TRAIN WIDE, COMMIT LATE: DELAYED CAPACITY COMMITMENT FOR PARAMETER-EFFICIENT FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning (PEFT) often makes the deployment budget binding before optimization ends. We challenge this coupling and introduce Delayed Capacity Commitment (DCC), a design principle that decouples adaptive compression from final deployment commitment. DCC retains a temporary surplus of candidate capacity after adaptive compression and resolves it only at final commitment, without changing the deployment budget. We instantiate DCC with Adaptive Competition Policy (ACP), which uses global rank-one competition and training-state-driven compression to govern capacity evolution. We define the capacity dividend as the paired performance difference between delayed and immediate commitment under matched initial and deployment capacities. On GSM8K, delayed commitment yields consistently positive dividends across five Qwen3-0.6B seeds at both tested buffer ratios, with gains up to 2.50 percentage points; a smaller-buffer setting also yields a positive result on Llama-3.1-8B. Broader evaluations reveal substantial task-, model-, and budget-dependence. These results identify when capacity becomes binding as a distinct design dimension in low-rank PEFT.

1 INTRODUCTION

Parameter-efficient fine-tuning (PEFT) adapts pretrained models under a small task-specific parameter budget, making increasingly capable models practical to specialize and deploy. Low-rank methods such as LoRA (Hu et al., 2022) achieve this by restricting weight updates to a low-dimensional space, while subsequent methods improve how this limited capacity is parameterized and allocated. At the heart of these approaches lies a fundamental tension: **deployment favors compact capacity, whereas optimization may benefit from greater flexibility**. Many low-rank PEFT methods, however, couple these requirements by allowing the deployment constraint to shape training-time capacity.

Figure 1 illustrates how existing low-rank PEFT methods couple training-time capacity with deployment constraints. In fixed-capacity methods (Figure 1(a)), the deployment budget B_D is imposed from the start of training. This is simple and deployment-aligned, but it also limits the number of simultaneously active rank-one components from the outset, although their learned directions can change during optimization. Adaptive rank-allocation methods (Figure 1(b)) relax this restriction by starting from a larger candidate capacity B_{init} and progressively deciding where capacity should be retained. AdaLoRA, for example, reallocates rank across modules while reducing a global budget toward B_D (Zhang et al., 2023). Yet when this schedule reaches and remains at B_D, the deployment constraint becomes binding even though optimization continues. Thus, fixed and adaptive methods differ in how much capacity is available and where it is allocated, but both can make deployment capacity binding before optimization is complete. This raises a third, largely implicit design question: beyond how much capacity to use and where to allocate it, when should the deployment capacity become binding?

Two limiting capacity trajectories illustrate the choice. Retaining the full initial capacity B_{init} until training ends preserves flexibility but postpones capacity selection almost entirely, whereas compressing to B_D early imposes the deployment capacity limit while optimization is still ongoing.

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.

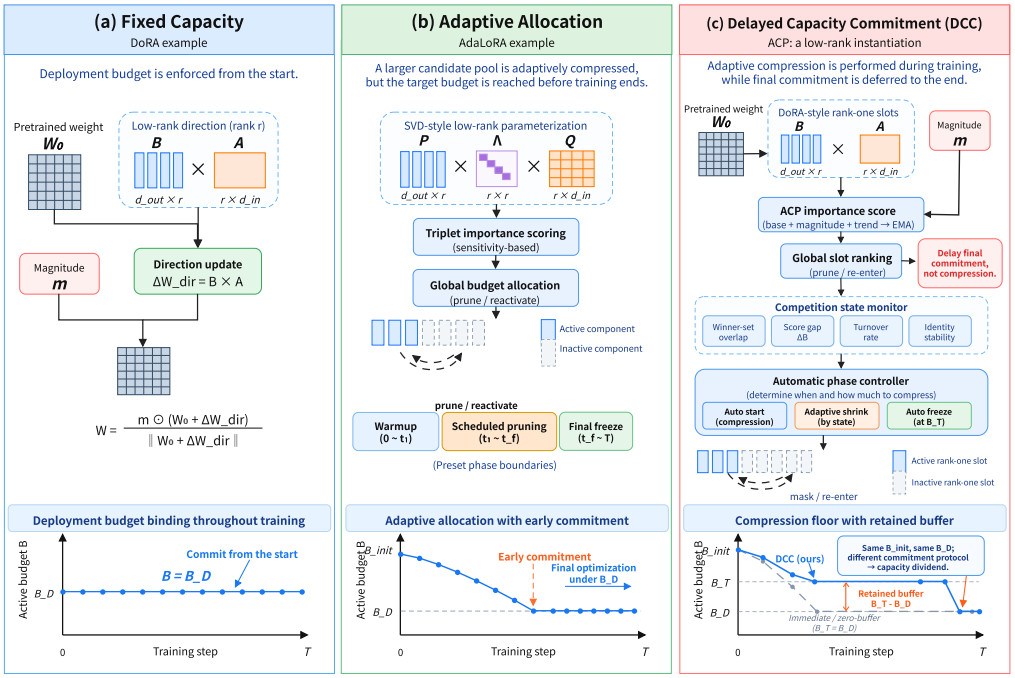

Figure 1: **From capacity allocation to capacity commitment.** **(a)** Fixed capacity binds the deployment budget throughout training. **(b)** Adaptive allocation varies capacity but commits to the deployment budget before training ends. **(c)** DCC delays final commitment while preserving adaptive compression; ACP is our low-rank instantiation.

DCC occupies the intermediate regime between these extremes. As illustrated in Figure 1(c), adaptive compression still removes most candidate capacity during training, but stops at an intermediate floor B_T, where $B_D < B_T < B_{\text{init}}$. The resulting buffer $B_T - B_D$ remains active until final commitment, when the adapter is reduced to B_D. We call this regime Delayed Capacity Commitment (DCC): it delays final commitment, not compression itself.

As illustrated in Figure 1(c), we instantiate DCC with Adaptive Competition Policy (ACP), a low-rank framework that represents directional updates as globally competing rank-one slots. ACP uses the evolving competition state to control adaptive compression, allowing candidate slots to compete and re-enter until the active budget reaches B_T. The surviving slots then continue training, and only final commitment reduces the adapter to B_D.

DCC enables a matched-endpoint comparison of commitment protocols. Delayed and immediate commitment start from the same candidate slot count B_{init} and deploy the same final slot count B_D, but use different compression floors: $B_T > B_D$ and $B_T = B_D$, respectively. We call their paired performance difference the capacity dividend. This contrast measures the protocol-level effect of retaining a temporary buffer under matched endpoint slot budgets; it does not isolate timing from changes in learned weights or final slot allocation.

Across our experiments, delayed commitment yields clear positive capacity dividends in the primary reasoning settings, while broader evaluations reveal substantial dependence on model, task, buffer size, and deployment rank. A complementary intervention that varies when the final structure is fixed further shows that commitment timing can affect final performance, although the effect is not monotonic. Together with additional controls, these results support neither a universal "later is better" rule nor a single mechanism for the observed dividend. Instead, they identify when capacity becomes binding as a distinct design dimension in low-rank PEFT.

Our contributions are threefold. First, we introduce Delayed Capacity Commitment (DCC), a design principle that decouples adaptive compression from final deployment commitment and explicitly exposes commitment timing as a design choice in low-rank PEFT. Second, we instantiate DCC

through Adaptive Competition Policy (ACP), which combines global rank-one competition with training-state-driven compression and retains a temporary capacity buffer before final commitment. Third, we formulate the capacity dividend as a matched-endpoint test of delayed versus immediate commitment and systematically examine its behavior across commitment times, models, tasks, buffer sizes, and deployment ranks.

2 Related Work

PEFT and low-rank adaptation. PEFT adapts pretrained models through small task-specific parameterizations, including adapters, prompts, selected parameters, and low-dimensional updates(Houlsby et al., 2019; Pfeiffer et al., 2021; R"ucklé et al., 2021; Mahabadi et al., 2021; Li & Liang, 2021; Lester et al., 2021; Ben Zaken et al., 2022; Liu et al., 2022; Mao et al., 2022; He et al., 2022; Chen et al., 2023; Aghajanyan et al., 2021). LoRA parameterizes the weight update with low-rank factors, and DoRA separates the magnitude and direction of the adapted weight (Hu et al., 2022; Liu et al., 2024). Later work changes quantization, basis sharing, optimization, or initialization while retaining a low-rank adaptation structure (Dettmers et al., 2023; Kopiczko et al., 2024; Hayou et al., 2024; Wang et al., 2024; Meng et al., 2024; Li et al., 2024; Xu et al., 2024; Nikdan et al., 2024; Yang et al., 2024; Zhang et al., 2024). Our work does not introduce another low-rank parameterization; instead, it asks when the final capacity constraint should become binding during low-rank adaptation.

Adaptive rank allocation. Adaptive low-rank methods relax fixed-rank training by allowing capacity to vary across components or modules. AdaLoRA starts from a larger candidate pool and progressively allocates a shrinking global rank budget according to importance(Zhang et al., 2023). DyLoRA supports multiple deployment ranks from a shared adapter, while SoRA and IGU-LoRA use learned sparsity or importance signals to determine which low-rank components remain active (Valipour et al., 2023; Ding et al., 2023; Cui et al., 2026). These methods primarily address how much capacity to retain and where to allocate it. DCC focuses on a complementary question: when should adaptive compression reach the final deployment budget? Our ACP instantiation therefore changes the stopping floor of adaptive compression while keeping the initial candidate pool and final deployment budget fixed.

