retains its stored A row, starts with a zero B column, and has its Adam moments cleared. All branches restore

Table 6: Rank transfers for three RTE module pairs at fixed total rank and active parameter count. Losses are validation cross-entropy in nats/example. Bold highlights the pair whose final loss and loss change favor opposite directions.

Module pair	Initial difference $L_0^- - L_0^+$	Loss-reduction difference D (20 updates)	Final difference F
1	**+0.022739**	**−0.005055**	**+0.017684**
2	+0.002735	+0.006150	+0.008885
3	−0.002083	+0.008223	+0.006140

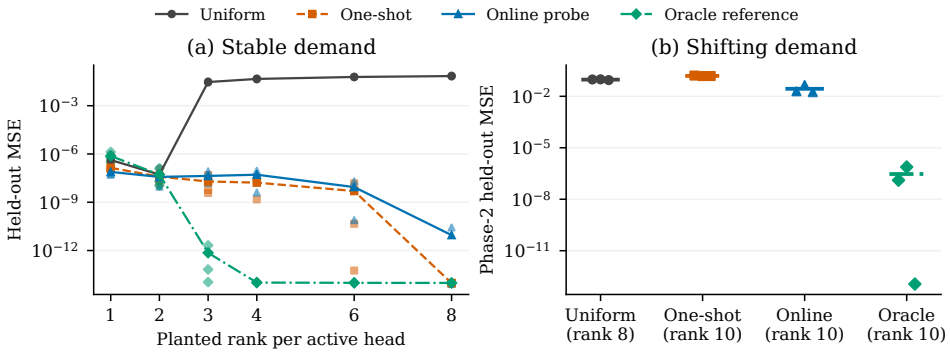


Figure 1: Synthetic controls on logarithmic axes. Points are individual runs; lines/short bars are means. (a) Two of eight heads have planted demand; uniform rank is two, and the oracle uses total rank $2d$. (b) Demand switches between head groups. During phase two, uniform uses total rank eight and the others ten. Oracle references use known demand and the same number of training updates.

the same model, optimizer, scheduler, allocator, and random state, then use the same 20 continuation batches. Total rank remains 432 and active LoRA parameters remain 995,328.

Let L_0^a and L_h^a be validation cross-entropy immediately after intervention and after $h = 20$ updates, with $a \in \{+, -\}$ denoting aligned and anti-aligned branches. We report

$$D = (L_0^+ - L_h^+) - (L_0^- - L_h^-), \qquad F = L_h^- - L_h^+. \tag{7}$$

Positive values favor moving rank to the higher-scored module. F compares final losses; D compares how much loss falls during continued training, with increases counted as negative reductions. The identity $F = (L_0^- - L_0^+) + D$ separates the initial loss difference from the change during training. D includes the effects of rank-dependent scaling and changed component states.

Two of the three D values favor aligned allocation; all three final loss differences F do. Module pair 1, previewed in Table 1, explains the difference: its aligned branch starts with a loss advantage of 0.022739, then loses 0.005055 of that advantage during continued training. Its lower final loss thus originates in the immediate effect of the transfer. In pairs 2 and 3, continued training instead shifts the loss difference in favor of aligned allocation. The two unchanged-allocation replays agree exactly in both initial and final validation loss.

5 When allocation helps: controlled capacity demands

Synthetic controls test two reasons to reallocate: a local capacity shortage despite sufficient total rank, and a demand shift that makes a fixed allocation obsolete. Independent modules, called heads, each apply a linear map followed by GELU and learn planted low-rank updates under noiseless mean squared error (MSE). The online policy replaces the learning-progress score with a *direct module probe*: temporarily update one module, measure calibration-loss reduction per active rank, then restore its parameters. The probe evaluates an update at the current rank, retaining the same allocator and factor storage.

5.1 Concentrating rank resolves local capacity shortages

Eight 128-dimensional heads share total rank 16; two contain planted updates of rank $d \in \{1, 2, 3, 4, 6, 8\}$. Uniform rank two can represent updates with $d \leq 2$. For $d > 2$, its MSE rises from approximately 0.030 to 0.070, while online probing yields 4.34×10^{-8} at $d = 3$ and 9.07×10^{-12} at $d = 8$. One-shot allocation also resolves the shortage, reaching 1.96×10^{-8} and 9.46×10^{-15} at the same demands. Figure 1(a) and Appendix I report errors and dispersions.

5.2 Online adjustment helps when demand moves

The second control has four heads and a phase change at step 800 of 1,600. Rank-four demand moves from the first two heads to the last two. Fixed one-shot allocation leaves 20% of its rank on the second group; online probing increases that share to 66.7% on average (Figure 2). At phase-two total rank ten, MSE is 0.1546 for one-shot and 0.0274 for online. Uniform rank two provides a lower-budget reference, with total rank eight and MSE 0.0938. The oracle reference uses the known phase support and achieves 2.95×10^{-7}.

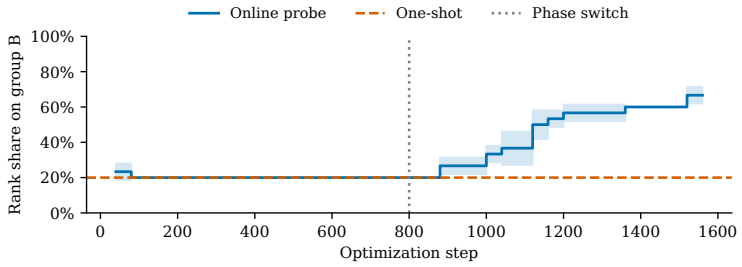


Figure 2: Rank share on the phase-two heads (online mean $\pm$ population standard deviation, three runs). Steps mark allocation events.

