COBRA: COMPLETION-ORDER BATCHING AS A REWARD-STRATIFIED ADVANTAGE

Anonymous authors
Paper under double-blind review

ABSTRACT

COBRA improves early learning from verifiable training problems in single-response reinforcement learning. We retain every rollout, form update blocks by completion order, and switch from local to pooled reward moments after an initial learning phase. From a shared format-calibrated Qwen2.5-7B start, two runs improve update-80 accuracy over random batching by 3.15–3.49 percentage points on DeepMath and 4.20–7.18 on MATH500. Each arm has seen about 45,900 distinct training problems in 61,440 prompt presentations. On the retained evaluation grid, mean MATH500 accuracy first exceeds 70% at update 80 versus 120–160 for random/local, after 24–35% fewer distinct training problems. Completion grouping creates length and reward strata; an exploratory, single-seed centering intervention supports local centering as a contributor to subsequent growth. Switching to pooled moments limits that growth through 200 updates on Qwen2.5-7B and Qwen3-4B. On held-out three-to-seven-person logic puzzles, the same recipe gains 2.35–6.13 points over scheduled random grouping.

1 INTRODUCTION

Completion-Order Batching as a Reward-Stratified Advantage (COBRA) uses the order in which responses finish to construct reinforcement-learning batches. It groups a fully retained rollout pool into blocks with distinct reward statistics. Nearby completions share a local reward reference during early training; a fixed switch to pooled moments then limits response growth. The grouping requires no additional rollouts, reward evaluations, or learned difficulty predictor. Figure 1 illustrates the recipe.

We assess sample efficiency by accuracy as a function of the number of distinct verifiable input–answer pairs encountered during training. Sampling is with replacement, and each prompt presentation generates one response. The four main mathematical arms within a seed receive the same problem multisets in every rollout pool. From a shared format-calibrated Qwen2.5-7B start, COBRA's update-80 gains over random batching are 3.15 and 3.49 percentage points on DeepMath development and 7.18 and 4.20 on MATH500 across two seeds. At this point, each arm has seen 45,882 or 45,847 distinct training problems in seeds 17 and 314159, respectively, through 61,440 prompt presentations. Moderate sampled-accuracy targets also appear earlier on the retained checkpoint grid, and the pooled-moment schedule limits response growth through 200 updates.

Each round generates one response per prompt under a fixed policy, partitions the completed pool into four blocks, and consumes them in four sequential updates. Completion-order blocks develop different length and reward distributions. Local normalization compares responses within these strata: for a late, low-reward block, it supplies a lower reward reference than the pool as a whole. The resulting learning coefficients differ for both successful and failed responses.

The early gains accompany longer responses. Continued local normalization can eventually produce severe growth and truncation, motivating a change in reward reference. The fixed switch to pooled moments combines the early-learning benefit with controlled response growth through the 200-update horizon. Training and inference token costs account separately for the longer responses.

Conditional advantage methods already use metric-dependent reward comparisons and mixtures of group-conditioned advantages (Chen et al., 2025; Zhu et al., 2025), while work on loss reduction an-

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.

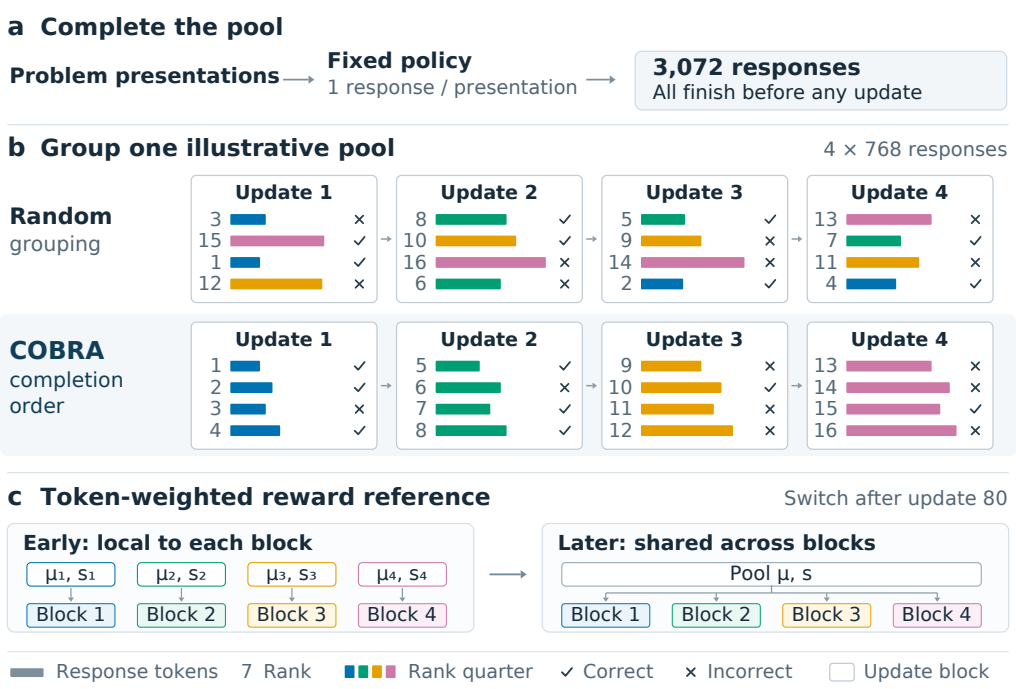


Figure 1: **COBRA overview.** (a) A fixed policy generates a complete pool before actor updates. (b) Four sequential updates consume every response once. The two rows regroup the same 16 illustrative responses; actual runs share problem multisets and generate their own responses. Numbers and colors indicate completion ranks and their quarters; bar lengths show response tokens. Check marks and crosses denote correct and incorrect outcomes. (c) Token-weighted reward moments are local through update 80 and pooled afterward. Each update retains its own token-mean loss denominator.

alyzes response-length weighting (Liu et al., 2025; Yu et al., 2025). Our contribution is the measured behavior of these components in a single-response, sequential-update trainer:

1. **Earlier useful accuracy.** Completion grouping improves early mathematical accuracy from the same labeled problems in both calibrated Qwen2.5 seeds. The 70% MATH500 target first appears at update 80 in both, versus 160 and 120 for either random control, after 35.0% and 23.5% fewer distinct training problems at those observed checkpoints.
2. **A normalization intervention that limits growth.** Completion-derived length and reward strata emerge before truncation. An exploratory common-checkpoint intervention identifies local centering as a contributor to growth; switching to pooled moments mitigates that growth through 200 updates on Qwen2.5-7B and Qwen3-4B.
3. **A held-out logic benefit with explicit costs.** On three-to-seven-person Knights and Knaves puzzles, final accuracy gains persist against both random controls, including the same normalization schedule. Response lengths and training-token counts quantify the cost of those gains.

2 Batch Construction and Normalization

Retained rollout pools. Let π_θ be a language-model policy. In round j, we draw N prompts and sample one response y_i per prompt x_i. A verifier supplies terminal reward $r_i \in \{0, 1\}$. Response i has L_i valid generated tokens and completion rank c_i, recorded when it enters the response buffer's ready set. All N responses finish before any actor update. Completion grouping sorts by c_i and partitions the list into K equal blocks $B_1, \ldots, B_K$, consumed in forward order. Random grouping uses a random permutation to form the blocks. Both consume every response exactly once. The main protocol uses $N = 3{,}072$ and $K = 4$, giving four 768-response optimizer updates per pool.

The rollout policy is synchronized after these four updates, so block B_b is consumed after $b-1$ intervening actor updates. Random and completion grouping share this synchronization schedule.

Reward-statistic scope. With discount factor one, the terminal return r_i is repeated over valid response tokens. For a trajectory set S, define

$$T_S = \sum_{i \in S} L_i, \qquad \mu_S = \frac{\sum_{i \in S} L_i r_i}{T_S}, \qquad s_S = \max\left(\sqrt{\frac{\sum_{i \in S} L_i (r_i - \mu_S)^2}{T_S - 1}}, \epsilon \right). \tag{1}$$

Here T_S counts valid response tokens and $\epsilon = 0.05$ floors the standard deviation. With $P = \bigcup_{b=1}^{K} B_b$, block-local and pooled advantages are

$$A_i^{\text{local}} = \frac{r_i - \mu_{B_b}}{s_{B_b}}, \quad i \in B_b, \qquad A_i^{\text{pool}} = \frac{r_i - \mu_P}{s_P}. \tag{2}$$

Padding is excluded and end-of-sequence tokens are included when present. A response stopped at the cap receives zero reward; its valid tokens remain in the moments and actor loss. Longer responses thus contribute more to the reward moments.