Training wide and compressing. Training with greater capacity than is ultimately deployed is well established in model compression and pruning(Li et al., 2020; Frankle & Carbin, 2019; Sanh et al., 2020; Frantar & Alistarh, 2023; Sun et al., 2024), and ReLoRA obtains high-rank training trajectories through repeated low-rank updates (Lialin et al., 2024). DCC differs from simply retaining the full training capacity until post-hoc compression: it delays final commitment, not compression itself. Most candidate capacity is still removed adaptively during training, but compression stops above the deployment budget and the remaining surplus is resolved only at final commitment. This formulation lets us compare delayed and immediate commitment under matched initial and deployment capacities.

3 DELAYED CAPACITY COMMITMENT

We now formalize Delayed Capacity Commitment (DCC) and its low-rank instantiation, Adaptive Competition Policy (ACP). DCC treats the point at which deployment capacity becomes binding as an explicit design variable, complementing the conventional questions of how much capacity to use and where to allocate it. We first formalize this commitment dimension, then instantiate it through ACP, and finally define a matched-endpoint comparison for quantifying the empirical consequence of delayed commitment.

3.1 CAPACITY COMMITMENT AS A DESIGN DIMENSION

Let B_t denote the active capacity at optimization step t, B_D the final deployment budget, and T the final training step. Capacity design involves not only how much capacity is active and where it is allocated, but also when the deployment budget becomes binding on subsequent optimization. We

formalize this third dimension through the commitment time

$$\tau_C = \min\{t : B_{t'} \le B_D,\ \forall t' \in [t, T]\}. \tag{1}$$

Thus, τ_C is the earliest step after which active capacity remains constrained to the deployment budget through the end of training. This definition constrains active capacity; it does not itself require the identities of the active components to remain fixed. Fixed-capacity training makes the deployment budget binding from the outset, corresponding to $\tau_C = 0$. Adaptive compression can postpone this point, but if it reaches and remains at B_D before optimization ends, then $0 < \tau_C < T$.

DCC treats τ_C as an explicit design variable rather than an implicit consequence of the compression schedule. Delaying commitment means shifting τ_C later while keeping the deployment budget unchanged. This separates commitment timing from both capacity amount and capacity allocation.

Importantly, DCC does not prescribe a particular parameterization, allocation rule, or compression mechanism. Instead, it separates capacity reduction from deployment commitment: substantial compression may occur during training without requiring the final deployment budget to become binding at the same time. In the ACP instantiation studied here, adaptive compression stops above B_D, so the deployment budget becomes binding only at final commitment.

3.2 Adaptive compression With Delayed Commitment

Capacity representation. We instantiate DCC using rank-one slots as units of low-rank capacity. For a pretrained weight $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, the masked low-rank update and DoRA weight are

$$\Delta W = \frac{\alpha}{r_{\text{init}}} \sum_{j=1}^{r_{\text{init}}} \chi_j b_j a_j^\top, \qquad W = m \odot \frac{W_0 + \Delta W}{\|W_0 + \Delta W\|_{2,\text{row}}} \tag{2}$$

Here, a_j and b_j define rank-one slot j, $\chi_j \in \{0, 1\}$ is its activity mask, and m is the trainable row magnitude initialized from W_0. The scaling factor α / r_{init} remains fixed as slots are deactivated.

Let $r_{\ell,t}$ denote the number of active slots in module ℓ at step t. We define the total active capacity and final deployment budget as

$$B_t = \sum_{\ell=1}^{N_{\text{module}}} r_{\ell,t}, \qquad B_D = r_{\text{target}} N_{\text{module}}. \tag{3}$$

The budget is global: individual module ranks may differ as long as their total number of active slots satisfies B_t. This global formulation allows ACP to redistribute capacity across modules during training.

Because adapted modules can have different input and output dimensions, matching B_D matches the number of active slots rather than the exact number of scalar parameters. For a compact realization with module-wise ranks r_ℓ, the corresponding parameter count is

$$P_{\text{compact}} = \sum_{\ell} r_\ell \left(d_{\text{in},\ell} + d_{\text{out},\ell}\right) + \sum_{\ell} d_{\text{out},\ell}, \tag{4}$$

where the second term accounts for the DoRA magnitudes. Thus, B_D should be interpreted as a structural rank budget, rather than an exact scalar-parameter or compute budget. Details of adapted modules, rank configurations, and masked storage are provided in Appendix A.

Adaptive competition and compression. Starting from an initial candidate capacity $B_{\text{init}} = r_{\text{init}} N_{\text{module}}$, ACP progressively reduces active capacity through global competition among rank-one slots. Each slot receives an importance score derived from its optimization state, including magnitude- and trend-based evidence. Scores are standardized within modules and smoothed over time before slots are ranked globally. High-scoring slots remain active, while lower-scoring slots are masked; masked slots remain eligible to re-enter as their scores evolve while competition is active. This global competition allows capacity to be redistributed across modules rather than committing early to a fixed module-wise rank allocation.

Rather than following a preset compression schedule, ACP monitors the evolving competition state. In particular, the controller tracks the stability and score separation of the leading B_D candidates,

capturing whether the prospective deployment set is still changing or has begun to stabilize. These signals determine when compression begins and how aggressively the active budget is reduced. Detailed scoring equations, competition-state statistics, controller rules, and hyperparameters are provided in Appendix B.

Retained buffer and final commitment. The defining DCC intervention in ACP is that adaptive compression need not continue to B_D. We introduce a compression floor $B_T = \text{round}\big((1+\rho)B_D\big)$, where $\rho \geq 0$ controls the temporary capacity buffer. For $\rho > 0$, compression stops at $B_T > B_D$, leaving $B_T - B_D$ additional slots active during the remainder of training. Once B_t reaches B_T, the active set is frozen while the parameters of its surviving slots continue to train. At final commitment, the top-B_D surviving slots are retained according to their scores.

The three capacity levels satisfy $B_{\text{init}} > B_T \geq B_D$. The boundary case $\rho = 0$ gives $B_T = B_D$: adaptive compression continues until the deployment budget becomes binding before training ends, and the remaining optimization proceeds under the deployment-sized structure. We refer to this as immediate commitment. For $\rho > 0$, $B_T > B_D$: compression stops above the deployment budget, the retained buffer $B_T - B_D$ remains active, and the adapter is reduced to B_D only at final commitment. This is the delayed-commitment regime studied in our main comparison.

The distinction is therefore not whether compression occurs, but whether adaptive compression is allowed to make the deployment budget binding before training ends. DCC delays final commitment, not compression itself.

3.3 MATCHED-ENDPOINT COMPARISON

To evaluate delayed commitment under controlled endpoints, we compare protocols with matched initial and deployment capacities. Both start from the same candidate pool B_{init} and commit to the same final budget B_D, but differ in the compression floor B_T. The immediate-commitment baseline uses $B_T = B_D$, whereas delayed commitment uses $B_T > B_D$. The comparison therefore changes the commitment protocol while holding the two endpoint capacities fixed.

For seed s, we define the capacity dividend as $U_s(\rho) = A_s(\rho) - A_s(0)$, where $A_s(\rho)$ is the test accuracy after final commitment under buffer ratio ρ, and $A_s(0)$ is the corresponding accuracy of the matched zero-buffer run. A positive $U_s(\rho)$ indicates higher final performance under delayed than immediate commitment for the matched pair.

This matched-endpoint construction clarifies what the capacity dividend does and does not measure. Because the two protocols share B_{init}, it is not an initial-capacity comparison; because they share B_D, it is not a deployment-capacity comparison. Rather, the capacity dividend measures the consequences of retaining additional active capacity between adaptive compression and final commitment under matched endpoint capacities.

Importantly, matching B_D does not imply identical final adapters. Delayed commitment can affect both the parameters learned by surviving slots and which slots are ultimately selected. The capacity dividend should therefore be interpreted as a protocol-level effect under matched endpoint capacities, rather than as a pure timing effect or a comparison between identical final parameterizations.

4 EXPERIMENTS

We empirically investigate three questions about delayed capacity commitment: (1) Does delaying the point at which the deployment budget becomes binding affect final performance under matched endpoint capacities? (2) What do targeted controls reveal about the observed difference? (3) How does the effect vary across models, tasks, buffer sizes, and deployment budgets?

4.1 EXPERIMENTAL SETUP

Models and datasets. We use Qwen3-0.6B for the primary study and Llama-3.1-8B for cross-model evaluation. GSM8K is the primary benchmark, with additional tasks used to examine task dependence. We further vary the buffer ratio and deployment rank to study sensitivity to retained capacity and deployment budget.

Evaluation and reporting. We report test accuracy, using numeric-answer exact match on all 1,319 GSM8K test examples. The main Qwen comparison uses five paired seeds, while diagnostic controls generally use three. We report means, sample standard deviations, and seed-wise paired differences. Paired tests are descriptive and restricted to five-seed cohorts; three-seed controls are treated as directional diagnostics and report positive-seed counts without confirmatory significance claims.

Comparison protocols. The primary comparison contrasts immediate ($\rho = 0$) and delayed ($\rho > 0$) commitment under matched initial and deployment capacities. Additional experiments vary commitment time, training exposure, final compression, training width, and the path to the final mask. The $\rho = 0.154$ setting originated from an exploratory implementation error and was subsequently rerun as an explicit controlled setting; it was not selected by theory. Historical main experiments and follow-up controls are kept as separate evaluation cohorts, and cross-snapshot values are not used as controlled contrasts.

Implementation details. Main ACP runs use $r_{\text{init}} = 32$, $r_{\text{target}} = 8$, and $\alpha = 16$. We use AdamW with a learning rate of 2×10^{-4}, an effective batch size of 32, and three epochs with cosine decay. Qwen GSM8K runs use 630 optimizer steps unless otherwise stated. Additional task-specific and optimization settings are provided in Appendix A.