5.3 Implications for adaptive LoRA

The stable-demand control separates choosing an allocation from continually revising it. One-shot allocation already resolves the planted rank shortage, leaving little error for online adjustment to remove. After demand shifts, however, its fixed allocation leaves most capacity on the previously active heads. Online probing redirects rank and lowers phase-two error at the same phase-two budget. These controls suggest prioritizing capacity placement under stable demand and evaluating the benefits of continued reallocation when demand changes.

Score design should distinguish shared loss-progress signs from module information. Compare candidate rules at matched active parameter cost, accounting for projection dimensions. From a common training state, measure loss immediately after transfer and after further training to separate initial perturbations from subsequent learning (Equation 7).

6 Conclusion

We provide an analytical and experimental framework for testing whether adaptive rank allocation makes better use of capacity. For the learning-progress score, an exact Spearman identity shows how a shared sign change can reverse module rankings while magnitude rankings persist. Resource accounting reveals that equal total rank can conceal different active parameter costs and training trajectories. RTE transfers at matched cost show that lower final loss can arise from an immediate transfer effect even when subsequent learning favors the reverse direction. Synthetic controls identify two distinct benefits: concentrating rank resolves stable local shortages, while online probing follows demand shifts and lowers error at a matched phase-two rank budget. These findings yield a concrete evaluation principle: test rank transfers from the same training state at matched active parameter cost, and measure both their immediate effects and the learning that follows.

AI USE STATEMENT

Generative AI tools were used to help implement and refine experimental, analysis, and plotting code, provide feedback on experimental design, and assist with the interpretation of results and the development and presentation of analytical arguments. The authors made the final methodological decisions and determined the scientific conclusions.

The authors prepared the initial manuscript. Generative AI tools subsequently assisted with revision, restructuring, translation, and language editing. They were also used to identify relevant literature and check bibliographic information against source documents. Numerical results are drawn from experiment records; the synthetic controls use the data-generation procedure in Appendix I. The authors take full responsibility for the final content of this work, including its scientific claims, experimental results, references, code, and text.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328, 2021. doi: 10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

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 arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

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. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-0441. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.

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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4133–4145, 2023. doi: 10.18653/v1/2023.emnlp-main.252. URL https://aclanthology.org/2023.emnlp-main.252/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783. URL https://arxiv.org/abs/2407.21783.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. In *Proceedings of the 41st International Conference on Machine Learning*, volume

235 of *Proceedings of Machine Learning Research*, pp. 17783–17806, 2024. URL https://proceedings.mlr.press/v235/hayou24a.html.

Haonan He, Peng Ye, Yuchen Ren, Yuan Yuan, Luyang Zhou, Shucun Ju, and Lei Chen. GoRA: Gradient-driven adaptive low rank adaptation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3805. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a5e4907a40c0dcb8433a35c714ba9d79-Abstract-Conference.html.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2111.09543.

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

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. MoRA: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*, 2024. doi: 10.48550/arXiv.2405.12130. URL https://arxiv.org/abs/2405.12130.

Vishnuprasadh Kumaravelu, Sunil Gupta, and P. K. Srijith. Post-optimization adaptive rank allocation for LoRA. *arXiv preprint arXiv:2604.27796*, 2026. doi: 10.48550/arXiv.2604.27796. URL https://arxiv.org/abs/2604.27796.

Jiaqi Liu, Haidong Kang, Qihui Zhao, Guo Yu, and Jingchao Wang. RSRA: Training-free probing of representation sensitivity for efficient LoRA rank allocation. *arXiv preprint arXiv:2607.09757*, 2026. doi: 10.48550/arXiv.2607.09757. URL https://arxiv.org/abs/2607.09757v2. Version 2.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121, 2024. URL https://proceedings.mlr.press/v235/liu24bn.html.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. DoRA: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11662–11675, 2024. doi: 10.18653/v1/2024.acl-long.626. URL https://aclanthology.org/2024.acl-long.626/.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Rifat Rafiuddin and Rafae Abdullah. Context-conditioned masked LoRA: Dynamic rank routing for compute-efficient parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26670–26689, 2026. doi: 10.18653/v1/2026.findings-acl.1329. URL https://aclanthology.org/2026.findings-acl.1329/.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference*, pp. 606–610, 2007. doi: 10.5281/zenodo.40328. URL https://zenodo.org/records/40328.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQA: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 4463–4473, 2019. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Yash Ganpat Sawant. When gradient importance lies: Adaptive LoRA rank allocation fails under GRPO. *arXiv preprint arXiv:2605.07366*, 2026. doi: 10.48550/arXiv.2605.07366. URL https://arxiv.org/abs/2605.07366v2.

Hao Nan Sheng, Zhi-Yong Wang, Hing Cheung So, and Mingrui Yang. AROMA: Autonomous rank-one matrix adaptation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3443–3459, 2025. doi: 10.18653/v1/2025.emnlp-main.170. URL https://aclanthology.org/2025.emnlp-main.170/.

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 *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, 2023. doi: 10.18653/v1/2023.eacl-main.239. URL https://aclanthology.org/2023.eacl-main.239/.