The local-to-pooled decomposition used in stratified advantage analysis (Zhu et al., 2025) gives, for $i \in B_b$,

$$A_i^{\text{local}} = \frac{s_P}{s_{B_b}} A_i^{\text{pool}} + \frac{\mu_P - \mu_{B_b}}{s_{B_b}}. \tag{3}$$

Local normalization applies a block-dependent rescaling and offset. If a late block has a lower reward mean than the pool, local centering moves every coefficient in that block upward before scaling. Failed responses receive less negative centered returns; successful responses receive more positive ones. This motivates varying the center and scale independently.

Completion order and reward variation. Completion rank orders responses without using their correctness to assign block membership. It allows blocks with different reward rates to retain both successful and failed responses for local comparison. With binary rewards, strict reward sorting followed by K contiguous blocks leaves at most one block containing both outcomes. Every other block has constant reward, giving $r_i - \mu_{B_b} = 0$ and hence zero local advantages in Eq. (2). Completion grouping supports reward stratification with informative local comparisons; the recorded completion blocks contain both outcomes (Appendix B). This argument concerns the local-normalization phase; pooled normalization uses reward variation across the entire pool.

Normalization schedule. The evaluated recipe, COBRA (completion/scheduled), uses completion-order blocks and a fixed local-to-pooled reward-moment schedule:

$$A_i(j) = \begin{cases} A_i^{\text{local}}, & j \le j_\star, \\ A_i^{\text{pool}}, & j > j_\star. \end{cases} \tag{4}$$

The main switch is $j_\star = 20$, so pooled moments begin after update 80. Its timing is selected on an exploratory seed and held fixed for both subsequent seeds. Scheduled random and completion arms use the same rule. Block construction and consumption order remain fixed across the switch. Scheduled conditional reward comparisons (Chen et al., 2025) and mixtures of local and global normalized advantages (Zhu et al., 2025) provide antecedents; here the schedule provides an intervention on completion-grouped learning.

Loss-denominator scope. Advantages multiply a clipped token-level surrogate from proximal policy optimization (PPO; Schulman et al., 2017). Old actor log probabilities remain fixed across the round's four updates. A detached, capped actor-to-behavior probability ratio accounts for differences between the rollout and actor implementations. Let $\ell_{it}(\theta)$ denote the resulting token loss and $T_b = \sum_{i \in B_b} L_i$. The block reduction is

$$\mathcal{L}_b(\theta) = \frac{1}{T_b} \sum_{i \in B_b} \sum_{t=1}^{L_i} \ell_{it}(\theta). \tag{5}$$

Each denominator counts valid tokens globally across the devices and dynamic microbatches for that update. Response-length effects of loss reduction are established (Liu et al., 2025; Yu et al., 2025); the relevant unit here is a sequential update block.

Appendix A specifies clipping, correction weights, optimizer settings, and the global reduction across dynamic microbatches.

Executing the recipe. Algorithm 1 in Appendix A.1 specifies one complete training run. Each pool's reward statistics remain fixed across its four updates. After the switch, blocks share pooled moments and retain separate token-mean loss denominators. Completion ranks determine block membership; verifier outcomes determine advantages.

3 Experimental Design

Models and training. The main Qwen2.5-7B experiments (Yang et al., 2024) share a 30-update format calibration that rewards output structure without checking correctness. This changes common-start MATH500 accuracy from 53.40% to 35.50%. All calibrated-start arms share the resulting weights and fresh task-training optimizer states. Matched random/local and completion/scheduled runs also start directly from official Qwen2.5-7B. Qwen3-4B-Base (Yang et al., 2025) receives no format calibration.

Main runs use seeds 17 and 314159 and train for 200 optimizer updates, using 153,600 prompt presentations each, with one sampled response per presentation. Random/local, random/scheduled, and completion/scheduled arms compare grouping under matched presentation and update budgets; mathematical completion/local controls test normalization stability. Logic uses the first three arms. Qwen2.5 mathematics also includes completion/pooled. The centering intervention uses one calibrated-start, seed-42 update-80 checkpoint and four continuations to update 160. Appendix A gives the allocation, hyperparameters, and starting-model identities.

Tasks. Mathematical training uses 99,447 prepared DeepMath prompts (He et al., 2025), sampled with replacement. All arms have access to this same collection. Distinct-problem counts measure the labeled problems encountered during task training from the shared calibrated start; presentation counts include repeated uses. Retained prompt identities verify identical problem multisets across the four main mathematical arms in every pool within each seed. Evaluation uses 512 DeepMath development problems and MATH500, the 500-problem subset of MATH (Hendrycks et al., 2021; Lightman et al., 2023). MATH500 also participates in recipe development and training monitoring. Deductive-logic training uses 4,479 Knights and Knaves puzzles, with 499 development and 500 held-out test puzzles containing three to seven people (Xie et al., 2024; 2025). Canonical person-renaming filters remove shared logical structures across splits. Both tasks use binary correctness; logic requires a complete, correct assignment of every person's type.

Evaluation and uncertainty. Independent evaluations draw eight responses per prompt at temperature 0.6, top-p 0.95, and a 4,096-token cap. Accuracy averages sampled correctness over all prompts and draws, including failed and capped responses. Mean response length also includes every draw. Paired comparisons first aggregate the eight draws within each prompt. We report training seeds separately; prompt standard errors and confidence intervals condition on each trained policy. Intervals are pointwise; Appendix A specifies the normal and prompt-bootstrap calculations. Response-cap sweeps generate fresh responses under each cap.

4 Early Learning with Completion-Order Batching

At update 80, COBRA improves on random/local by 3.15 ± 0.97 and 3.49 ± 0.85 percentage points on DeepMath, where $\pm$ denotes one paired prompt standard error. MATH500 gains are 7.18 and 4.20 points. Within each seed, the four main arms have seen the same 45,882 or 45,847 distinct training problems through 61,440 prompt presentations. Table 1 gives endpoint accuracies and Figure 2 shows all retained independent checkpoints. The exploratory seed-42 factorial likewise locates the large early difference under local normalization: completion/local reaches 49.68% Deep-

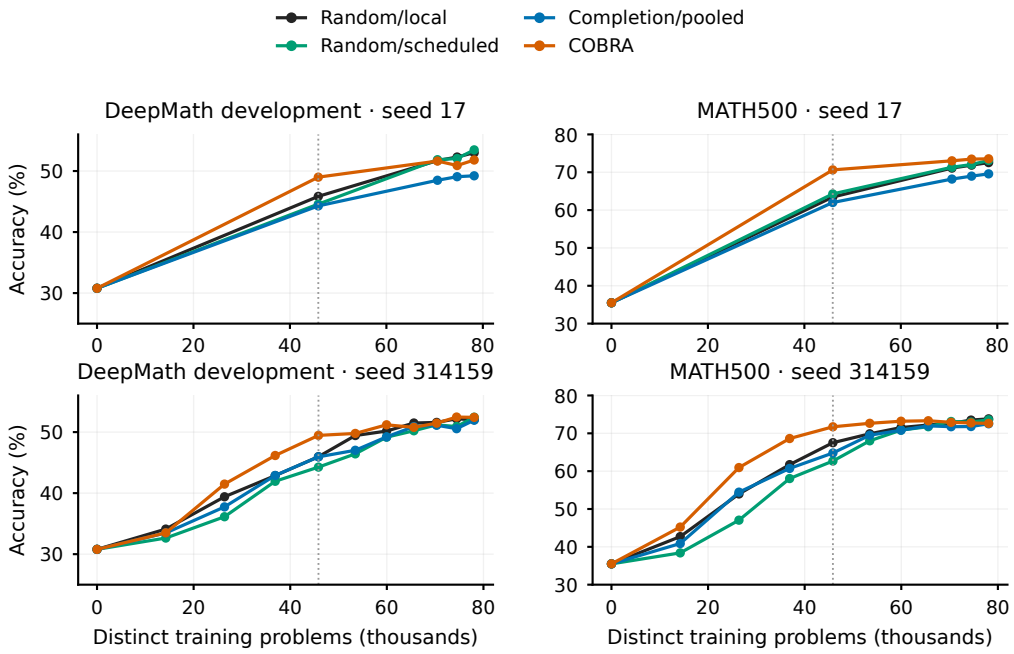


Figure 2: Early mathematical learning from the shared format-calibrated Qwen2.5-7B start. Each point averages eight responses per prompt at a 4,096-token cap. Columns show DeepMath and MATH500; rows retain both training seeds. The horizontal axis counts distinct labeled problems seen during task training; repeated presentations reuse the same problem. Every arm within a seed receives identical problem multisets in each pool. The guide marks the switch after update 80, at 61,440 presentations. Lines connect observed checkpoints; seed 17 has a sparser retained grid, so the curves do not identify exact target-crossing times.