4.2 Main Results: The Capacity Dividend

We first test whether the commitment protocol affects final performance when the initial candidate capacity and final deployment capacity are held fixed. Within each model, the immediate- and delayed-commitment variants share B_{init} and B_D, but differ in whether adaptive compression stops above B_D before final commitment. We quantify the resulting paired performance difference by the capacity dividend $U_s(\rho)$.

Table 1 reports the matched-endpoint comparisons. On Qwen3-0.6B GSM8K, retaining a buffer yields $U(0.154) = +1.60 \pm 0.42$ and $U(0.3) = +2.50 \pm 0.93$ percentage points. Both contrasts are positive across all five paired seeds. These differences occur under the same $6272 \rightarrow 1568$ initial-to-deployment slot budgets, without increasing either endpoint slot count.

On Llama-3.1-8B, the smaller buffer also yields a positive dividend, $U(0.154) = +1.95 \pm 0.12$ points, with positive differences across all three seeds under the same $7168 \rightarrow 1792$ endpoint capacities. The larger buffer does not replicate this result, yielding $U(0.3) = 0.00 \pm 2.25$. Thus, a positive capacity dividend is not confined to Qwen3-0.6B, but neither is it guaranteed by retaining more temporary capacity.

Together, these matched-endpoint comparisons show that final performance can depend on the commitment protocol even when the initial and deployment capacities are held fixed.

Table 1: Capacity dividend under matched endpoint capacities.

Model	ρ (B_T)	n	$A(0)$	$A(\rho)$	$U(\rho)$	Pos. seeds
Qwen3-0.6B	0.154 (1809)	5	38.44 ± 0.28	40.04 ± 0.54	$+1.60 \pm 0.42$	5/5
	0.3 (2038)	5	38.44 ± 0.28	40.94 ± 1.05	$+2.50 \pm 0.93$	5/5
Llama-3.1-8B	0.154 (2068)	3	57.75 ± 1.41	59.69 ± 1.51	$+1.95 \pm 0.12$	3/3
	0.3 (2330)	3	57.75 ± 1.41	57.75 ± 2.67	0.00 ± 2.25	1/3

4.3 VARYING THE COMMITMENT TIME

We next directly vary the commitment time τ_C to test whether the point at which the deployment budget becomes binding affects the final outcome. We set $\tau_C/T \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, corresponding to commitment at 20%, 40%, 60%, 80%, or 100% of the 630-step training trajectory. All runs continue training to step 630, so earlier commitment changes how long optimization proceeds under the deployment-sized structure rather than shortening the overall training duration.

Figure 2 shows the effect of commitment timing on final accuracy; Appendix Table 16 reports the corresponding per-seed values. Committing at 20% gives the lowest performance, whereas delaying commitment to the end of training increases accuracy from 35.94% to 38.84%, a paired improve-

ment of 2.91 ± 0.37 percentage points across three seeds. The improvement is positive for all three paired seeds. However, the intermediate commitment points are not monotonic; in particular, committing at 80% does not outperform committing at 60%. The sweep therefore does not support a general rule that later commitment is always better. Instead, it shows that making the deployment budget binding at different stages of training can lead to meaningfully different final outcomes.

This intervention should not be interpreted as isolating a pure timing effect. Changing τ_C also changes the subsequent optimization trajectory and can alter the final selected slots. Nevertheless, the result provides direct evidence that the point at which deployment capacity becomes binding is a consequential training-design choice.

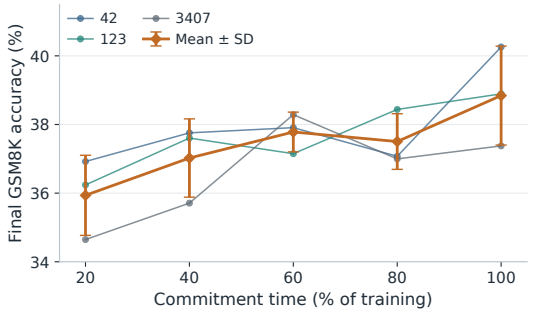

Figure 2: Commitment timing on Qwen3-0.6B GSM8K.

4.4 Diagnosing the Capacity Dividend

We use targeted controls to examine active-slot exposure, final slot identities, terminal compression, and training width. These comparisons use separate cohorts and match different parts of the training and deployment protocol.

Active-slot exposure. Table 2(a) tests whether additional strict training recovers the buffered result. We extend zero-buffer ACP to approximately match the integrated active-slot count of buffered ACP. The extended strict arm remains below the buffered arm on all three seeds ($+3.06 \pm 2.19$ pp), indicating that matched active-slot exposure does not recover the buffered result. This control matches active-slot-time rather than FLOPs; details are in Appendix D. (Appendix Table 14).

Fixed-mask replay. Table 2(b) examines whether the selected slot identities reproduce the buffered outcome in three fresh pairs. We replay each buffered run's final mask from its corresponding initialization. Buffer-minus-replay differences are $+5.23, -0.08, +0.15$ pp, showing strong seed dependence. Thus, final slot identity alone does not consistently reproduce the buffered outcome, nor does the replay establish a uniform advantage for the adaptive path. Further initialization and mask checks accompany Appendix Table 20.

Table 2: Exposure and final-mask diagnostics on Qwen3-0.6B GSM8K. Comparisons are within-panel; parentheses in (b) show correct-answer counts.

Arm / contrast	Seed 42	Seed 123	Seed 3407	Mean ± SD
(a) Active-slot exposure: historical control				
Strict, 630 steps	38.74	38.67	38.44	38.62 ± 0.16
Extended strict	38.74	39.50	36.92	38.39 ± 1.32
Buffered ACP ($\rho = 0.3$)	42.23	40.18	41.93	41.45 ± 1.10
Buffered − extended	+3.49	+0.68	+5.00	$+3.06 \pm 2.19$
(b) Final-mask replay: fresh pairs				
Buffered ACP ($\rho = 0.154$)	41.39 (546)	38.29 (505)	37.76 (498)	39.15 ± 1.97
Fixed-mask replay	36.16 (477)	38.36 (506)	37.60 (496)	37.38 ± 1.12
Buffered − replay	+5.23	−0.08	+0.15	$+1.77 \pm 3.00$

Final compression. Table 3 compares terminal deployment operations while holding the source checkpoint fixed within each seed. We compare terminal deployment operations from the same rank-24 source checkpoints. Score selection reaches 40.61 ± 1.27, exceeding matched-rank SVD by 3.16 ± 0.22 points and the uncompressed Full update by 2.73 ± 1.06, with both differences positive across all three seeds. Thus, the terminal conversion rule can materially affect accuracy at fixed source weights, although this control does not by itself explain the strict-versus-buffer dividend.

Table 3: Final compression from the same rank-24 source checkpoints on Qwen3-0.6B GSM8K.

Deployment update	Active slots	Accuracy	Paired difference
Score selection	1568	40.61 ± 1.27	–
SVD, matched module ranks	1568	37.45 ± 1.39	$+3.16 \pm 0.22$
Full update	4704	37.88 ± 0.64	$+2.73 \pm 1.06$

Training width and stepwise selection. Fixed rank-24 training without stepwise selection reaches 40.89 ± 1.80, within the buffered ACP accuracy range (Table 4). A scale-matched follow-up similarly favors rank 24 over rank 9 by $+1.92 \pm 1.51$ points, with positive differences across all three seeds. Thus, wider fixed-rank training can reach comparable performance without online selection, although these comparisons do not establish equivalence with ACP.

Table 4: Training width and selection on Qwen3-0.6B GSM8K. (a) Historical comparison; (b) scale-matched follow-up.

(a) Historical width and selection comparison

Arm	Active-slot trajectory	Peak	Mean active	Online selection	Accuracy
A Strict	$6272 \rightarrow 1568$	6272	4602.6	yes	38.62 ± 0.16
B Buffered ($\rho = 0.154$)	$6272 \rightarrow 1809$	6272	4636.8	yes	40.31 ± 0.48
C Fixed rank 9	1764	1764	1764.0	no	37.88 ± 0.63
D Fixed rank 32	6272	6272	6272.0	no	40.86 ± 3.09
E Fixed rank 24	4704	4704	4704.0	no	40.89 ± 1.80

(b) Scale-matched width comparison

Training rank	α	α/r	Accuracy	Rank 24 − rank 9
Rank 9	4.5	0.5	38.69 ± 0.52	–
Rank 24	12	0.5	40.61 ± 1.27	$+1.92 \pm 1.51$

The controls support distinct observations: the tested strict extension does not recover buffered accuracy; replay differences are heterogeneous; final conversion affects accuracy at fixed source weights; and fixed-width training can reach the buffered accuracy range without online selection in this pilot. They do not identify a unique mechanism for the main dividend.

4.5 Boundary Conditions of the Capacity Dividend

In this section, we examine how the capacity dividend varies with buffer size, task and model, and final deployment budget to characterize its boundary conditions.

Buffer size. We first vary the retained buffer while keeping the initial and deployment capacities fixed. The separate three-seed scan over seven buffer ratios in Appendix Table 10 shows a non-monotonic mean response: the largest mean dividend occurs at $\rho = 0.23$, while every larger tested ratio falls below that mean. Together with the main comparisons in Table 1—where Qwen3-0.6B benefits from both $\rho = 0.154$ and 0.3, whereas Llama-3.1-8B benefits from the smaller buffer but not the larger one—this result shows that retaining more temporary capacity does not monotonically improve performance. The tested scan therefore characterizes the response to buffer size rather than identifying a universal optimum.