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. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

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*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Feiyu Zhang, Liangzhi Li, Junhao Chen, Zhouqiang Jiang, Bowen Wang, and Yiming Qian. IncreLoRA: Incremental parameter allocation method for parameter-efficient fine-tuning. *arXiv preprint arXiv:2308.12043*, 2023a. doi: 10.48550/arXiv.2308.12043. URL https://arxiv.org/abs/2308.12043.

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

Colin Zhao, Qinghua Yao, Xinyuan Song, and Wei Zhu. ShapLoRA: Allocation of low-rank adaption on large language models via shapley value inspired importance estimation. In *Conference on Parsimony and Learning*, volume 328 of *Proceedings of Machine Learning Research*, pp. 242–264, 2026. URL https://proceedings.mlr.press/v328/zhao26a.html.

Table 7: Allocation timing and the corresponding resource question. Examples describe method families, not additional experimental baselines.

Timing	Examples / information	Resource to account for
Static	Uniform rank; no task probe	Active and stored factors
Before training	GoRA; gradient snapshot	Probe plus fixed allocation
During training	AdaLoRA, SoRA, IncreLoRA; evolving scores	Rank trajectory, probes, updates
After training	PARA; trained singular spectrum	Larger training model plus compression
Per input	CCM-LoRA; input-conditioned routing	Expected active rank and routing work

Table 8: Training configuration. R is instantiated factor capacity, not active rank. GLUE task-specific epoch counts are listed in Table 9.

Setting	GLUE main	Commonsense	GSM8K
Maximum sequence length	256	512	512
Batch $\times$ accumulation	32×1	8×2	4×4
Learning rate (LoRA-based)	2×10^{-4}	2×10^{-4}	2×10^{-4}
Learning rate (AdaLoRA)	10^{-3}	3×10^{-4}	3×10^{-4}
Epochs	30 small / 3 large	1	3
Warmup fraction	0.10	0.03	0.03
Weight decay	0.01	0	0
LoRA α / dropout	16 / 0.05	32 / 0.05	32 / 0.05
Target total rank / R	432 / 12	3,584 / 32	3,584 / 32

A Training and evaluation details

A.1 Data, repetitions, and model selection

The main GLUE runs use training seeds 1, 2, and 42. The component and commonsense comparisons use 42, 43, and 44; GSM8K uses 42. The synthetic controls use 0, 1, and 2. Tables with dispersion report population standard deviations; the paired intervals use sample standard deviations of within-seed differences. AdaLoRA's displayed 0.00 on RTE and Winogrande corresponds to identical metric values across the three runs.

The main GLUE runs train for 30 epochs on RTE, MRPC, CoLA, and STS-B, and for three epochs on SST-2, QNLI, MNLI, and QQP. All use AdamW with the learning rates, weight decay, and warmup in Table 8. These learning rates are fixed settings for the comparison. The component runs use 30 epochs on RTE, CoLA, and STS-B and ten on MNLI; they instantiate rank 16 and start from reference rank four.

Table 9: Encoder data and primary metrics. Training uses each task's training split; evaluation uses validation, or validation-matched for MNLI. Counts describe the local GLUE files.

Task	Train	Evaluation	Primary metric	Epochs
RTE	2,490	277	Accuracy	30
MRPC	3,668	408	F1	30
CoLA	8,551	1,043	Matthews correlation	30
STS-B	5,749	1,500	Pearson correlation	30
SST-2	67,349	872	Accuracy	3
QNLI	104,743	5,463	Accuracy	3
MNLI	392,702	9,815	Matched accuracy	3
QQP	363,846	40,430	F1	3

Encoder evaluation is scheduled at epoch boundaries. Ordinary LoRA-based methods save the checkpoint with the highest logged selection metric and restore its model and rank state for final evaluation. For these MRPC and QQP runs, selection uses accuracy, while the tables report F1 at the selected checkpoint. AdaLoRA scores are selected using the best logged primary task metric

(F1 for MRPC/QQP) at exactly the target active rank. Masked singular components can become nonzero between mask applications, producing above-budget validation events; the target-rank filter removes those events from the candidate set. All methods share the evaluation schedule, with AdaLoRA selecting from this filtered set. Its saved compressed adapters and the events used for metric selection are identified separately in the resource table.

A.2 Decoder data and scoring

The eight tasks are ARC-Challenge and ARC-Easy (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), Social IQA (Sap et al., 2019), and Winogrande (Sakaguchi et al., 2020). Training selects the first 30,000 records from the local merged instruction file before minibatch shuffling. This is a fixed mixture, not per-task-balanced sampling. Evaluation takes the first 300 records in each formatted validation file, or all records if fewer are available: ARC-Challenge has 299 and the other tasks have 300, giving 2,399 in total.

The instruction wrapper is identical across methods:

```
Below is an instruction that describes a task.  Write a
response that appropriately completes the request.
### Instruction:
{instruction and optional input}
### Response:
```

The target format is "the correct answer is {label}". For seven tasks, evaluation uses greedy generation, a maximum of 32 new tokens, and extraction of the first task-label token. The loss masks prompt tokens and trains on response tokens. Decoder training uses NF4 double quantization, bfloat16 computation, and gradient checkpointing.

Winogrande's primary result uses the first 300 formatted validation examples and the conditional log-likelihood of two complete continuations: "the correct answer is option1" and "the correct answer is option2". For candidate c, the score is

$$s(c \mid x) = \sum_{j=1}^{|c|} \log p(c_j \mid x, c_{<j}). \tag{8}$$

Prompt tokens are excluded. No length normalization is applied; the tokenizer check requires equal-length candidates differing only in their final option token. Ties choose option 1. The protocol was fixed before evaluating the complete four-method, three-run checkpoint matrix. It measures forced-choice likelihood rather than the ability to emit a parseable answer string.