Evaluation set	Update	Presentations	Random/local	Random/sched.	COBRA
DeepMath	80	61,440	45.85 / 45.97	44.58 / 44.26	49.00 / 49.46
MATH500	80	61,440	63.45 / 67.53	64.28 / 62.68	70.62 / 71.73
Held-out logic	200	153,600	46.15 / 46.23	45.25 / 51.50	51.38 / 53.85

Table 1: Early mathematical and final logic accuracy (%) for Qwen2.5-7B. Entries retain seeds 17 / 314159 and both random controls. Evaluation uses eight responses per prompt and a 4,096-token cap; the logic set contains 500 held-out three-to-seven-person puzzles. At update 80, the mathematical arms have seen the same 45,882 / 45,847 distinct training problems. Presentation counts include repeated uses of problems.

Math accuracy versus 46.48% for random/local, while the pooled arms reach 45.07% and 46.51% (Appendix B).

Learning from matched labeled problems. The shared input stream matches the identity and frequency of training problems, as well as the number of presentations. At every pool boundary, the four main arms within a seed have received the same problem multiset and verifier targets. This makes the early curves a comparison of how the update construction uses the same labeled data. The policies can still generate different responses to those problems, and their response lengths can differ. Distinct-problem counts therefore measure labeled-data exposure, while generated tokens account for the amount of generation used to learn from that exposure. The fixed pool size also matches the number of optimizer updates at a given point on each seed's curves. Completion grouping changes which responses share an update block and reward reference within that matched training stream.

Seed	Arm	Update	Distinct problems	Presentations	Accuracy (%)
17	COBRA	80	45,882	61,440	70.62 ± 1.64
17	Random/local	160	70,591	122,880	71.10 ± 1.65
17	Random/scheduled	160	70,591	122,880	71.33 ± 1.65
314159	COBRA	80	45,847	61,440	71.73 ± 1.62
314159	Random/local	120	59,944	92,160	71.53 ± 1.60
314159	Random/scheduled	120	59,944	92,160	70.85 ± 1.63

Table 2: First independently observed 70% MATH500 accuracy for the calibrated Qwen2.5-7B runs. Distinct problems and presentations count task training from the shared initialization; repeated presentations reuse labeled problems. Accuracy is mean $\pm$ prompt standard error over eight responses per problem at a 4,096-token cap. Entries are the first above-target checkpoints on the retained grid; exact crossing times are unobserved.

First observed target attainment. Mean MATH500 accuracy first exceeds 70% at update 80 for completion in both seeds, versus 160 and 120 for either random control. These checkpoints correspond to 45,882/45,847 distinct training problems for completion versus 70,591/59,944 for either control: 35.0%/23.5% fewer distinct problems at the first observed above-target checkpoints. The corresponding prompt presentations are 61,440 in both completion runs versus 122,880/92,160 for the controls. The comparison concerns mean sampled accuracy on the retained evaluation grid; intermediate policies are unobserved, so the exact first-crossing data requirements are undetermined. Table 2 gives the independently evaluated checkpoints and their data consumption. Sampling with replacement makes distinct-problem counts grow more slowly than presentation counts, so the two reductions measure different aspects of training-data use.

All six declared targets and both random comparators appear in Appendix C. The pattern is target dependent: DeepMath's 48% target first appears earlier in both completion runs, while its 50% target appears at the same updates as random/local. Completion grouping's early gain thus has a useful moderate-accuracy regime, with the full target set delimiting that regime. Longer responses make generated-token cost a separate tradeoff; the appendix reports token accounting alongside the full target set.

Single-seed construction controls help locate that effect. At update 80, sorting by realized response length reaches 48.97% DeepMath accuracy; randomizing the consumption order of completion-formed blocks reaches 48.07%, versus 49.46% for forward completion order. These controls support a role for block membership, with no demonstrated unique benefit of forward order (Appendix B.1).

5 Response Growth and Normalization

5.1 Completion-derived strata and response growth

Completion grouping creates ordered length and reward strata early in training. In the two calibrated Qwen2.5 mathematical seeds, the four blocks first become jointly ordered by increasing mean length and decreasing token-weighted reward at updates 20 and 16. The first-to-last length ranges are 252 and 201 tokens, with no truncated response in any block. Both logic seeds show the same joint ordering by update 24, again before truncation. Random partitions remain near zero in the continuous block contrasts (Appendix Figure 5). The joint-order event is a descriptive screen; it is not a validated predictor of later failure.

These strata persist as responses grow. In the exploratory completion/local run's final pool, block mean length rises from 1,346 to 3,935 tokens and token-weighted reward falls from 0.694 to 0.141, against a pooled mean of 0.399. The last block has 55.73% truncation (Table 6). Local moments therefore supply markedly different reward references within the same fixed-policy rollout pool.

These observed moments give a concrete example of the centering effect. Before division by the standard deviation, a failed response in the last block has coefficient -0.141 under local centering and -0.399 under pooled centering. A successful response has coefficient 0.859 or 0.601, respectively. The terminal reward is the same in either case; the choice of reference changes how that

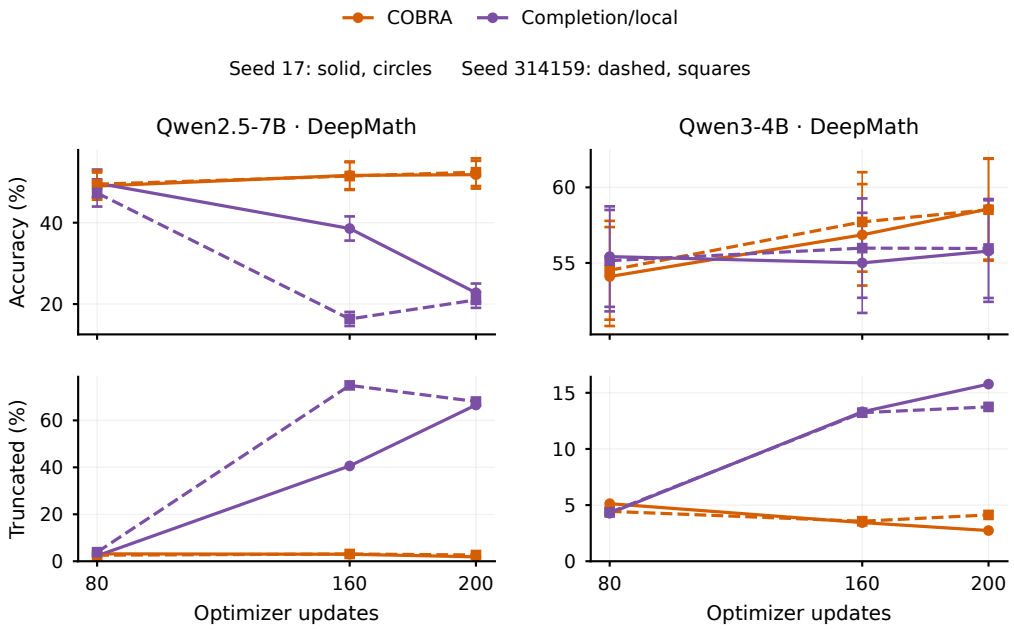


Figure 3: Switching to pooled reward moments limits response growth through update 200. Both models use completion-order blocks; scheduled arms switch after update 80 and controls retain local moments. Each line is one training seed. Both columns evaluate DeepMath, with different vertical scales. Accuracy bars are 1.96 prompt standard errors; truncation includes all evaluated responses. Qwen2.5 uses the shared format-calibrated start.

outcome contributes to the update. Scaling then changes the magnitudes of these coefficients, which motivates the separate center-by-scale intervention below.

The two-seed stability comparison is pronounced (Figure 3). At update 200, Qwen2.5 completion/local reaches 22.78%/21.02% DeepMath accuracy and 66.55%/68.04% truncation; COBRA reaches 51.81%/52.42% accuracy and 1.88%/2.66% truncation. Slash-separated values list seeds 17/314159. The corresponding MATH500 accuracies are 41.78%/33.45% for local normalization and 73.55%/72.62% for the schedule. The independently trained controls share the pre-switch design, with distinct stochastic trajectories.