Tasks and models. We next evaluate the matched-endpoint comparison beyond the primary Qwen3-0.6B GSM8K setting at $\rho = 0.154$. As summarized in Table 5(a), the dividend remains positive on GSM8K for both Qwen3-0.6B ($+1.60 \pm 0.42$, five seeds) and Llama-3.1-8B ($+1.95 \pm 0.12$, three

seeds). Across additional tasks, however, the results are mixed. SVAMP gives $+2.60 \pm 2.58$ points on Qwen3-0.6B and $+0.40 \pm 2.36$ on Llama-3.1-8B, while BoolQ, PiQA, and MAWPS remain near zero or negative within the tested runs. AQuA-RAT yields a positive five-seed mean of $+0.79 \pm 2.26$, but the estimate decreases from the initial three-seed result after adding two seeds (Appendix Table 23). These results do not support a task-general benefit; rather, the strongest replicated evidence is concentrated in the GSM8K reasoning setting, with broader transfer depending on the task and model.

Deployment budget. Table 5(b) reports the comparisons across final deployment ranks, with initial and deployment capacities matched within each strict–buffer pair. On Qwen3-0.6B GSM8K with $\rho = 0.154$, target rank 4 yields a dividend of $+1.74 \pm 0.77$ points with positive differences on all three seeds, rank 8 yields $+1.60 \pm 0.42$ across five seeds, and rank 16 yields $+0.45 \pm 0.72$ with two of three differences positive. At rank 16, increasing the buffer to $\rho = 0.3$ gives $+0.94 \pm 0.86$, positive on all three seeds. Thus, positive dividends appear at multiple deployment budgets, but their magnitude varies with the target rank and buffer. These pilots do not establish a monotonic rank dependence or a rule for selecting the buffer from the deployment budget.

Overall, the capacity dividend is neither universal nor monotonic. Its magnitude depends on the retained buffer, model, task, and deployment budget. The experiments therefore support delayed commitment as a consequential design choice whose value is setting-dependent, rather than a uniformly superior policy.

Table 5: Scope of the capacity dividend.

(a) Task and model dependence

Task	Model	n	$U(0.154)$	Seed sign	Role
GSM8K	Qwen3-0.6B	5	$+1.60 \pm 0.42$	5/5 positive	main positive
GSM8K	Llama-3.1-8B	3	$+1.95 \pm 0.12$	3/3 positive	cross-model positive
SVAMP	Qwen3-0.6B	5	$+2.60 \pm 2.58$	4/5 non-negative	positive mean
SVAMP	Llama-3.1-8B	5	$+0.40 \pm 2.36$	2/5 positive	mixed seed signs
BoolQ	Qwen3-0.6B	3	-0.02 ± 0.31	1/3 positive	near-zero mean
PiQA	Qwen3-0.6B	3	$+0.04 \pm 1.12$	2/3 positive	near-zero mean
MAWPS	Qwen3-0.6B	3	-0.83 ± 2.29	1/3 positive	inconclusive
AQuA-RAT	Qwen3-0.6B	5	$+0.79 \pm 2.26$	4/5 positive	inconclusive

(b) Deployment-rank dependence on Qwen3-0.6B GSM8K

r_{target}	κ	ρ	n	$U(\rho)$	Positive seeds
4	8	0.154	3	$+1.74 \pm 0.77$	3/3
8	4	0.154	5	$+1.60 \pm 0.42$	5/5
16	2	0.154	3	$+0.45 \pm 0.72$	2/3
16	2	0.3	3	$+0.94 \pm 0.86$	3/3

5 Conclusion

We introduced Delayed Capacity Commitment (DCC), a design principle that treats when deployment capacity becomes binding as an explicit dimension of low-rank PEFT. Our ACP instantiation separates adaptive compression from final commitment, enabling delayed and immediate commitment to be compared under matched initial and deployment capacities. Across the primary GSM8K experiments, delayed commitment yields positive capacity dividends, while direct interventions on commitment time show that when the deployment budget becomes binding can materially affect final performance.

The effect is neither universal nor monotonic: its magnitude varies across models, tasks, buffer sizes, and deployment budgets, and our diagnostic controls do not identify a unique mechanism. Together, these results suggest that capacity design in low-rank PEFT involves not only how much capacity to use and where to allocate it, but also when to commit to the deployment budget. DCC exposes this timing decision as a design variable rather than an implicit consequence of the compression schedule.

REPRODUCIBILITY STATEMENT

The supplement provides scoring and controller equations, cohort definitions, per-seed results, conversion and replay procedures, evaluation versions, and recorded resource use. The accompanying experiment records identify each comparison's configuration and evaluation snapshot. Training and evaluation code, configurations, and run records will be made available with the release.

AI USE STATEMENT

We used generative AI tools to assist with conceptual framing, feedback on experimental design and result interpretation, code editing, experiment bookkeeping, manuscript drafting and revision, literature and reference search, and figure-design feedback. AI tools were not treated as an independent source of experimental evidence or as the final authority for research decisions. Suggestions concerning experimental design and interpretation were checked against the implemented protocols, run logs, per-seed results, and controlled comparisons reported in the paper. AI-assisted code changes were reviewed and tested before use; references were checked against the corresponding conference or journal records; and manuscript text and figures were reviewed against the underlying experiments and source material. The authors made the final decisions on the research questions, experimental procedures, analyses, and claims, and take responsibility for all content in the submission, including AI-assisted text, code, and artifacts.

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*, pp. 7319–7328, 2021.

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

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. Parameter-efficient fine-tuning design spaces. In *International Conference on Learning Representations*, 2023.

Xuan Cui, HuiYue Li, Run Zeng, Yunfei Zhao, Jinrui Qian, Wei Duan, Bo Liu, and Zhanpeng Zhou. IGU-LoRA: Adaptive Rank Allocation via Integrated Gradients and Uncertainty-Aware Scoring. In *International Conference on Learning Representations*, 2026. arXiv:2603.13792.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems*, 2023.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse Low-rank Adaptation of Pre-trained Language Models. In *Conference on Empirical Methods in Natural Language Processing*, 2023. arXiv:2311.11696.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*, 2019. arXiv:1803.03635.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10323–10337, 2023.

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*, pp. 17783–17806, 2024.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2022.

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

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. arXiv:2106.09685.

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

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

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

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. In *International Conference on Learning Representations*, 2024.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train Big, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers. In *International Conference on Machine Learning*, volume 119, pp. 5958–5968, 2020. URL `https://proceedings.mlr.press/v119/li20m.html`.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *International Conference on Learning Representations*, 2024.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, 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. In *International Conference on Machine Learning*, 2024. arXiv:2402.09353.

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

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

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

Mahdi Nikdan, Soroush Tabesh, Elvir Crnčević, and Dan Alistarh. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 38187–38206, 2024.

Jonas Pfeiffer, Aishwarya Kamath, Andreas R"ucklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, 2021.

Andreas R"ucklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. Adapterdrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7930–7946, 2021.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024.

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 *European Chapter of the Association for Computational Linguistics*, 2023. arXiv:2210.07558.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems*, 2024.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2024.

Adam X. Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian low-rank adaptation for large language models. In *International Conference on Learning Representations*, 2024.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. In *International Conference on Learning Representations*, 2024.

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. In *International Conference on Learning Representations*, 2023. arXiv:2303.10512.

A Protocol and Statistical Reporting

A.1 Models, data, and optimization

Table 6 lists the common training and evaluation settings. GSM8K is the primary task. Llama-3.1-8B GSM8K is a three-seed cross-model check. The remaining tasks are used to characterize the boundary of the dividend.

Table 6: Common protocol. Initial and target ranks are nominal per-module settings; ACP allocates active slots globally across modules. The total deployment budgets are $1568 = 196 \times 8$ for Qwen3-0.6B and $1792 = 224 \times 8$ for Llama-3.1-8B.

Item	Setting
Base models	Qwen3-0.6B; Llama-3.1-8B
Primary task	GSM8K
Boundary tasks	SVAMP, BoolQ, PiQA, MAWPS, MATH, ARC-C, AQuA-RAT
Initial candidate rank	$r_{\text{init}} = 32$
Deployment rank	$r_{\text{target}} = 8$ (pilot also tests $4, 16$)
Main ACP scale	$\alpha/r_{\text{init}} = 16/32 = 0.5$
Fixed-rank DoRA-r8 scale	$\alpha/r = 16/8 = 2$
Optimizer	AdamW
Learning rate	2×10^{-4}
Learning-rate schedule	cosine decay
Effective batch size	32
Epochs	3
ACP competition interval	$\Delta T = 10$ optimizer steps
Score smoothing	EMA with $\beta = 0.85$
Score weights	$\lambda_m = 0.1$, $\lambda_t = 0.05$
Buffer commitment	end select to B_D
Evaluation	exact match, batch size 128

For Qwen3-0.6B, the effective batch is formed by 16×2 gradient accumulation. For Llama-3.1-8B, it is 32×1. The same effective batch size and optimizer-step schedule are used across models. ACP uses global allocation: rank slots compete across all adapted modules rather than within layer-local groups.

A.2 Buffer provenance and capacity definitions

ACP was originally intended to compress the Qwen candidate pool from 6272 to 1568 active slots. An early implementation error stopped compression at 1809 slots; the resulting observation motivated subsequent experiments with an explicit retained buffer. The setting $\rho = 0.154$ is the rounded surplus ratio $1809/1568 - 1$, not a theoretical optimum. Its exploratory origin should be distinguished from the later same-seed comparisons at specified buffer settings.

In the reported protocol, strict means zero-buffer ACP: it starts at 6272 and compresses to 1568. Buffered ACP starts at the same 6272-slot pool, compresses to 1809 for $\rho = 0.154$, and selects 1568 slots at the end. The surplus refers to the compression floor, not the entire training trajectory. In the three-seed five-arm cohort, the average active counts are 4602.6 for strict and 4636.8 for buffered ACP. Both have a peak of 6272.