For each run, the eight-task score replaces only the original Winogrande contribution:

$$A_{\text{suite}} = \frac{2399A_{\text{generation}} - 300A_{\text{Wino,gen}} + 300A_{\text{Wino,LL}}}{2399}. \tag{9}$$

The generation parser produced method-dependent format failures: mean Winogrande scores were 23.89, 29.11, 7.00, and 53.44 for uniform, one-shot, online, and AdaLoRA. Conditional-likelihood scoring removes this dependence on answer formatting while preserving the other seven task scores. All methods use the same examples, wrapper, candidate strings, and tie rule.

Commonsense checkpoints are evaluated at the end of the one-epoch run. GSM8K trains on its 7,473 training examples for three epochs and evaluates the first 500 test examples. Generation is greedy with at most 256 new tokens; answers are parsed after "####", falling back to the last number, with numeric equality tolerance 10^{-4}. We select and report the highest scheduled accuracy on these 500 test examples; AdaLoRA candidates are restricted to target-rank events.

B Exact allocation and factor handling

GLUE and commonsense evaluate 128 calibration examples per event. GSM8K's batch limit evaluates the first 64 examples of its 128-example panel. GLUE resamples the panel at each event; the decoder keeps it fixed.

Integer allocation. For m modules, set $E = \min\{\max(B - mr_{\min}, 0), m(R - r_{\min})\}$. Replace non-finite EMA values with zero, then min–max normalize to S_i. If the EMA range is at most 10^{-12}, the normalized scores are zero. Let $a_i = ES_i / \sum_j S_j$ when the score sum exceeds 10^{-12}, and $a_i = E/m$ otherwise. Initialize $e_i = \min\{\lfloor a_i \rfloor, R - r_{\min}\}$. Sort modules by decreasing $(a_i - \lfloor a_i \rfloor) + 10^{-9} S_i$ with a stable sort. Cycle through that order, assigning one unit to each uncapped module, until all remaining units have been distributed. Set $r_i = r_{\min} + e_i$. This capped floor-and-remainder rule allows all integer ranks within the configured range; final ties follow injection order.

Table 10: Online event procedure used by the real-task runs.

1	Compute the ordinary training loss and its factor gradients for the pending optimizer update.
2	If t is an eligible measurement step, evaluate the calibration loss in evaluation mode. Use a deterministically resampled panel on GLUE and the fixed panel on the decoder.
3	Compute $G_{i,k}$ and its EMA using Equations 4–5; the first event has zero loss difference.
4	Apply capped integer allocation to the EMA. Change active slices without reallocating tensors or resetting optimizer state. Restore training mode.
5	Apply the pending optimizer update and scheduler step. Stop rank changes for the final 15% of training.

Factor tensors have shapes $A_i \in \mathbb{R}^{R \times d_{\text{in},i}}$ and $B_i \in \mathbb{R}^{d_{\text{out},i} \times R}$. At construction, A receives Kaiming-uniform initialization and B is zero. Shrinking rank deactivates trailing columns/rows; it does not delete them or select components by singular value. Expanding rank reactivates the stored entries. Values and Adam moments are retained, including entries that were active earlier. Inactive entries receive no forward contribution, but retained momentum and weight decay can affect their stored values. The forward scaling is recomputed as α/r whenever rank changes, so a change in rank can alter the contribution of already-active components immediately.

In the GLUE loop, allocation occurs after the current backward pass and before its optimizer step. Thus the pending gradient was computed under the preceding rank. The decoder applies the same event logic at optimizer-step boundaries with gradient accumulation. For the main GLUE trajectory, rank eight is used through step 199 and the first allocation at step 200 is uniform rank six because all first-event EMA values are tied. Low-budget online runs instead initialize at their target rank.

Direct probes in the synthetic controls. For $r_i > 0$, the probe score is $\widehat{G}_i = [\ell(\theta) - \ell(\theta - \eta g_i)]/r_i$, where only the current module's parameters are updated, using its training gradient. Losses are measured on a calibration loader in evaluation mode. Parameters are saved, perturbed, evaluated, and restored; optimizer state is not advanced. An event uses one baseline evaluation and one per probed module. Its EMA and integer allocation follow the same rules as above. Zero-rank heads skip the perturbation, receive score zero, and decay their previous EMA by 0.9. The phase-shift control retains a rank-one floor to keep every head available for probing.

C GoRA component definitions

The allocation-enabled component arms use mean sensitivity $u_i = \text{mean}\,|W_i \odot \bar{g}_i|$ rather than the sum used by the main one-shot baseline. A shared evaluation-mode gradient snapshot averages four unshuffled batches. With reference rank $r_0 = 4$, define the dimension-weighted input budget $P_0 = r_0 \sum_i c_i$. The implementation assigns

$$\widetilde{r}_i = \text{clip}_{[1,16]} \left(\left\lfloor \frac{\text{round}(P_0 u_i / \sum_j u_j)}{c_i} \right\rfloor \right). \tag{10}$$

Rounding uses the language runtime's round-to-even rule. Residual budget is not redistributed after clipping, which explains the realized total-rank range 208–271. The fifth, sum-sensitivity one-shot arm uses the full rank-288 floor-and-remainder budget and ordinary initialization.