Qwen3-4B shows the same response-growth contrast at a smaller magnitude. COBRA averages 868/964 DeepMath tokens and 2.73%/4.13% truncation, versus 2,076/1,709 tokens and 15.77%/13.75% for completion/local. Its accuracy gains over the local controls are 2.78 points (95% paired prompt interval $[1.02, 4.55]$) and 2.56 points ($[0.93, 4.20]$). MATH500 accuracy is similar between the two completion arms, while the schedule reduces response growth on both evaluation sets.

5.2 Local reward centering contributes to growth

An exploratory factorial in seed 42 varies local versus pooled reward means and standard deviations independently. Four continuations share the same calibrated completion/local update-80 parent and train to update 160 (Table 3). With pooled centering, DeepMath responses average 834 or 900 tokens under local or pooled scaling. With local centering, they average 2,125 or 3,002 tokens. Pooled centering also keeps truncation below 2.5% under either scale. The resumed local/local branch is the matched continuation control.

The intervention supports local centering as a contributor to the tested growth, with scaling also affecting the outcome. It concerns one trained state and one seed; the two-seed uninterrupted schedule comparison supplies separate evidence across training runs. A seed-17 timing grid switches after

Center	Scale	DeepMath accuracy	Mean tokens	Truncated
Local	Local	48.36%	2,125	17.82%
Local	Pooled	36.40%	3,002	42.21%
Pooled	Local	51.86%	834	1.54%
Pooled	Pooled	50.76%	900	2.47%

Table 3: Exploratory centering-by-scaling intervention, seed 42. Four completion-grouped continuations share a format-calibrated Qwen2.5-7B update-80 parent and are evaluated on DeepMath at update 160, with eight draws per problem and a 4,096-token cap.

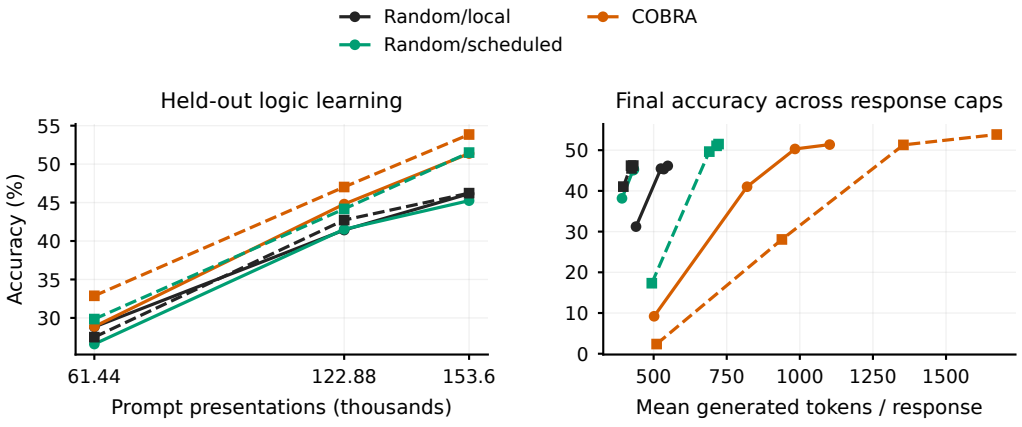


Figure 4: Held-out Knights and Knaves accuracy and response cost. Left: independent learning curves at a 4,096-token cap against prompt presentations, including repeated uses of the 4,479 training puzzles. Each optimizer update uses 768 presentations. Right: final accuracy against mean generated response tokens under caps of 512, 1,024, 2,048, and 4,096. Solid circles denote seed 17; dashed squares denote seed 314159. Completion grouping's final accuracy gains accompany longer responses and depend on the available response budget.

update 40, 80, or 120: final completion-grouped DeepMath accuracy spans 50.56–51.83%, with response length and MATH500 accuracy varying with timing (Appendix B.2).

Using a fixed schedule. The intervention separates the reward reference from its scale: pooled centering limits growth under either tested scale, while changing only the scale leaves long responses under local centering. The training recipe pools both moments after update 80 and continues to compute them from each newly completed rollout pool. This keeps the reference tied to the current policy's outcomes. The switch uses the round index and requires no additional verifier calls or problem presentations. The timing grid characterizes nearby fixed choices; the replicated main comparison uses the same update-80 rule in both seeds and in the scheduled random comparator.

6 Held-Out Logic Accuracy and Token Cost

After 200 updates and 153,600 prompt presentations, COBRA reaches 51.38%/53.85% held-out logic accuracy. Its paired gains over random/scheduled are 6.13 ± 1.02 and 2.35 ± 1.13 percentage points, and over random/local are 5.23 ± 1.10 and 7.62 ± 1.35 (Figure 4). The gain thus persists when random grouping receives the same normalization schedule.

The completion policies average 1,102/1,673 generated tokens, versus 433/722 for random/scheduled. Prompt-plus-response training costs are 176.13M/206.98M versus 110.07M/128.08M tokens. At a 2,048-token cap, COBRA reaches 50.30%/51.30%, compared with 45.43%/51.08% for random/scheduled. The 4,096-token gain therefore describes an accuracy–cost tradeoff on the trained task's held-out three-to-seven-person puzzles.

The result extends the recipe to a second verifiable task while making its response budget explicit. Figure 4 reports both training seeds and both random comparators across caps. Complete costs and response-cap evaluations are in Appendix C.

7 Related Work

Single-response and conditional advantages. REINFORCE and PPO supply the policy-gradient background (Williams, 1992; Schulman et al., 2017). Group Relative Policy Optimization (GRPO) compares several responses to the same prompt (Shao et al., 2024). REINFORCE++ uses global advantage normalization without a critic (Hu et al., 2025); Single-stream Policy Optimization combines global normalization with a persistent value tracker (Xu & Ding, 2025). Rethinking Groups in Critic-Free Reinforcement Learning with Verifiable Rewards studies negative-token filtering for single-rollout training (Wu et al., 2026). Shrinkage baselines combine within-prompt and across-prompt reward means (Zeng et al., 2025).

Conditional Advantage Estimation for Reinforcement Learning in Large Reasoning Models (CANON) constructs metric-dependent groups, including length and entropy groups, and schedules within-group and across-group advantage mixtures (Chen et al., 2025). Stratified GRPO groups search-agent trajectories by structural features, derives the local/global rescaling-plus-offset relationship, and blends the two advantages for stability (Zhu et al., 2025). These works establish conditional reward comparisons and local/global advantage mixtures. Our empirical setting combines completion-derived groups across prompts, one response per prompt, token-weighted moments, and sequential updates. Matched comparisons measure early learning and held-out logic accuracy, while normalization interventions characterize response growth.

Response length and loss reduction. Dr. GRPO analyzes response-length bias in policy-loss normalization (Liu et al., 2025). Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) motivates token-level reduction by the treatment of long, low-quality responses (Yu et al., 2025). We examine token-mean reduction when a retained rollout pool is divided into sequential updates. Unequal block token counts create unequal token weights.

Rollout scheduling. AReaL decouples generation and training while controlling staleness (Fu et al., 2025); AsyncFlow streams data between asynchronous producers and consumers (Han et al., 2025). RollPacker defers prompts with unfinished speculative rollouts to dedicated long rounds (Gao et al., 2025). Active Partial Rollouts in Reinforcement Learning (APRIL) continues incomplete responses in later steps (Zhou et al., 2025); concurrent-policy rollout systems preserve trajectory consistency, data integrity, and bounded staleness (Hu et al., 2026). Our synchronous protocol completes each pool before learning and retains all trajectories, exposing the statistical effect of block construction within a fixed rollout policy.

8 Conclusion

COBRA improves early mathematical accuracy from the same labeled training problems in the calibrated Qwen2.5 setting. A fixed switch from local to pooled reward moments limits subsequent response growth through 200 updates on two models. The same recipe also improves held-out logic accuracy, with training and inference costs quantifying the accompanying resource tradeoff. Distinct labeled problems, repeated presentations, and generated tokens provide separate accounts of data use and generation cost. Starting-policy and longer-horizon comparisons appear in Appendices D.1 and B.3.