Global slot budgets are not scalar parameter budgets. The compact factor parameter count is $\sum_\ell r_\ell(d_{in,\ell} + d_{out,\ell})$, plus $\sum_\ell d_{out,\ell}$ DoRA magnitude parameters. The serialized implementation can retain full-size factor tensors and masks; neither a matching slot count nor a smaller mask establishes a smaller checkpoint file.

A.3 Evaluation versions and cohorts

The historical five-seed Qwen GSM8K main cohort uses seeds $42, 123, 3407, 99, 777$. Its ACP evaluations use `evaluate.py` v1.4.0, batch size 128, the canonical 1,319-example GSM8K test split, and exact match. The main ACP records report script SHA256 prefix `4c23f3e2`. Follow-up compression and scale-matched width evaluations use `f298ab99`; the fresh replay cohort and

subsequent rank-24 repeat use `9fb2b370`. All three snapshots are labeled v1.4.0, so the version label alone does not identify the evaluation code. DoRA-r8 and LoRA-r8 share the same seed IDs. Their first three seeds are historical v1.1.0 evaluations; the two added seeds use v1.4.0. We therefore report their aggregate accuracy and the same-seed paired differences separately. The controlled claim is the within-method dividend $U(\rho)$.

The seven-ratio buffer scan is a separate three-seed shape cohort over seeds $42, 123, 3407$. It is not pooled with the five-seed main cohort. Its role is only to show the shape of $U(\rho)$ over the tested widths.

AQuA-RAT uses a fixed 7,473-example training subset drawn without replacement from the official training split with RNG seed 42. The official 254-example validation and test splits are used for training-time dev and final test evaluation. The initial three-seed result is reported separately from the post-hoc five-seed extension.

A.4 Statistical policy

For a seed s, the paired dividend is

$$U_s(\rho) = A_s(\rho) - A_s(0).$$

Reported $\pm$ values are seed standard deviations. For five-seed cohorts we report a two-sided paired t-test on $U_s(\rho)$. These tests describe differences at the specified settings; they do not account for the exploratory choice of the buffer ratio. For three-seed pilots we report the mean, seed SD, and number of positive seeds, without a confirmatory significance claim. The non-monotonicity statement is restricted to the tested width range.

B Scoring, Compression, and Logged Dynamics

B.1 Importance scores and slot selection

Each slot carries a heuristic importance score based on parameter magnitudes and optimizer statistics. The base score is

$$q_j^{\text{base}} = \frac{1}{d_{in}} \sum_k |a_{j,k}| \sqrt{v_{A,j,k}} + \frac{1}{d_{out}} \sum_k |b_{j,k}| \sqrt{v_{B,j,k}},$$

where v_A, v_B are the Adam second-moment statistics of A and B. This is the magnitude-times-sensitivity form used by AdaLoRA-style importance scores, computed here per slot with Adam's second-moment estimate in place of the squared gradient.

Because ACP uses DoRA, the magnitude m is also trainable and couples to the direction B. We add a magnitude score

$$q_j^{\text{mag}} = \frac{1}{d_{out}} \sum_k \sqrt{v_{m,k}}\, |b_{j,k}|,$$

and a trend score

$$q_j^{\text{trend}} = \text{ReLU}\big(q_j^{\text{base}}(t) - q_j^{\text{base}}(t - \Delta T)\big)\,,$$

which rewards slots whose base importance is still increasing. Each component is standardized across the module, and the raw score is

$$s_j = \tilde{q}_j^{\text{base}} + \lambda_m \tilde{q}_j^{\text{mag}} + \lambda_t \tilde{q}_j^{\text{trend}},$$

with $\lambda_m = 0.1$ and $\lambda_t = 0.05$. Let t_k denote the optimizer step of the k-th score update. During soft competition, these updates occur every $\Delta T = 10$ optimizer steps. The exponential moving average is initialized from the first score and subsequently updated as

$$\bar{s}_j(t_1) = s_j(t_1), \qquad \bar{s}_j(t_k) = \beta\, \bar{s}_j(t_{k-1}) + (1-\beta)\, s_j(t_k), \quad k \geq 2, \qquad \beta = 0.85.$$

At every competition step (every $\Delta T = 10$ optimizer steps), we sort all slots globally by $\bar{s}_j$ and keep the top B_t:

$$\chi_j(t+1) = \mathbf{1}[\bar{s}_j(t) \geq \tau_t]\,, \qquad \sum_j \chi_j(t+1) = B_t,$$

where τ_t is the B_t-th largest score, with ties resolved to retain exactly B_t slots. This is a global allocation across adapted modules.

In *soft* mode a masked slot may re-enter if its score re-crosses the threshold; in *hard* mode excluded slots cannot re-enter. The reported main runs use soft competition during compression and freeze the surviving set when the physical active budget reaches the compression floor. The surviving parameters continue training, but the EMA ranking used for selection is no longer updated. At the end of training, final commitment retains the top B_D surviving slots according to the last saved ranking. Thus, a remaining buffer participates in training until the end even though the ranking used for its final removal has already been fixed.

B.2 Compression controller

The adaptive controller updates at competition logging events and at development-set evaluation events. These controller updates are distinct from score updates and physical mask updates. Let u index controller events and let t_u be the corresponding optimizer step. Its start signal is the online winner-set overlap

$$\mathrm{ov@}K(t_u) = \frac{|S_{B_D}(t_u) \cap S_{B_D}(t_u^-)|}{B_D},$$

where S_{B_D} contains the top-B_D slots under the latest available EMA ranking, and t_u^- is the latest recorded history step no later than $t_u - K$. The overlap is set to zero when no such history entry exists. Development-set evaluation events reuse the available ranking rather than recomputing the EMA.

Compression starts when $\mathrm{ov@}K > \mathrm{ov}_{lo}$ and the normalized score gap $g_u > g_{lo}$. To compute g_u, candidates are ordered by their smoothed scores; the difference in mean raw base scores between the first and second quartiles is divided by the sample standard deviation of raw base scores over the candidate pool, with 10^{-12} added to the denominator. Once compression has started, the overlap determines

$$\phi_u = \mathrm{clip}\left(\frac{\mathrm{ov@}K(t_u) - \mathrm{ov}_{lo}}{\mathrm{ov}_{hi} - \mathrm{ov}_{lo}},\, 0,\, 1\right).$$

Before compression starts, $\phi_u = 0$. At a development-set evaluation event, ϕ_u is also set to zero if the current development token accuracy is more than δ_{acc} below the previous evaluation value. This metric is the mean of batch-level next-token accuracies over non-ignored labels under teacher forcing, rather than generated-answer exact match. The current value then becomes the reference for the next evaluation event.

Let C_u be the integer controller budget before event u, with $C_0 = B_{\mathrm{init}}$, and let N_{module} be the number of adapted modules. The update is

$$C_{u+1} = \max\left\{B_T,\, \mathrm{round}\left[N_{\mathrm{module}} \max\left(\frac{B_T}{N_{\mathrm{module}}},\, \frac{C_u}{N_{\mathrm{module}}} - \Delta b_{\max}\phi_u^2\right)\right]\right\}.$$

Rounding follows Python's nearest-integer rule, with ties to even. The controller budget is consumed by the pruning routine; an evaluation event does not directly change the physical mask. Consequently, the controller budget and the physically active slot count need not change at the same optimizer step.

The main GSM8K configurations use $K = 100$, $\mathrm{ov}_{lo} = 0.7$, $\mathrm{ov}_{hi} = 0.9$, $g_{lo} = 0.02$, $\Delta b_{\max} = 2.5$, and $\delta_{\mathrm{acc}} = 0.005$. Development evaluation uses `eval_ratio`=0.05, giving an interval of $\max(1, \lfloor 0.05T \rfloor)$ optimizer steps for a training horizon of T steps. The configurations enable `acp_hard_at_reach`, which bypasses the optional three-update stability wait. The surviving set is frozen once the pruning routine has applied the compression floor. The buffer ratio fixes that floor; the controller and its event schedule determine the path to it.

B.3 Logged diagnostics

The alive set E_t obeys the exact retention identity

$$I_t = \frac{|E_t \cap E_{t-1}|}{|E_t|} = 1 - \frac{J_t}{B_t},$$

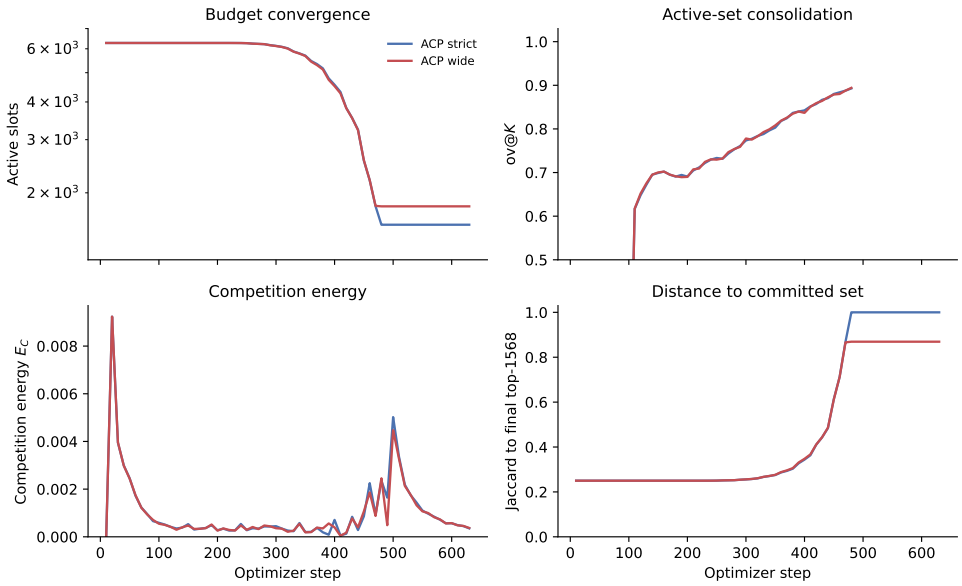

Figure 3: ACP training dynamics on one GSM8K run pair. We show budget convergence, the online overlap $\mathrm{ov}@K$ used by the controller, competition energy, and Jaccard similarity between each step's active set and the final top-1568 commit set. The wide-run Jaccard is bounded by the 1809-slot active set relative to the 1568-slot commit set.