For gradient-compression initialization, we use the authors' released GoRA implementation (He et al., 2025). Let A be Kaiming-uniform and G the averaged base-weight gradient. The initialization is

$$B = \frac{\gamma}{\alpha} G A^\top (A A^\top + 10^{-8} I)^\dagger, \qquad \gamma = \frac{0.05}{\sqrt{r/d_{\text{in}}}} \sqrt{r}, \tag{11}$$

with stabilized forward scaling $\alpha/\sqrt{r}$. The allocation-only arm uses zero-initialized B and α/r; initialization-only applies the compression/scaling package at uniform rank four. The combined arm applies both. The positive initialization sign follows the released implementation. We apply these implementation formulas to DeBERTa using its projection geometry, a four-batch gradient window, a rank-four reference, and ranks from one to 16. This places the components in a shared encoder setting.

D COMPLETE PAIRED COMPARISONS

For matched seed s, let $d_s = 100(a_{M,s} - a_{U,s})$, where $a_{M,s}$ and $a_{U,s}$ are the task scores for method M and uniform allocation. The reported interval is

$$\bar{d} \pm t_{0.975,\,2}\frac{s_d}{\sqrt{3}}, \qquad s_d^2 = \frac{1}{2}\sum_{s=1}^{3}(d_s - \bar{d})^2. \tag{12}$$

The unit of analysis is the training-seed pair. We report pointwise 95% paired-t intervals, assuming independent seed pairs and approximately normal paired effects. All deltas use unrounded recorded scores. The delta columns follow seeds 1/2/42 for GLUE and 42/43/44 for the other suites.

Table 11: All method-minus-uniform paired effects, in percentage points.

Suite	Task	Method	Three paired deltas	Mean	95% interval
GLUE	RTE	One-shot	+1.08, +0.36, -2.17	-0.24	[-4.48, +4.00]
GLUE	RTE	Online	-0.36, +0.00, -0.36	-0.24	[-0.76, +0.28]
GLUE	RTE	AdaLoRA	+0.36, +1.08, -1.08	+0.12	[-2.62, +2.86]
GLUE	MRPC	One-shot	+0.17, -0.41, +0.23	-0.00	[-0.89, +0.88]
GLUE	MRPC	Online	+0.40, -0.05, -0.09	+0.09	[-0.60, +0.77]
GLUE	MRPC	AdaLoRA	-0.40, -0.89, -0.45	-0.58	[-1.24, +0.09]
GLUE	CoLA	One-shot	-1.29, -0.63, -1.89	-1.27	[-2.83, +0.30]
GLUE	CoLA	Online	-0.87, -1.07, -1.35	-1.10	[-1.70, -0.50]
GLUE	CoLA	AdaLoRA	-0.89, +0.16, -2.21	-0.98	[-3.94, +1.98]
GLUE	STS-B	One-shot	-0.33, -0.04, +0.77	+0.13	[-1.28, +1.55]
GLUE	STS-B	Online	+0.02, -0.11, +0.52	+0.14	[-0.68, +0.97]
GLUE	STS-B	AdaLoRA	+0.06, +0.26, +1.51	+0.61	[-1.35, +2.57]
GLUE	SST-2	One-shot	-0.23, +0.00, +0.57	+0.11	[-0.91, +1.14]
GLUE	SST-2	Online	-0.23, -0.69, +0.34	-0.19	[-1.48, +1.09]
GLUE	SST-2	AdaLoRA	+0.00, -0.34, +0.00	-0.11	[-0.61, +0.38]
GLUE	QNLI	One-shot	+0.24, +0.13, -0.05	+0.10	[-0.26, +0.47]
GLUE	QNLI	Online	-0.04, -0.15, +0.05	-0.04	[-0.29, +0.21]
GLUE	QNLI	AdaLoRA	+0.02, +0.11, +0.35	+0.16	[-0.26, +0.58]
GLUE	MNLI	One-shot	-0.05, +0.18, +0.11	+0.08	[-0.22, +0.38]
GLUE	MNLI	Online	+0.16, +0.20, +0.10	+0.16	[+0.03, +0.28]
GLUE	MNLI	AdaLoRA	+0.35, +0.38, +0.18	+0.30	[+0.04, +0.56]
GLUE	QQP	One-shot	+0.12, -0.13, -0.00	-0.00	[-0.32, +0.31]
GLUE	QQP	Online	-0.07, -0.18, -0.17	-0.14	[-0.30, +0.02]
GLUE	QQP	AdaLoRA	-0.14, -0.09, +0.17	-0.02	[-0.43, +0.40]
Components	RTE	One-shot	-0.36, +0.36, -2.17	-0.72	[-3.96, +2.51]
Components	RTE	Allocation	-0.72, +0.72, -1.44	-0.48	[-3.22, +2.26]
Components	RTE	Initialization	-1.08, -1.44, -1.44	-1.32	[-1.84, -0.81]
Components	RTE	Combined	-1.08, -2.17, -2.17	-1.81	[-3.36, -0.25]
Components	CoLA	One-shot	-1.37, +3.09, -1.20	+0.17	[-6.11, +6.46]
Components	CoLA	Allocation	-1.52, -0.89, -0.64	-1.02	[-2.14, +0.11]
Components	CoLA	Initialization	-1.12, +0.33, -0.17	-0.32	[-2.15, +1.51]
Components	CoLA	Combined	-1.41, -0.18, +0.45	-0.38	[-2.73, +1.96]
Components	STS-B	One-shot	+0.53, -0.11, -0.48	-0.02	[-1.29, +1.25]
Components	STS-B	Allocation	+0.35, -0.51, -0.58	-0.25	[-1.54, +1.04]
Components	STS-B	Initialization	+0.44, -0.25, -0.39	-0.07	[-1.16, +1.03]
Components	STS-B	Combined	+0.43, -0.05, -0.51	-0.04	[-1.21, +1.12]
Components	MNLI	One-shot	+0.06, +0.00, -0.01	+0.02	[-0.08, +0.11]
Components	MNLI	Allocation	+0.23, +0.05, -0.04	+0.08	[-0.27, +0.43]
Components	MNLI	Initialization	+0.08, -0.11, -0.35	-0.13	[-0.66, +0.41]