Reproducibility Statement

The appendix specifies starting models, data splits, batch construction, normalization, the actor objective, rollout and evaluation settings, and uncertainty calculations. The manuscript archive contains processed evaluation and cost data and code that regenerates figures and tables. The separate `cobra_reproducibility.zip` supplement provides selected frozen trainer and response-buffer sources, a resolved representative configuration, prompt-template and selected task-preparation and scoring sources, public model revisions, prepared-split hashes, dependency versions, and content hashes. These materials support implementation inspection; mathematical data preparation requires the identified prepared inputs. Model weights, prepared datasets, and complete sampled-response traces are outside the supplied archives. Independent scoring validates the reported measurements; the appendix records unavailable intermediate checkpoints.

AI Use Statement

Generative AI tools assist with experimental planning, implementation, experiment launch and monitoring, data analysis, literature search, and manuscript preparation. Automated checks verify dataset and checkpoint identities, evaluation coverage, score aggregation, and figure sources. The authors are responsible for the scientific claims, interpretation, references, and final submitted content. AI tools are not listed as authors.

References

Guanxu Chen, Yafu Li, Yuxian Jiang, Chen Qian, Qihan Ren, Jingyi Yang, Yu Cheng, Dongrui Liu, and Jing Shao. Conditional advantage estimation for reinforcement learning in large reasoning models, 2025. URL https://arxiv.org/abs/2509.23962.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, et al. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning, 2025. URL https://arxiv.org/abs/2505.24298.

Wei Gao, Yuheng Zhao, Dakai An, Tianyuan Wu, Lunxi Cao, Shaopan Xiong, et al. RollPacker: Mitigating long-tail rollouts for fast, synchronous RL post-training, 2025. URL https://arxiv.org/abs/2509.21009.

Zhenyu Han, Ansheng You, Haibo Wang, Kui Luo, Guang Yang, Wenqi Shi, et al. AsyncFlow: An asynchronous streaming RL framework for efficient LLM post-training, 2025. URL https://arxiv.org/abs/2507.01663.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, et al. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025. URL https://arxiv.org/abs/2504.11456.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, 2021. URL https://arxiv.org/abs/2103.03874.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL https://arxiv.org/abs/2501.03262.

Tianhao Hu, Xiangcheng Liu, Yuchun Miao, Youshao Xiao, Hongyu Zang, Yang Zheng, et al. DORA: A scalable asynchronous reinforcement learning system for language model training, 2026. URL https://arxiv.org/abs/2604.26256.

K-and-K. Knights and knaves dataset. Hugging Face dataset, 2024. URL https://huggingface.co/datasets/K-and-K/knights-and-knaves.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992. doi: 10.1007/BF00992696.

Yihong Wu, Liheng Ma, Lingfeng Xiao, Muzhi Li, Xinyu Wang, Yingxue Zhang, and Jian-Yun Nie. Rethinking groups in critic-free RLVR, 2026. URL https://arxiv.org/abs/2606.17250.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning, 2024. URL https://arxiv.org/abs/2410.23123.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, et al. Logic-RL: Unleashing LLM reasoning with rule-based reinforcement learning, 2025. URL https://arxiv.org/abs/2502.14768.

Zhongwen Xu and Zihan Ding. Single-stream policy optimization, 2025. URL https://arxiv.org/abs/2509.13232.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Guanning Zeng, Zhaoyi Zhou, Daman Arora, and Andrea Zanette. Shrinking the variance: Shrinkage baselines for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2511.03710.

Yuzhen Zhou, Jiajun Li, Yusheng Su, Gowtham Ramesh, Zilin Zhu, Xiang Long, et al. APRIL: Active partial rollouts in reinforcement learning to tame long-tail generation, 2025. URL https://arxiv.org/abs/2509.18521.

Mingkang Zhu, Xi Chen, Bei Yu, Hengshuang Zhao, and Jiaya Jia. Stratified GRPO: Handling structural heterogeneity in reinforcement learning of LLM search agents, 2025. URL https://arxiv.org/abs/2510.06214.

A Training and Evaluation Details

A.1 Training algorithm

Algorithm 1 COBRA with local-to-pooled reward moments

Inputs: Initial policy parameters θ; verifiable training problems D; pool size $N = 3{,}072$; blocks $K = 4$; rounds $J = 50$; switch round $j_\star = 20$; standard-deviation floor $\epsilon = 0.05$.

for round $j = 1, \ldots, J$ **do**
  Set the rollout and fixed old policy to $\pi_{\theta_{\text{old}}} = \pi_\theta$.
  Draw N problem presentations from D, with replacement.
  Generate one response per presentation under $\pi_{\theta_{\text{old}}}$.
  Record reward r_i, valid response length L_i, and completion rank c_i.
  Finish and retain all N responses before updating the actor.
  Sort the complete pool P by c_i; split it into K equal blocks $B_1, \ldots, B_K$.
  for block $b = 1, \ldots, K$, in order, **do**
    Set $S = B_b$ if $j \le j_\star$; otherwise set $S = P$.
    Compute token-weighted reward moments (μ_S, s_S) using Eq. (1).
    Set $A_i = (r_i - \mu_S)/s_S$ for every response $i \in B_b$.
    Update θ once using Eq. (5), with these advantages and fixed old-policy probabilities.
  end for
end for; **return** π_θ after $KJ = 200$ optimizer updates.

Each block contains 768 problem presentations. Capped responses receive zero reward and remain in both the reward moments and token loss.

A.2 Complete actor objective

Let π_j denote the actor at the start of round j, and let π_j^{beh} denote the rollout behavior policy. Both their token log probabilities remain fixed during the round's four actor updates. For token t of response i, define

$$q_{it}(\theta) = \exp(\text{clip}(\log \pi_\theta(y_{it} \mid x_i, y_{i,<t}) - \log \pi_j(y_{it} \mid x_i, y_{i,<t}), -20, 20)), \tag{6}$$

$$w_{it} = \text{stopgrad}\left[\min\left(2, \frac{\pi_j(y_{it} \mid x_i, y_{i,<t})}{\pi_j^{\text{beh}}(y_{it} \mid x_i, y_{i,<t})}\right)\right], \tag{7}$$

$$g_{it} = \max(-A_i q_{it}, -A_i \,\text{clip}(q_{it}, 0.8, 1.28)), \tag{8}$$

$$\ell_{it} = w_{it} \begin{cases} \min(g_{it}, -10A_i), & A_i < 0, \\ g_{it}, & A_i \ge 0. \end{cases} \tag{9}$$

The block loss is $\mathcal{L}_{B_b} = D_b^{-1} \sum_{i \in B_b} \sum_{t=1}^{L_i} \ell_{it}$, with $D_b = T_{B_b}$ for the block reduction and $D_b = T_P/K$ for the common-denominator intervention. The same valid-token mask is used for return normalization and the token-mean actor loss. Statistics are computed over the selected block or entire pool before device distribution. Dynamic packing does not create additional normalization groups or optimizer updates. At a fixed actor parameter vector, averaging the K block losses weights a token in block B_b by $1/(KT_{B_b})$; a token-mean loss over the full pool assigns weight $1/T_P$. Sequential optimizer steps also evaluate these losses at different parameter vectors. The log-probability difference clamp, asymmetric clipping interval, dual clipping for negative advantages, and rollout correction are common to all compared arms.

Completion rank is the response buffer's monotonic counter assigned when a completed prompt group enters the ready set. In this one-response setting, the counter orders completed responses received by the buffer; harvesting sorts by that counter. The vLLM rollout engine admits up to 1,024 concurrent sequences and 8,192 tokens per engine batch, with chunked prefill enabled. The separate implementation supplement contains the counter, harvesting, and rollout configuration sources.

A.3 Starting models and format calibration

The Qwen2.5-7B source is the official base model at revision `d149729398750b98c0af14eb82c78cfe92750796`. A common 30-update calibration rewards a nonempty reasoning segment, its closing `</think>` tag, an `<answer>` segment

Setting	Value
Prompts per rollout pool	3,072; one response per prompt draw
Update blocks per pool	Four blocks of 768 trajectories
Training horizon	50 pools; 200 optimizer updates
Normalization switch	After pool 20; after update 80
Training response / prompt cap	4,096 / 2,048 tokens
Training temperature / top-p	1.0 / 1.0
Optimizer	Megatron Adam, $\beta = (0.9, 0.999)$
Learning rate / weight decay	10^{-6} / 0.1
Learning-rate schedule	Constant after 10 optimizer warmup updates
Gradient-norm clipping	1.0
Actor epochs per block	One, without block-internal reshuffling
Entropy / reference-divergence coefficient	Zero / zero
Actor / rollout tensor parallelism	2 / 1
Dynamic packing limit	8,192 tokens per GPU microbatch
Qwen2.5 / Qwen3 training allocation	Eight / four H20 GPUs per arm
Independent evaluation	Eight responses per prompt
Evaluation temperature / top-p	0.6 / 0.95
Main evaluation response cap	4,096 tokens

Table 4: Shared mathematical and logic training settings. GPUs are graphics processing units.

containing a balanced, nonempty boxed answer, and terminal `</answer>`. The chat template prefills `<think>`. Calibration has no correctness reward. It consumes 15,360 sampled responses, 6,339,644 generated response tokens, and 9,827,756 prompt-plus-response tokens. Task comparisons share this fixed overhead; practical cost for preparing a new model includes it once.