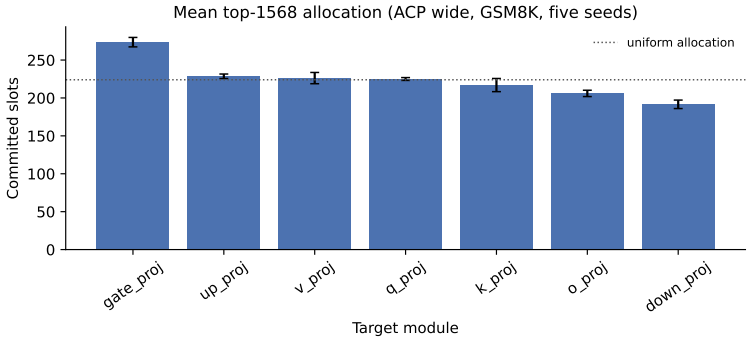

Figure 4: Mean committed slot allocation by target module across the five GSM8K ACP-wide seeds. Error bars are the seed-wise standard deviation. ACP assigns more slots to gate/up projections and fewer to down projections.

where $J_t = |E_t \setminus E_{t-1}|$ counts newly entering slots. For the diagnostic plotted below, let Δt be the optimizer-step interval between observations, M_t the mean score gap between the first and second quartiles of the candidate pool, and p_t the mean trend score in the first quartile. The logger computes

$$E_S = \tfrac{1}{4}\left(J_t + \left|\frac{I_t - I_{t-1}}{\Delta t}\right|\right),$$
$$E_A = \tfrac{1}{4}\left(\left|\frac{M_t - M_{t-1}}{\Delta t}\right| + \left|\frac{p_t - p_{t-1}}{\Delta t}\right|\right), \qquad E_C = E_S + E_A.$$

These mixed-scale diagnostics describe logged structural and score changes; E_C is not a physical energy or the control signal. The reported compression controller uses $\mathrm{ov}@K$ instead.

Table 7: Five-seed GSM8K fixed-rank baselines and ACP arms on Qwen3-0.6B.

Method	n	Active-slot trajectory	Deployment slots	GSM8K acc
LoRA-r8	5	1568	1568	38.59 ± 0.75
DoRA-r8	5	1568	1568	39.10 ± 0.86
ACP strict ($\rho = 0$)	5	$6272 \to 1568$	1568	38.44 ± 0.28
ACP wide ($\rho = 0.154$)	5	$6272 \to 1809$	1568	40.04 ± 0.54
ACP wide ($\rho = 0.3$)	5	$6272 \to 2038$	1568	40.94 ± 1.05

ACP starts with 6272 active slots and commits to 1568 at deployment. These are slot budgets, not matched scalar parameter counts. The historical AdaLoRA run is excluded because it is not implementation-matched. LoRA/DoRA evaluation-version differences are detailed in the supplement.

C Per-Seed Main Results

C.1 S1: Qwen3-0.6B GSM8K five-seed cohort

Against DoRA-r8 on the same five seeds, ACP with $\rho = 0.154$ is higher by 0.94 ± 0.82 points (4 of 5 seeds) and against LoRA-r8 by 1.45 ± 0.71 points (5 of 5). These are contextual comparisons with scale, allocation, and evaluation-version differences; the within-method dividend in the main paper is the controlled result.

Table 8: Per-seed Qwen3-0.6B GSM8K accuracy and paired dividends for the five-seed main cohort. All ACP rows use evaluation v1.4.0.

Seed	$A(0)$	$A(0.154)$	$U(0.154)$	$A(0.3)$	$U(0.3)$
42	38.74	40.86	+2.12	42.23	+3.49
123	38.67	40.03	+1.36	40.18	+1.51
3407	38.44	40.03	+1.59	41.93	+3.49
99	38.29	39.35	+1.06	40.33	+2.04
777	38.06	39.95	+1.89	40.03	+1.97
Mean	38.44 ± 0.28	40.04 ± 0.54	$+1.60 \pm 0.42$	40.94 ± 1.05	$+2.50 \pm 0.93$
Positive seeds	–	–	5/5	–	5/5
p (paired)	–	–	0.0010	–	0.0038

Table 9: Fixed-rank baseline context for the same five seeds. Differences are ACP $A(0.154)$ minus the corresponding baseline. The first three baseline seeds use historical v1.1.0 evaluations; seeds 99 and 777 use v1.4.0.

Seed	DoRA-r8	ΔDoRA	LoRA-r8	ΔLoRA
42	39.04	+1.82	38.82	+2.04
123	39.80	+0.23	38.59	+1.44
3407	40.11	−0.08	39.73	+0.30
99	38.06	+1.29	37.91	+1.44
777	38.51	+1.44	37.91	+2.04
Mean ± SD	39.10 ± 0.86	$+0.94 \pm 0.82$	38.59 ± 0.75	$+1.45 \pm 0.71$
Positive seeds	–	4/5	–	5/5

Every main-cohort seed has a positive within-method contrast at both tested buffer ratios. The ACP-to-DoRA contrast is weaker and also has scale, allocation, and evaluation-version differences; it is baseline context rather than the controlled claim.

Table 10: GSM8K accuracy for the separate three-seed buffer-ratio scan. The penultimate row reports mean accuracy and seed SD; the final row reports the mean paired dividend U relative to $\rho = 0$.

Seed	$\rho = 0$	0.077	0.154	0.23	0.3	0.5	0.8
42	39.20	40.03	41.32	39.80	41.77	40.18	40.71
123	38.36	39.20	40.03	41.39	40.18	39.04	38.21
3407	38.51	37.38	39.04	40.49	38.44	38.82	39.35
Mean ± SD	38.69±0.45	38.87±1.36	40.13±1.14	40.56±0.80	40.13±1.67	39.35±0.73	39.42±1.25
U (mean)	0.00	+0.18	+1.44	+1.87	+1.44	+0.66	+0.73

C.2 S2: Seven-ratio shape scan

The observed mean dividend is non-monotonic over the tested buffer ratios. The scan does not identify a universal optimum, and this three-seed cohort is kept separate from the five-seed main result.

C.3 S3: Llama-3.1-8B GSM8K cross-model check

Table 11: Llama-3.1-8B GSM8K per-seed results. ACP uses the same seeds as the Qwen main cohort's first three seeds. DoRA-r8 is included as a fixed-rank context row.

Seed	ACP $A(0)$	ACP $A(0.154)$	$U(0.154)$	ACP $A(0.3)$	$U(0.3)$	DoRA-r8
42	56.79	58.61	+1.82	54.66	−2.12	61.64
123	59.36	61.41	+2.05	59.14	−0.23	61.64
3407	57.09	59.06	+1.97	59.44	+2.35	60.73
Mean ± SD	57.75±1.41	59.69±1.51	+1.95±0.12	57.75±2.67	0.00±2.25	61.33±0.53
Positive seeds	–	–	3/3	–	1/3	–

The $(0, 0.154)$ dividend is positive on all three Llama seeds. The larger $\rho = 0.3$ buffer has approximately zero mean dividend and is positive on only one seed. These exploratory results do not support a general buffer ordering or a cross-method advantage over DoRA-r8.

D Training Width and Selection Controls

D.1 S4: Width and selection during training

Table 12: Five-arm comparison of training width and selection protocols. Slot-time is the integrated active-slot count in millions of slot-steps, not measured FLOPs. C/D/E disable stepwise selection and commit only at the end.

Arm	Mean active	Slot-time (M)	s42	s123	s3407	Mean ± SD	Stepwise	Commit	Wall (s)
A Strict	4602.6	2.900	38.74	38.67	38.44	38.62±0.16	yes	1568	1476.5
B Wide-Select	4636.8	2.921	40.86	40.03	40.03	40.31±0.48	yes	end-select 1568	1244.6
C Fixed rank 9	1764.0	1.111	37.38	37.68	38.59	37.88±0.63	no	end-select 1568	1422.8
D Fixed rank 32	6272.0	3.951	44.43	39.27	38.89	40.86±3.09	no	end-select 1568	1501.9
E Fixed rank 24	4704.0	2.964	39.27	42.84	40.56	40.89±1.80	no	end-select 1568	1477.3

The width contrast $E - C$ is positive on all three seeds, but C uses scale $4/9$ whereas E uses $12/24 = 0.5$; it is not a pure width intervention. B and E have approximately matched average active counts (4636.8 versus 4704), not identical trajectories. Their mean difference ($E-B = +0.58$ pp, $2/3$ positive) shows that no-selection training can attain similar accuracy in this pilot, not that selection is equivalent or harmful. The full-width arm D has substantial variance and no further mean improvement over E. None of these three-seed contrasts identifies a precise saturation point.

Table 13: Paired five-arm contrasts in percentage points, computed from integer correct-answer counts before rounding.

Contrast	s42	s123	s3407	Mean
$E - C$	+1.90	+5.16	+1.97	+3.01
$E - B$	-1.59	+2.81	+0.53	+0.58
$D - E$	+5.16	-3.56	-1.67	-0.03
$D - C$	+7.05	+1.59	+0.30	+2.98
$E - A$	+0.53	+4.17	+2.12	+2.27
$B - C$	+3.49	+2.35	+1.44	+2.43

Table 14: Additional-training control on Qwen3-0.6B GSM8K. Zero-buffer ACP is extended to approximately match integrated active slots to the buffered run; strict still begins with 6272 slots.