Suite	Task	Method	Three paired deltas	Mean	95% interval
Components	MNLI	Combined	+0.13, +0.08, -0.19	+0.01	[-0.43, +0.44]
Decoder	Eight-task mean	One-shot	-0.58, +0.58, -0.21	-0.07	[-1.55, +1.41]
Decoder	Eight-task mean	Online	-0.79, +0.63, -0.67	-0.28	[-2.23, +1.67]
Decoder	Eight-task mean	AdaLoRA	-1.88, -2.67, -3.29	-2.61	[-4.38, -0.85]
Decoder	Winogrande	One-shot	+2.33, -0.67, -0.33	+0.44	[-3.64, +4.53]
Decoder	Winogrande	Online	+0.67, +0.33, -0.67	+0.11	[-1.61, +1.83]
Decoder	Winogrande	AdaLoRA	-3.00, -6.33, -5.00	-4.78	[-8.95, -0.61]

E PARAMETER AND TRAJECTORY ACCOUNTING

For DeBERTa, there are 48 attention matrices of size 768×768 and 24 feed-forward matrices of size 3072×768 or 768×3072. Therefore $\sum_i c_i = 165,888$. A compact rank-six adapter has 995,328 parameters, while rank-12 storage has 1,990,656. The binary-task classifier contributes 1,538 parameters, the regression head 769, and the three-class MNLI head 2,307. The component experiment's rank-16 storage has 2,654,208 LoRA parameters.

Llama has 32 layers. Each contributes query/output matrices with per-rank cost 8,192; key/value matrices with cost 5,120; and gate/up/down matrices with cost 18,432. Thus $\sum_i c_i = 2,621,440$, giving 41,943,040 active parameters at uniform rank 16 and 83,886,080 stored parameters at rank 32. AdaLoRA adds 7,168 trainable singular-value scalars to rank-32 storage.

Table 12 reports run means. For ordinary LoRA, active parameter counts come from the saved selected rank state. For AdaLoRA, they are reconstructed from the saved compression mask and module dimensions, including active singular-value scalars. These counts describe the saved compressed adapter; validation metrics use the selected earlier target-rank event. The accompanying records also retain raw uncompressed AdaLoRA rank totals, including events above the target.

Table 12: GLUE resource records. Rank denotes total rank. Rank and active parameters refer to the selected ordinary adapter or saved AdaLoRA compression mask. Instantiated counts include the task head. Time is the recorded training timer, in seconds.

Task	Method	Rank	Active adapter	Instantiated	Rank-steps	Seconds
RTE	Uniform	432	995,328	1,992,194	1,010,880	637
RTE	One-shot	432	1,153,536	1,992,194	1,010,880	643
RTE	Online	432	1,158,144	1,992,194	1,039,536	648
RTE	AdaLoRA	432	1,005,744	1,993,058	1,812,065	788
MRPC	Uniform	432	995,328	1,992,194	1,490,400	615
MRPC	One-shot	432	1,094,400	1,992,194	1,490,400	598
MRPC	Online	432	1,162,752	1,992,194	1,519,056	605
MRPC	AdaLoRA	432	1,071,024	1,993,058	2,672,103	744
CoLA	Uniform	432	995,328	1,992,194	3,473,280	1,106
CoLA	One-shot	432	1,109,760	1,992,194	3,473,280	1,075
CoLA	Online	432	1,112,832	1,992,194	3,501,936	1,110
CoLA	AdaLoRA	432	1,019,568	1,993,058	6,228,585	1,631
STS-B	Uniform	432	995,328	1,991,425	2,332,800	891
STS-B	One-shot	432	1,087,488	1,991,425	2,332,800	911
STS-B	Online	432	1,186,560	1,991,425	2,361,456	890
STS-B	AdaLoRA	432	1,117,872	1,992,289	4,183,199	1,170
SST-2	Uniform	432	995,328	1,992,194	2,728,080	670
SST-2	One-shot	432	1,159,680	1,992,194	2,728,080	684
SST-2	Online	432	1,031,424	1,992,194	2,756,736	705
SST-2	AdaLoRA	432	1,021,104	1,993,058	4,892,182	1,108
QNLI	Uniform	432	995,328	1,992,194	4,243,104	1,154
QNLI	One-shot	432	1,115,136	1,992,194	4,243,104	1,161
QNLI	Online	432	1,071,360	1,992,194	4,271,760	1,190
QNLI	AdaLoRA	432	998,064	1,993,058	7,609,716	1,884
MNLI	Uniform	432	995,328	1,992,963	15,904,512	4,223
MNLI	One-shot	432	1,137,408	1,992,963	15,904,512	4,180

Task	Method	Rank	Active adapter	Instantiated	Rank-steps	Seconds
MNLI	Online	432	1,036,800	1,992,963	15,933,168	4,255
MNLI	AdaLoRA	432	908,976	1,993,827	28,526,241	6,923
QQP	Uniform	432	995,328	1,992,194	14,736,816	3,732
QQP	One-shot	432	1,109,760	1,992,194	14,736,816	3,745
QQP	Online	432	1,007,616	1,992,194	14,765,472	3,766
QQP	AdaLoRA	432	945,072	1,993,058	26,432,143	6,226

Table 13: Decoder resource records, with the same count definitions as Table 12. Active counts exclude the frozen base model; AdaLoRA uses its saved compression mask.