Calibration uses DeepMath prompts with answer labels removed. After deduplication and exact prompt filtering against DeepMath development and MATH500, a NumPy permutation with seed 20260921 reserves 512 prompts for validation and 512 for audit; the remaining 99,447 form the training split. The calibration optimizer uses batch-normalized REINFORCE++ advantages (Hu et al., 2025), with 64 prompts and eight responses per prompt in each update. Training temperature and top-p are both 1.0. Adam uses learning rate 10^{-6}, constant after five warmup updates, and training seed 42. Weight decay, entropy regularization, and reference-divergence penalties are zero. Prompt and response caps are 2,048 and 4,096 tokens; actor and rollout tensor parallelism are two and one, respectively, on eight H20 GPUs. The retained starting policy is the update-30 checkpoint.

Starting policy	DeepMath development	MATH500
Official Qwen2.5-7B	28.61%	53.40%
Format-calibrated Qwen2.5-7B	30.79%	35.50%
Official Qwen3-4B-Base	41.41%	52.35%

Table 5: Common-start answer accuracy under each model's independent eight-draw evaluation protocol, with a 4,096-token cap.

Calibration changes the answer-production policy and lowers scored MATH500 accuracy. The mathematical verifier accepts recognized correct answers even without the complete calibrated wrapper; these scores jointly measure answer production, extraction, and termination. All calibrated-start Qwen2.5 arms use the same calibrated weights and fresh optimizer, scheduler, and prompt-stream states. The Qwen3 source is official Qwen3-4B-Base at revision `906bfd4b4dc7f14ee4320094d8b41684abff8539`; it receives no format calibration.

A.4 Data, scorers, and statistical units

Prepared mathematical training data contain 99,447 DeepMath prompts. Exact normalized prompt-overlap filters separate training and evaluation files. The primary development set has 512 problems; the secondary MATH500 set contains all 500 problems. MATH500 is used during recipe development and training monitoring. Exact text filtering does not address unknown pretraining exposure.

Knights and Knaves data use revision `2f68547989981b1af37cb3dde5fdefa847aa8619` of `K-and-K/knights-and-knaves` (K-and-K, 2024). From each three-to-seven-person training file, a fixed seed reserves 100 puzzles for development. Canonical statement trees are compared under person renaming to remove shared logical identities. The resulting training, development, and test counts are 4,479, 499, and 500; the test set contains 100 puzzles per size. The logic scorer requires one `<answer>` block with a complete exact assignment. Duplicate, missing, unknown, or contradictory names and roles receive zero. Format validity and parsing errors remain diagnostics. A capped response scores zero on both mathematical and logic tasks.

For m prompts and $D = 8$ draws, let $z_{id} \in \{0, 1\}$ be correctness. The accuracy estimate is $\hat{a} = m^{-1} \sum_i \bar{z}_i$, with $\bar{z}_i = D^{-1} \sum_d z_{id}$. For two arms, let $\delta_i = \bar{z}_i^{(1)} - \bar{z}_i^{(0)}$. The paired difference is $\bar{\delta}$ and its prompt standard error is

$$\mathrm{SE}(\bar{\delta}) = \sqrt{\frac{\sum_{i=1}^{m} (\delta_i - \bar{\delta})^2}{m(m-1)}}. \tag{10}$$

Normal intervals use $\bar{\delta} \pm 1.96\,\mathrm{SE}$. This accuracy averages sampled correctness; it is not success from choosing the best of eight answers. Per-response seeds match within each paired comparison. Qwen2.5 mathematical evaluations use base sampling seed 20260923; logic and Qwen3 evaluations use 20260925. Paired prompt bootstrap intervals for the external evaluation resample whole prompts, preserving their draws and seed pairing. Two training seeds are shown separately or averaged as a fixed pair; prompt intervals do not estimate population-level variation over training seeds.

B Mechanism and Short Controls

Block	Mean length	Sequence acc.	Token reward	Tokens
1	1,346	70.18%	0.694	1,033,714
2	2,156	60.81%	0.606	1,656,069
3	3,045	46.09%	0.457	2,338,441
4	3,935	14.97%	0.141	3,021,967
Pooled	2,621	48.01%	0.399	8,050,191

Table 6: The final exploratory completion/local rollout at update 160. Each block contains 768 responses. Sequence accuracy averages rewards over responses; the token-weighted reward mean weights each response by its valid length. The pooled mean is 0.39924 over 8,050,191 valid tokens.

The exploratory factorial experiment uses seed 42 and 160 optimizer updates. Table 7 gives all four cells at both selected endpoints. This experiment motivates the fixed update-80 switch; seeds 17 and 314159 provide the subsequent replication.

Partition / normalization	DeepMath 80	DeepMath 160	MATH500 80	MATH500 160
Random / pooled	46.51	50.32	65.65	70.80
Completion / pooled	45.07	49.71	65.30	69.25
Random / local	46.48	51.49	64.95	72.28
Completion / local	49.68	34.62	72.53	55.05

Table 7: Exploratory factorial accuracies (%), with eight draws per problem and the 4,096-token response cap.

The centering/scaling continuations in Table 3 have corresponding MATH500 accuracy / mean tokens / truncation as follows: local/local, 72.33% / 1,398 / 10.68%; local/pooled, 61.15% / 2,220 / 29.90%; pooled/local, 73.48% / 594 / 1.18%; pooled/pooled, 74.30% / 605 / 1.38%. The resumed local/local branch differs from the uninterrupted factorial trajectory. The factorized comparison uses the four resumed branches, while the main stability result uses uninterrupted training runs.

In the failing factorial run, the final four blocks have correctness 70.18%, 60.81%, 46.09%, and 14.97%, with mean response lengths 1,346, 2,156, 3,045, and 3,935. Completion rank has Spearman correlation 0.987 with realized length, -0.409 with reward, and 0.060 with logged dataset

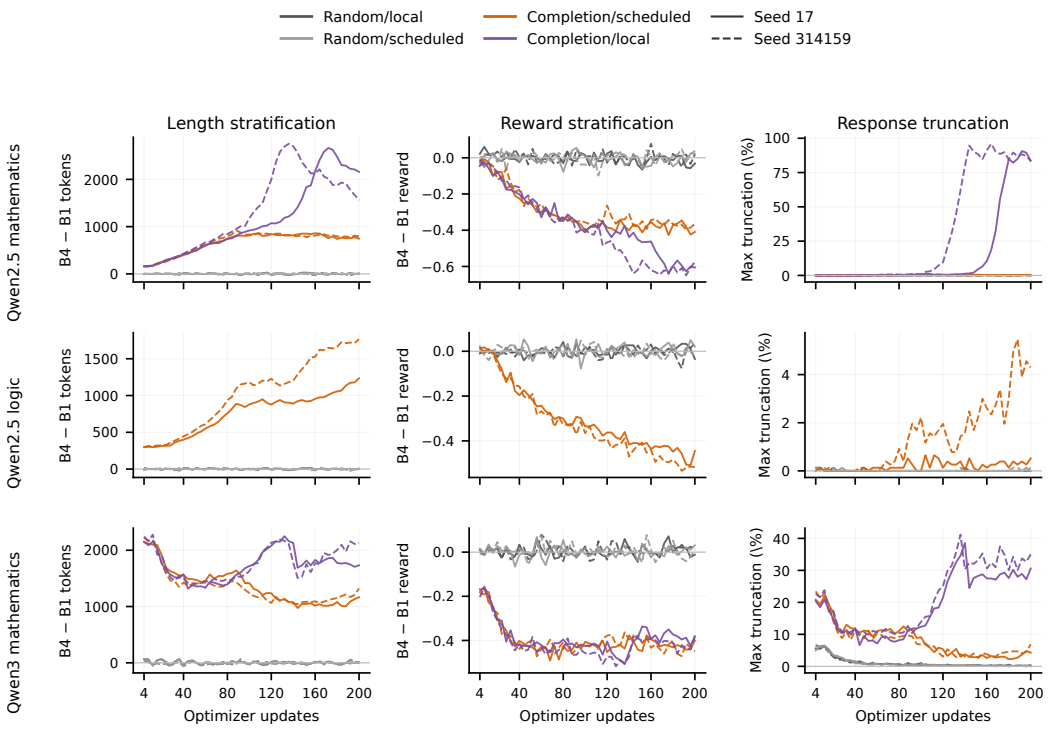


Figure 5: Training-pool block contrasts over 200 optimizer updates. Columns show block-4 minus block-1 mean response length, the corresponding difference in token-weighted reward mean, and the maximum truncation fraction among four blocks. Lines retain both seeds and all available arms. Completion ordering creates sustained length and reward strata; uninterrupted local normalization shows the sharpest later truncation. These sampled-rollout diagnostics are descriptive and do not make the onset rule a causal predictor.