Arm	s42	s123	s3407	Mean $\pm$ SD
Strict baseline	38.74	38.67	38.44	38.62 ± 0.16
Slot-time matched	38.74	39.50	36.92	38.39 ± 1.32
Buffer $\rho = 0.3$	42.23	40.18	41.93	41.45 ± 1.10

D.2 S5: Active-slot-time control

The extended zero-buffer runs use 648, 643, and 645 optimizer steps for seeds 42, 123, and 3407, respectively, versus 630 in the reference runs. The cosine learning-rate schedule is rescaled to each extended horizon, so this intervention changes both the number of updates and the learning-rate trajectory. They are close to strict on average and below the buffer arm on all three seeds. The matched quantity is $\sum_t B_t$, an active-capacity proxy. Dense masked factors retain their tensor dimensions in the implementation, so this is not an exact computational-cost match and does not exclude every compute-based explanation.

D.3 S6: End-commitment and timing controls

Table 15: Illustrative end-commitment comparison on Qwen3-0.6B GSM8K (seed 42). These are separate trained adapters, not alternative masks evaluated on identical weights. Zero-buffer ACP is the strict reference.

Structure	Test accuracy	Final capacity
Zero-buffer ACP	38.74	1568
Wide-to-narrow	40.64	1568
Wide-to-wide	38.51	1809

Table 16: Commitment-time sweep on Qwen3-0.6B GSM8K (three seeds). Each run commits at the indicated fraction of $T = 630$ optimizer steps, continues training the retained slots to T, and is then evaluated.

Commit fraction	s42	s123	s3407	Mean $\pm$ SD
20%	36.92	36.24	34.65	35.94 ± 1.17
40%	37.76	37.60	35.71	37.02 ± 1.14
60%	37.91	37.15	38.29	37.78 ± 0.58
80%	37.07	38.44	37.00	37.50 ± 0.81
100%	40.26	38.89	37.38	38.84 ± 1.44

The end-commitment mean exceeds the mean at each earlier tested point. Its paired improvement over 20% commitment is 2.91 ± 0.37 percentage points, positive on all three seeds. The means are not monotonic across intermediate commitment times. All runs share the training horizon, but their final masks can differ. For the separate seed-42 Wide$\rightarrow$Narrow and Wide$\rightarrow$Wide runs, the top-1568

alive-slot sets reconstructed from the logged freeze snapshots share 1492 slots, giving an overlap of $1492/1568 = 95.2\%$. This statistic compares ranked subsets from those snapshots; it does not establish identical deployed masks or weights. The controls characterize commitment sensitivity without isolating final-structure quality from optimization history.

E Final Compression, Scale-Matched Width, and Mask Replay

E.1 Compression from the same trained checkpoint

The source adapters are the rank-24 arm E, trained with $\alpha = 12$ and all $196 \times 24 = 4704$ slots active throughout 630 optimizer steps. Score selection retains 1568 slots at saving time. The source files retain the full factor tensors, which allows us to compare three deployment updates without retraining. The pre-conversion structural checks confirm 196 modules, rank-24 factors, factor scale 0.5, 4704 active training slots, and a 1568-slot saved mask.

For module ℓ, let r_ℓ be the number of slots retained by Score. The SVD control uses the full unmasked product $B_\ell A_\ell$ and computes a rank-r_ℓ approximation. To avoid constructing a large dense product, we compute thin QR factorizations $B_\ell = Q_B R_B$ and $A_\ell^\top = Q_A R_A$, followed by the SVD of the small matrix $R_B R_A^\top$. Retaining its first r_ℓ singular components gives balanced factors, with the singular-value square roots split between them. QR and SVD are computed in float64; the factors are then saved in the source tensor format, padded to the original rank 24. The scale remains $12/24$, and the DoRA magnitude is copied. Full uses the unchanged factors with all 4704 slots enabled. SVD and Score share the per-module rank pattern, so their compact parameter counts match.

Conversion reports identify the same source checkpoint hash for the SVD and Full alternatives of each seed. The Score rank ranges across modules are 4–12, 3–12, and 5–13 for seeds 42, 123, and 3407, respectively; each sums to 1568. The conversion audit checks source hashes, configuration fields, and recorded rank patterns. It does not include a separate round-trip comparison of serialized factor values. Table 17 reports all three seeds under one evaluation snapshot.

Table 17: Same-checkpoint compression results on GSM8K. All accuracies are $100 \times \text{correct}/1319$; contrasts are computed before rounding.

Seed	Score	SVD	Full	Score − SVD	Score − Full
42	39.88	36.47	37.15	+3.41	+2.73
123	42.08	39.04	38.29	+3.03	+3.79
3407	39.88	36.85	38.21	+3.03	+1.67
Mean ± SD	40.61 ± 1.27	37.45 ± 1.39	37.88 ± 0.64	$+3.16 \pm 0.22$	$+2.73 \pm 1.06$

Score has 526, 555, and 526 correct answers; SVD has 481, 515, and 486; Full has 490, 505, and 504. The SVD objective minimizes the Frobenius-norm approximation error of each unnormalized factor product at its prescribed module rank. This does not optimize task accuracy after DoRA row normalization, which changes when directions are removed. The comparison concerns these deployment operators on fixed source weights, rather than the performance of a separately trained SVD-based adapter method.

E.2 Scale-matched width control

The historical rank-9 arm uses $\alpha = 4$, whereas rank 24 uses $\alpha = 12$. We train a rank-9 control with $\alpha = 4.5$ to match the scale at 0.5. All other shared configuration fields match the historical rank-24 runs except the run name and initial rank. Across each of the three controls, 63 budget records cover steps 10–630, with 1764 active slots throughout and no slot entries, exits, or hard pruning. End selection retains 1568 slots. Training times are 1506.38, 1521.70, and 1533.41 seconds; recorded peak memory is 22.57 GB for each.

The gap persists after matching scale. Initialization dimensions and the learned final masks still differ, so the comparison does not isolate the role of training history while holding the final structure fixed.

Table 18: Width comparison at factor scale 0.5. The rank-24 row uses the Score evaluation from Table 17; the rank-9 controls are newly trained checkpoints.

Training width	s42	s123	s3407	Mean ± SD
Rank 9, $\alpha = 4.5$	38.29	38.51	39.27	38.69 ± 0.52
Rank 24, $\alpha = 12$	39.88	42.08	39.88	40.61 ± 1.27
Paired difference	+1.59	+3.56	+0.61	$+1.92 \pm 1.51$

E.3 Additional historical width seeds

Ranks 9 and 24 also have complete historical runs for seeds 99 and 777. We report all five available seeds in Table 19, without extending the rank-32 arm beyond its three available seeds. These results use the historical evaluation snapshot and retain the original scale difference. They are not pooled with the scale-matched control.

Table 19: Complete historical five-seed fixed-width comparison, with final selection to 1568 slots. The factor scales are $4/9$ and $12/24$.

Arm	s42	s123	s3407	s99	s777	Mean ± SD
Rank 9, $\alpha = 4$	37.38	37.68	38.59	37.30	37.98	37.79 ± 0.52
Rank 24, $\alpha = 12$	39.27	42.84	40.56	39.35	42.15	40.83 ± 1.62
Paired difference	+1.90	+5.16	+1.97	+2.05	+4.17	$+3.05 \pm 1.52$

E.4 Replay with the final mask fixed from initialization

The replay experiment creates three new buffer/fixed-mask pairs. Before buffer training, the initial rank-32 factors and DoRA magnitudes are saved. After training, the committed mask is extracted and loaded with that initial state for replay. The replay trains only those active slots throughout; it does not resume the trained buffer checkpoint. Training-subset selection, development splitting, and sampler order use separate random generators so that constructing the different masked models does not change the example order. The saved initial-state hashes and actual training-order records agree within each pair. Because the mask is obtained from the completed buffer run, replay is a diagnostic conditional on that selected structure, rather than a standalone procedure for choosing the structure before training.

Both arms use $\alpha = 16$, learning rate 2×10^{-4}, effective batch size 32, and 630 optimizer steps. Each has 63 budget records and 90 training-metric records ending at step 630, with a terminal learning rate of zero. Buffer runs first log the 1809-slot floor at steps 470, 450, and 480; these are observation times at ten-step intervals. Replays retain exactly 1568 slots throughout. Exported final masks match element by element in all 196 modules for every pair, with per-module rank ranges 3–14, 3–13, and 2–13. The mask audit recomputes these counts from the exported arrays instead of relying on a pass flag.

Table 20: Fresh final-mask replay cohort. Accuracy and paired differences are computed from the correct-answer counts shown in parentheses.

Seed	Buffered ACP	Fixed-mask replay	Paired difference
42	41.39 (546)	36.16 (477)	+5.23
123	38.29 (505)	38.36 (506)	−0.08
3407	37.76 (498)	37.60 (496)	+0.15
Mean ± SD	39.15 ± 1.97	37.38 ± 1.12	$+1.77 \pm 3.00$

The paired median is $+0.15$ points, and two of three differences are positive. Seed 42 contributes 69 of the 70 net additional correct answers. Both its runs reach the prescribed horizon with finite training losses; their mean loss over the last ten records is 0.37670 versus 0.37702. These checks do not explain the accuracy gap, but they provide no evidence of an incomplete replay run. Matching

the final slot identities makes different masks unavailable as an explanation within these pairs. The heterogeneous result still leaves a broad claim of uniformly better training history unsupported.