Task	Method	Rank	Active adapter	Instantiated	Rank-steps	Seconds
Commonsense	Uniform	3,584	41,943,040	83,886,080	6,720,000	6,508
Commonsense	One-shot	3,584	48,172,373	83,886,080	6,720,000	6,890
Commonsense	Online	3,584	42,475,861	83,886,080	6,720,000	6,455
Commonsense	AdaLoRA	3,584	41,485,824	83,893,248	12,043,511	7,201
GSM8K	Uniform	3,584	41,943,040	83,886,080	5,021,184	12,493
GSM8K	One-shot	3,584	47,750,144	83,886,080	5,021,184	12,987
GSM8K	Online	3,584	47,961,088	83,886,080	5,021,184	12,455
GSM8K	AdaLoRA	3,584	46,159,360	83,893,248	8,994,994	14,063

Table 14: Realized component-study resource ranges over the four tasks and their runs. All arms instantiate 2,654,208 LoRA factor parameters, plus the task head.

Method	Total rank range	Active LoRA parameter range
Uniform	288	663,552
One-shot	288	760,320–813,312
Allocation	208–271	453,120–533,760
Initialization	288	663,552
Combined	208–271	453,120–533,760

Training timers include in-loop online calibration and exclude initial model loading. Inclusion of preliminary scoring follows each runner's timer boundary. The 60 component runs consumed 68.19 GPU-hours of recorded execution time, including 67.79 hours within the training timers. Timing summaries use elapsed durations from the recorded execution configurations. The intervention ran on an RTX 3090. Effective-rank utilization, defined through spectral entropy (Roy & Vetterli, 2007), provides a separate description of how broadly singular-value mass is distributed.

F LOWER-BUDGET RESULTS

At total rank 144, the feasible set contains multiple non-uniform allocations. At total rank 72 with a rank-one floor, it contains only the all-ones vector. These runs initialize and remain at rank one in every module. The online path still evaluates calibration loss and advances its measurement logic, so the rank-72 rows in Table 15 compare execution paths with allocation held fixed.

Table 15: Lower-budget GLUE comparisons. Metrics are multiplied by 100 (mean $\pm$ population standard deviation over three runs). Total rank 72 is a degenerate-allocation boundary.

Total rank	Task	Uniform	Online
72	RTE	86.88±1.04	86.52±0.17
72	MRPC	93.58±0.31	93.42±0.33
72	CoLA	69.04±1.17	69.18±0.69
144	RTE	88.45±0.78	87.85±0.34
144	MRPC	93.42±0.43	93.37±0.85
144	CoLA	69.80±0.48	69.46±0.11
144	MNLI	90.24±0.14	90.10±0.02

G PER-RUN SIGNAL DIAGNOSTICS

Table 16 reports event counts, valid adjacent-event correlations, measurement range, and each diagnostic before cross-run averaging. The first event has an all-zero instantaneous score and is omitted from undefined correlations. Entropy is omitted when positive score mass is zero; CV is omitted when the mean is zero. Constant vectors produce no Spearman coefficient. Ties use average ranks. The commonsense seed-43 trace contains repeated writes of identical event/module fields; retaining the final identical write yields one observation per event/module, as for the other runs.

Table 16: Online score diagnostics by task and training seed. The Pairs column counts valid correlations between consecutive measurements, taken every 100 updates. Adjacent is the signed-score correlation; Next score correlates the current EMA with the next instantaneous score.

Task	Seed	Events	Pairs	Step range	H	CV	Adjacent	Next score
RTE	1	18	16	200–1900	0.917	0.858	-0.054	0.204
RTE	2	18	16	200–1900	0.890	1.016	-0.092	0.175
RTE	42	18	16	200–1900	0.895	1.061	0.098	0.136
MRPC	1	28	26	200–2900	0.897	1.186	-0.325	-0.019
MRPC	2	28	26	200–2900	0.872	1.153	-0.179	0.087
MRPC	42	28	26	200–2900	0.818	1.528	-0.281	0.057
CoLA	1	67	65	200–6800	0.857	1.231	-0.178	-0.026
CoLA	2	67	65	200–6800	0.854	1.229	-0.152	0.014
CoLA	42	67	65	200–6800	0.805	1.681	-0.370	0.012
STS-B	1	44	42	200–4500	0.886	1.038	-0.051	0.059
STS-B	2	44	42	200–4500	0.895	1.002	-0.210	-0.004
STS-B	42	44	42	200–4500	0.862	1.164	-0.259	0.088
SST-2	1	52	50	200–5300	0.765	1.816	-0.280	-0.035
SST-2	2	52	50	200–5300	0.773	1.691	-0.261	-0.070
SST-2	42	52	50	200–5300	0.807	1.543	-0.067	-0.025
QNLI	1	82	80	200–8300	0.830	1.494	-0.264	-0.077
QNLI	2	82	80	200–8300	0.803	1.830	-0.246	0.027
QNLI	42	82	80	200–8300	0.831	1.513	-0.262	-0.033
MNLI	1	311	309	200–31200	0.833	1.628	-0.236	-0.069
MNLI	2	311	309	200–31200	0.839	1.600	-0.151	-0.018
MNLI	42	311	309	200–31200	0.835	1.553	-0.235	-0.049
QQP	1	288	286	200–28900	0.836	1.575	-0.223	-0.033
QQP	2	288	286	200–28900	0.839	1.578	-0.304	-0.030
QQP	42	288	286	200–28900	0.846	1.444	-0.249	-0.004
Commonsense	42	14	12	200–1500	0.657	3.031	-0.006	-0.182
Commonsense	43	14	12	200–1500	0.738	2.665	-0.160	0.085
Commonsense	44	14	12	200–1500	0.780	1.968	-0.298	-0.084