difficulty. The between-block component of sequence-reward variance is 0.0438, compared with approximately 0.0002 under random partitioning. No recorded 768-trajectory block is entirely correct or entirely incorrect. Final block gradient norms are 0.181, 0.156, 0.125, and 0.107, below the clipping threshold of one. The observed degeneration consists of response growth, truncation, and lower all-prompt accuracy.

B.1 Length sorting and block consumption order

Two 80-update controls use seed 314159 and the same mathematical recipe. One sorts the complete pool by realized valid response length before forming blocks. The other forms blocks by completion rank and randomly permutes their consumption order each round. Both use local moments throughout their 80-update horizon. Constant learning rate, warmup, response cap, and batch size match the main early comparison.

Construction	DeepMath	Δ vs. completion	MATH500	Δ vs. completion
Forward completion	49.46	—	71.73	—
Length sorting	48.97	-0.49 ± 0.75	71.03	-0.70 ± 0.62
Random block order	48.07	-1.39 ± 0.75	71.23	-0.50 ± 0.69

Table 8: Short controls at update 80. Accuracy and paired differences are percentage points; uncertainty is one paired prompt standard error.

Length sorting produces mean responses of 926/641 tokens on DeepMath/MATH500; random block order produces 825/558. Corresponding truncation rates are 2.6%/2.0% and 2.0%/1.1%. These single-seed controls show similar early behavior from length-based membership and quantify the smaller difference associated with forward block order.

B.2 Normalization-switch timing

A seed-17 sensitivity grid applies random or completion-order blocks with the same calibrated Qwen2.5-7B starting policy and switches from local to pooled normalization after update 40, 80, or 120. Each of the six arms trains for 200 updates on 153,600 responses. The update-80 arms are the main seed-17 runs; the other four arms use the same data, scorer, training horizon, and independent eight-draw evaluation protocol. The update-80 timing remains the fixed main recipe.

Partition	Switch	Set	Accuracy	Length	Trunc.	Train resp.	Train total
Random	40	DeepMath	52.69	608	0.81	56.67	91.34
Random	40	MATH500	72.17	457	0.90	56.67	91.34
Random	80	DeepMath	53.47	622	1.34	58.73	93.41
Random	80	MATH500	73.12	466	0.90	58.73	93.41
Random	120	DeepMath	51.73	681	1.61	61.51	96.19
Random	120	MATH500	72.05	502	1.27	61.51	96.19
Completion	40	DeepMath	50.56	591	2.03	60.11	94.78
Completion	40	MATH500	70.10	470	1.85	60.11	94.78
Completion	80	DeepMath	51.81	796	1.88	94.93	129.61
Completion	80	MATH500	73.55	571	1.47	94.93	129.61
Completion	120	DeepMath	51.83	1184	5.71	120.12	154.80
Completion	120	MATH500	75.38	788	3.57	120.12	154.80

Table 9: Complete seed-17 switch-timing endpoints at update 200. Accuracy and truncation are percentages; length is mean generated response tokens. Training response and prompt-plus-response totals count tokens once per rollout pool, in millions, and are repeated across the two evaluation sets for each trained arm. All evaluations use a 4,096-token response cap.

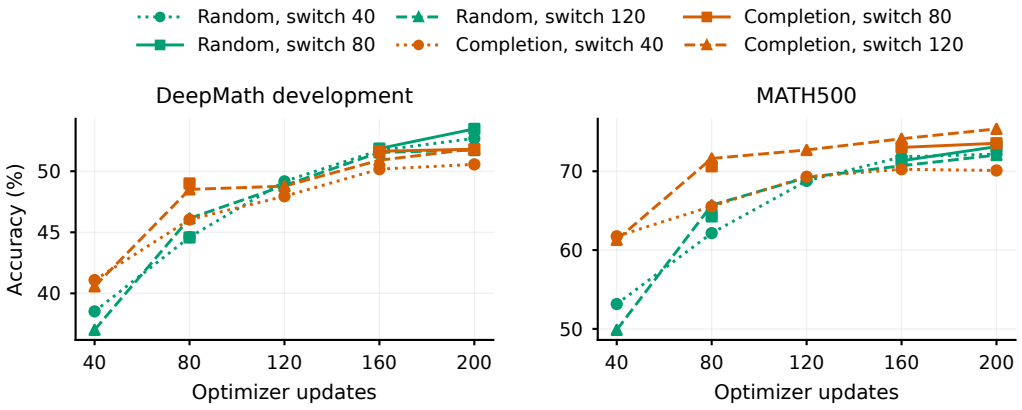


Figure 6: Available independent switch-timing learning curves for seed 17. Lines connect adjacent scored checkpoints only. The original update-80 arms have no retained weights at updates 40 and 120, leaving eight unavailable dataset cells in the 60-cell grid; all 12 final cells are scored.

At update 200, completion-grouped DeepMath accuracy lies between 50.56% and 51.83% across the three switch times. The switch-after-120 completion arm averages 1,184 generated response tokens on DeepMath, compared with 591 after switch 40 and 796 after switch 80. Its MATH500 accuracy is 75.38%, versus 70.10% and 73.55% for switches 40 and 80. Generated training response tokens increase from 60.11M at switch 40 to 94.93M at switch 80 and 120.12M at switch 120.

B.3 CONTINUED TRAINING THROUGH UPDATE 400

The longer-horizon comparison resumes the six Qwen2.5-7B random/local, random/scheduled, and completion/scheduled runs from their update-200 checkpoints and continues them to updates 300 and 400. Resumption restores the full actor and optimizer state, constant learning-rate scheduler, random state, and saved prompt-stream state. The cumulative training costs below include the first 200 updates. Independent evaluation keeps the original DeepMath development and MATH500 prompts, eight draws, sampling seed, scorer, and 4,096-token response cap. The comparison contains all 36 evaluation cells (three arms, two seeds, three checkpoints, and two datasets) and 36 direct prompt-paired arm contrasts. It does not contain a completion/local baseline beyond update 200.

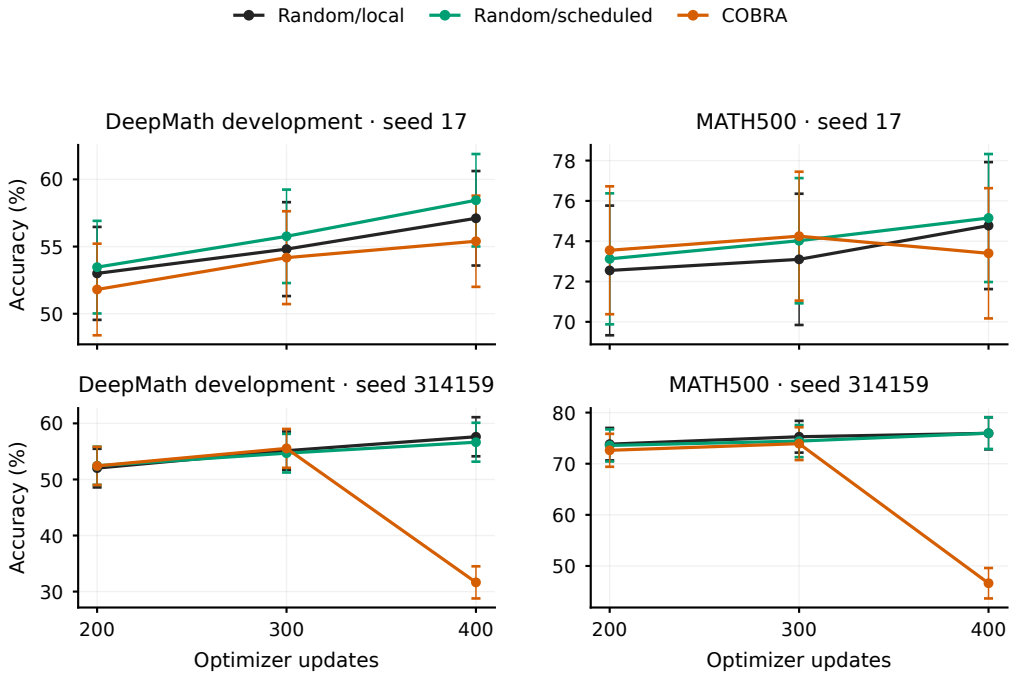


Figure 7: Independent accuracy through update 400 for both seeds, all three continued arms, and both mathematical datasets. Error bars are 1.96 prompt-level standard errors. The late completion/scheduled degradation is seed dependent.