E.5 Evaluation snapshots and repeatability

The historical main and fixed-width evaluations use script hash prefix `4c23f3e2`. The same-checkpoint compression and scale-matched width evaluations use `f298ab99`; the fresh replay and rank-24 repeat use `9fb2b370`. All are labeled evaluation v1.4.0. The follow-up GSM8K records specify batch size 128 and maximum input length 1024; the evaluation code uses greedy generation with a default maximum of 512 new tokens. Accuracy is recomputed from integer correct/total counts.

Table 21: Rank-24 Score evaluations of the same named source adapters. Follow-up results are stable under the repeat; the historical snapshot gives different counts.

Evaluation	s42 correct	s123 correct	s3407 correct	Accuracy mean ± SD
Historical (`4c23f3e2`)	518	565	535	40.89 ± 1.80
Compression control (`f298ab99`)	526	555	526	40.61 ± 1.27
Repeat (`9fb2b370`)	526	555	526	40.61 ± 1.27

All 3957 generated answers, extracted predictions, targets, and correctness labels in the repeat match the corresponding compression-control records exactly. The changed script fixes a logging error: earlier prediction files stored a stale prompt string while generation and scoring used the correctly aligned prompt and target lists. The fix changes the stored prompt field, not these repeated answers or scores. The repeat's ordered question/target sequence also matches the replay cohort. It supports stability of the follow-up Score reference; it does not establish identical hardware, dependency versions, or model-file contents across historical and follow-up evaluations. The cause of the historical count differences remains unresolved, so the old and new scores are not mixed within a comparison.

Fixed-rank baseline re-evaluations under `f298ab99` give LoRA-r8 accuracies 39.65, 37.53, and 39.12 (mean 38.77 ± 1.10), and DoRA-r8 accuracies 38.97, 39.73, and 39.80 (mean 39.50 ± 0.46). These are a separate three-seed evaluation cohort, not replacements for individual rows in the historical five-seed table.

F Task and Model Dependence

F.1 S7: Full task summary

Table 22: Capacity dividend at $\rho = 0.154$ across tasks and models. Seed signs count strictly positive differences unless explicitly marked non-negative. ARC-C lacked a supported exact-match evaluation, so no comparable effect estimate or seed-sign count is reported. AQuA-RAT is a post-hoc five-seed extension.

Task	Model	n	$A(\rho)$ mean	$U(\rho)$ mean ± SD	Seed signs	Role
GSM8K	Qwen3-0.6B	5	40.04	$+1.60 \pm 0.42$	5/5	main positive
GSM8K	Llama-3.1-8B	3	59.69	$+1.95 \pm 0.12$	3/3	cross-model positive
SVAMP	Qwen3-0.6B	5	31.80	$+2.60 \pm 2.58$	4/5 non-negative	positive mean
SVAMP	Llama-3.1-8B	5	68.40	$+0.40 \pm 2.36$	2/5	mixed seed signs
BoolQ	Qwen3-0.6B	3	83.73	-0.02 ± 0.31	1/3	near-zero mean
PiQA	Qwen3-0.6B	3	75.45	$+0.04 \pm 1.12$	2/3	near-zero mean
BoolQ	Llama-3.1-8B	1	91.44	+0.06	1/1	single-seed estimate
MAWPS	Qwen3-0.6B	3	38.97	-0.83 ± 2.29	1/3	inconclusive
MATH	Qwen3-0.6B	3	1.40	$+0.73 \pm 0.76$	–	exploratory, low floor
AQuA-RAT	Qwen3-0.6B	5	32.05	$+0.79 \pm 2.26$	4/5	inconclusive
ARC-C	Qwen3-0.6B	3	–	–	–	unsupported evaluation

Table 23: AQuA-RAT test results. The three-seed cohort is the initial pilot; seeds 99 and 777 are a post-hoc extension. Evaluation uses 254 test examples and `max_new_tokens`=8. Accuracies, paired differences, and sample SDs are computed from integer correct-answer counts before rounding.

Cohort / seed	$A(0)$	$A(0.154)$	$U(0.154)$	Note
42	30.31	33.86	+3.54	initial
123	31.50	32.68	+1.18	initial
3407	30.31	31.50	+1.18	initial
99	31.50	32.28	+0.79	post-hoc
777	32.68	29.92	−2.76	post-hoc
3-seed mean	30.71±0.69	32.68±1.18	+1.97±1.36	3/3 positive
5-seed mean	31.26±0.99	32.05±1.46	+0.79±2.26	4/5 positive

Table 24: Within-method GSM8K dividend across deployment ranks on Qwen3-0.6B.

r_{target}	κ	ρ	n	$U(\rho)$	Positive seeds
4	8	0.154	3	$+1.74 \pm 0.77$	3/3
8	4	0.154	5	$+1.60 \pm 0.42$	5/5
16	2	0.154	3	$+0.45 \pm 0.72$	2/3
16	2	0.3	3	$+0.94 \pm 0.86$	3/3

The $r = 8$ row is the five-seed main cohort; the other rows are three-seed pilots. Both tested $r = 16$ buffers are shown to avoid conflating changes in ρ with changes in κ.

F.2 S8: AQuA-RAT initial and post-hoc cohorts

The post-hoc extension weakens the three-seed result. The diagnostic paired test on five seeds gives $p = 0.480$; it is not a confirmatory claim. The added seed 777 also has weaker post-reach dev behavior, so the reversal is not treated as a test-only anomaly.

For cross-method context, the fixed-rank AQuA-RAT controls over seeds 42/123/3407 are LoRA-r8 $= 31.89 \pm 0.39$ and DoRA-r8 $= 32.02 \pm 0.99$. ACP wide reaches 32.68 ± 1.18 on the same three seeds. These summaries report seed SDs, not confidence intervals. We treat the three-seed comparison as descriptive and do not claim a robust cross-method advantage.

G Pilots Across Deployment Ranks

G.1 S9: Deployment-rank pilot

The within-method dividend is positive on all three seeds at $r = 4$ with $\rho = 0.154$, and at $r = 16$ with the larger $\rho = 0.3$ buffer. The $r = 16$ result with $\rho = 0.154$ is weaker. These three-seed pilots provide directional evidence at the tested rank–buffer combinations, without establishing a general trend across ranks. Buffered ACP is below the fixed-rank DoRA reference on all three seeds at $r = 4$, whereas the $r = 16$ contrasts have mixed signs. The positive ACP-to-DoRA difference reported for the historical $r = 8$ cohort remains contextual because scale, allocation, and evaluation-version differences limit that comparison.

H Compute, Reproducibility, and Artifact Audit

H.1 Compute accounting

The five-seed Qwen GSM8K main cohort records mean training time 1499.4 s for strict and 1391.4 s for buffered ACP, with 22.8 GB peak GPU memory in both. The three-seed Llama resource records instead report 43.92 GB peak memory for ACP and 43.16 GB for DoRA; a 24 GB hardware description therefore applies only to the Qwen setting, not both models. These are recorded run

Table 25: Three-seed target-rank pilot on Qwen3-0.6B GSM8K. All ACP arms use $r_{\text{init}} = 32$; DoRA is trained directly at the target rank. The compression ratio is $\kappa = r_{\text{init}}/r_{\text{target}}$. Statistics and paired differences are computed from integer correct-answer counts before rounding.

r_{target}	Seed	ACP strict	ACP wide	U	Buffer	DoRA	ACP wide − DoRA
4	42	33.36	35.25	+1.90	0.154	38.97	−3.71
4	123	31.69	34.12	+2.43	0.154	38.89	−4.78
4	3407	32.60	33.51	+0.91	0.154	38.06	−4.55
4	Mean	32.55±0.84	34.29±0.89	+1.74±0.77	0.154	38.64±0.50	−4.35
16	42	37.98	37.98	0.00	0.154	38.51	−0.53
16	123	38.74	40.03	+1.29	0.154	38.36	+1.67
16	3407	37.60	37.68	+0.08	0.154	39.42	−1.74
16	Mean	38.11±0.58	38.56±1.28	+0.45±0.72	0.154	38.77±0.57	−0.20
16	42	37.98	39.88	+1.90	0.30	38.51	+1.36
16	123	38.74	39.42	+0.68	0.30	38.36	+1.06
16	3407	37.60	37.83	+0.23	0.30	39.42	−1.59
16	Mean	38.11±0.58	39.04±1.07	+0.94±0.86	0.30	38.77±0.57	+0.28

costs, not controlled hardware-speed comparisons. Table 12 reports slot-time and wall time for the width arms. Active-slot-time does not account for module dimensions, dense masked computation, optimizer work, or the frozen base model, and is not a measured FLOP count.

H.2 Reproduction contract

The release includes the training scripts `astdora.py`, `dora.py`, `lora.py`, and `adalora.py`; the evaluation script `evaluate.py`; per-run `config.json` files; the frozen evidence CSV; and the evaluation JSON files. The main cohort uses seeds $42, 123, 3407, 99, 777$. The shape cohort uses seeds $42, 123, 3407$. The target-rank pilot uses seeds $42, 123, 3407$. The AQuA-RAT initial cohort uses seeds $42, 123, 3407$, with the post-hoc extension labeled separately.

The primary evaluation command is structurally

```
python evaluate.py --model_path <model>
  --adapter_paths <committed-adapter> --dataset <task>
  --data_file <test.jsonl> --batch_size 128
  --output_file <eval.json>
```

For GSM8K, the default generation limit is used. For AQuA-RAT, `max_new_tokens=8` is used because each answer is a single option letter. The v1.4.0 evaluation records include the evaluation version, data file, batch size, and script hash in their metadata.

The historical main-evidence CSV and the separately audited follow-up evaluation records provide the numerical sources for the manuscript. Follow-up compression, width, and replay results are kept separate from the historical cohorts. Historical DoRA, LoRA, and AdaLoRA rows retain their original evaluation-version labels. No cloud adapter paths, author identities, or tracking links are included in the submission.