H INTERVENTION DETAILS

The RTE intervention branches at training step 1,200 from the stored model state with all 72 modules fixed at rank six. Selection ranks EMA-score gaps within projection-type and parameter-cost classes and chooses three disjoint pairs before branch execution; all selected pairs are feed-forward input projections. Added components retain stored A rows, zero B columns, and clear Adam moments; ordinary online expansion reactivates stored slices.

Deterministic PyTorch, cuDNN, and cuBLAS settings make the bracketing unchanged-allocation replays agree exactly in initial loss, final loss, and restored logits. For pair 1, aligned and anti-aligned losses start at 0.677425 and 0.700164 and end at 0.777465 and 0.795149. Both rise, but the aligned branch retains a smaller advantage after training.

I SYNTHETIC CONSTRUCTION AND ABSOLUTE ERRORS

For head i, the teacher is $y_i = \mathrm{GELU}((W_i + \Delta_i)x)$ with $x \sim \mathcal{N}(0, I_{128})$. Frozen base matrices are shared with the student. On planted heads, $\Delta_i = UV/\sqrt{128d}$, where $U \in \mathbb{R}^{128\times d}$ and $V \in \mathbb{R}^{d\times 128}$

have independent standard-normal entries; elsewhere $\Delta_i = 0$. There is no additive label noise. Loss is mean squared error over examples, heads, and output coordinates. There are 4,096 training and 1,024 evaluation inputs; the per-run initialization and datasets are shared across methods.

The stable control uses Adam for 800 steps (batch size 128, learning rate 0.01, maximum rank eight, $\alpha = 8$, no dropout). Direct probes run every 50 steps on two calibration batches, with EMA coefficient 0.9 and minimum rank zero. One-shot allocation scores eight initial batches. The oracle uses known planted heads at the planted rank, up to total rank 16, and trains for the same horizon. Table 17 reports the residuals.

Table 17: Stable-demand MSE (mean $\pm$ population standard deviation over three runs); d is planted rank on each of two heads.

d	Uniform	One-shot	Online probe	Oracle reference
1	$(4.23 \pm 3.56) \times 10^{-7}$	$(1.38 \pm 0.408) \times 10^{-7}$	$(7.68 \pm 3.05) \times 10^{-8}$	$(7.50 \pm 4.26) \times 10^{-7}$
2	$(5.27 \pm 5.32) \times 10^{-8}$	$(3.91 \pm 1.49) \times 10^{-8}$	$(3.77 \pm 2.03) \times 10^{-8}$	$(5.27 \pm 5.32) \times 10^{-8}$
3	$(2.96 \pm 0.233) \times 10^{-2}$	$(1.96 \pm 2.09) \times 10^{-8}$	$(4.34 \pm 2.96) \times 10^{-8}$	$(7.21 \pm 9.65) \times 10^{-13}$
4	$(4.63 \pm 0.278) \times 10^{-2}$	$(1.66 \pm 1.08) \times 10^{-8}$	$(5.15 \pm 3.62) \times 10^{-8}$	$(1.02 \pm 0.0150) \times 10^{-14}$
6	$(6.12 \pm 0.246) \times 10^{-2}$	$(4.97 \pm 7.00) \times 10^{-9}$	$(8.79 \pm 8.32) \times 10^{-9}$	$(9.81 \pm 0.0267) \times 10^{-15}$
8	$(6.96 \pm 0.383) \times 10^{-2}$	$(9.46 \pm 0.203) \times 10^{-15}$	$(9.07 \pm 12.8) \times 10^{-12}$	$(9.69 \pm 0.118) \times 10^{-15}$

The phase control uses four heads with planted rank four, a rank-one floor, maximum rank four, and $\alpha = 4$. Adam training lasts 1,600 steps; demand switches at step 800. Probes run every 40 steps on two calibration batches, while one-shot scores only phase one. Uniform integer division gives rank two per head (total eight). One-shot and oracle use total rank ten throughout; online starts at eight and reaches ten at step 40.

Table 18: Phase-two absolute MSE (mean $\pm$ population standard deviation over three runs). Individual values are shown in Figure 1(b).

Method	Phase-two total rank	MSE
Uniform	8	0.093801 ± 0.004089
One-shot	10	0.154644 ± 0.004970
Online probe	10	0.027376 ± 0.012627
Oracle reference	10	$(2.951 \pm 3.307) \times 10^{-7}$