At update 400, seed 17 completion/scheduled remains at 55.40% DeepMath and 73.40% MATH500 accuracy, with 5.03% and 3.23% truncation. Seed 314159 instead falls from 55.54% to 31.64% on DeepMath between updates 300 and 400, and from 73.92% to 46.62% on MATH500; truncation reaches 52.25% and 44.45%. Its random/local endpoints are 57.62% and 75.95%, while its random/scheduled endpoints are 56.64% and 75.98%. The corresponding completion/scheduled cumulative cost is 218.89M generated response tokens and 288.34M prompt-plus-response tokens. Seed 17 completion/scheduled uses 207.24M and 276.63M tokens, compared with 142.74M/212.14M for random/local and 154.26M/223.66M for random/scheduled. Thus the update-200 stability result does not imply stability under a longer horizon; the observed late failure is specific to one of the two continued seeds.

Table 10: Continued-training endpoints at updates 300 and 400. Accuracy and truncation are percentages; length is mean generated response tokens. Cumulative response and prompt-plus-response training tokens include the first 200 updates and are in millions.

Arm	Seed	Accuracy	Length	Trunc.	Train resp.	Train total
DeepMath development; update 300						
Random/local	17	54.81	639	0.68	100.41	152.46
Random/local	314159	55.13	877	1.27	127.46	179.53
Random/scheduled	17	55.76	715	1.15	102.57	154.62
Random/scheduled	314159	54.69	888	1.22	120.41	172.48
COBRA	17	54.17	895	4.47	150.22	202.26
COBRA	314159	55.54	812	2.17	150.82	202.90
MATH500; update 300						
Random/local	17	73.10	458	0.50	100.41	152.46
Random/local	314159	75.28	598	0.75	127.46	179.53
Random/scheduled	17	74.02	501	0.62	102.57	154.62
Random/scheduled	314159	74.42	606	0.75	120.41	172.48
COBRA	17	74.25	659	3.55	150.22	202.26
COBRA	314159	73.92	641	2.35	150.82	202.90
DeepMath development; update 400						
Random/local	17	57.10	640	0.71	142.74	212.14
Random/local	314159	57.62	902	2.05	187.22	256.67
Random/scheduled	17	58.45	800	0.63	154.26	223.66
Random/scheduled	314159	56.64	920	1.64	180.69	250.14
COBRA	17	55.40	864	5.03	207.24	276.63
COBRA	314159	31.64	2503	52.25	218.89	288.34
MATH500; update 400						
Random/local	17	74.78	469	0.43	142.74	212.14
Random/local	314159	75.95	601	1.18	187.22	256.67
Random/scheduled	17	75.15	567	0.85	154.26	223.66
Random/scheduled	314159	75.98	630	1.20	180.69	250.14
COBRA	17	73.40	612	3.23	207.24	276.63
COBRA	314159	46.62	2163	44.45	218.89	288.34

C Data Exposure and Token Cost

Prompt presentations count uses of verifiable training problems. Each presentation generates one response, and each optimizer update consumes 768 presentations. Sampling is with replacement, so repeated presentations can reuse a labeled problem. The mathematical dataset contains 99,447 distinct prompt identifiers, with a verifiable answer for each problem. Retained rollout identifiers map dataset row indices to those identifiers. Distinct-problem counts accumulate the identifiers encountered during task training from the common calibrated initialization. All arms have access to the full dataset throughout.

Across the four main mathematical arms and two seeds, the retained records contain 400 complete pools of 3,072 presentations. Every pool has the same problem multiset across arms within its seed. The numerical artifact `checkpoint_problem_counts.csv` records cumulative presentations, distinct problems, and repeated presentations at every 20-update checkpoint.

Generated training response tokens sum the valid response lengths over all rollouts. Prompt-plus-response training tokens count the input and output tokens once per pool. Neither quantity counts every actor forward/backward operation. Logged component timers also exclude some loading, exporting, and evaluation work. Inference cost averages generated tokens across all responses, including incorrect and truncated outputs. The tables keep these quantities separate.

Tables 11 and 12 report every declared accuracy target for completion/scheduled against both random comparators. DeepMath targets are 45%, 48%, and 50%; MATH500 targets are 65%, 70%, and 72%. Entries describe first observed attainment on the independent checkpoint grid. Each entry pairs the first evaluated checkpoint whose estimated accuracy meets the target with the preceding evalu-

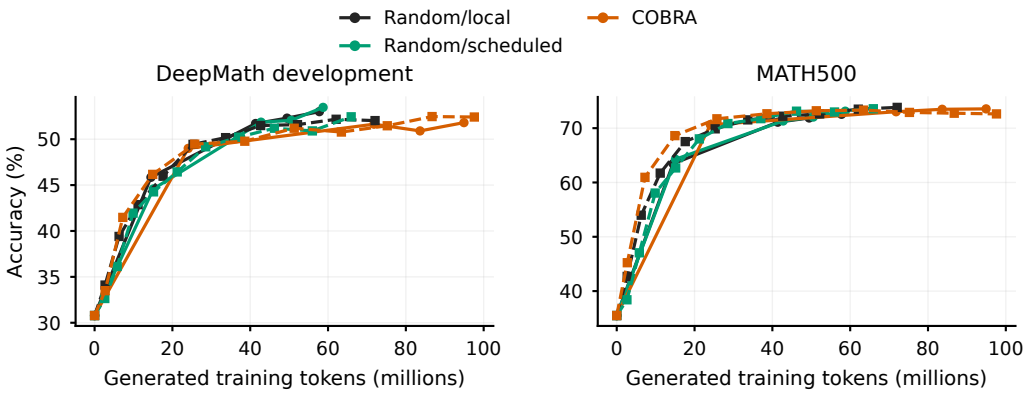


Figure 8: Generated-token accounting for mathematical training. Both seeds and both random controls are retained; solid circles denote seed 17 and dashed squares denote seed 314159. The horizontal axis counts generated training tokens, including every sampled response, and excludes the shared format-calibration stage. This cost depends on response length as well as problem presentations. Points are independent evaluations; lines do not identify unobserved target crossings.

ated checkpoint below it. These observations do not bound the true first-crossing cost: intermediate policies are unobserved and accuracy can be nonmonotone.

Dataset	Target	Seed	Update span C / R	Distinct problems C / R	Response-token span C / R (M)
DeepMath	45%	17	0–80 / 0–80	45,882 / 45,882	0.00–24.07 / 0.00–14.49
DeepMath	45%	314159	40–60 / 60–80	36,886 / 45,847	7.26–14.94 / 11.18–17.61
DeepMath	48%	17	0–80 / 80–160	45,882 / 70,591	0.00–24.07 / 14.49–41.38
DeepMath	48%	314159	60–80 / 80–100	45,847 / 53,470	14.94–25.74 / 17.61–25.30
DeepMath	50%	17	80–160 / 80–160	70,591 / 70,591	24.07–71.80 / 14.49–41.38
DeepMath	50%	314159	100–120 / 100–120	59,944 / 59,944	38.60–51.30 / 25.30–33.67
MATH500	65%	17	0–80 / 80–160	45,882 / 70,591	0.00–24.07 / 14.49–41.38
MATH500	65%	314159	40–60 / 60–80	36,886 / 45,847	7.26–14.94 / 11.18–17.61
MATH500	70%	17	0–80 / 80–160	45,882 / 70,591	0.00–24.07 / 14.49–41.38
MATH500	70%	314159	60–80 / 100–120	45,847 / 59,944	14.94–25.74 / 25.30–33.67
MATH500	72%	17	80–160 / 180–200	70,591 / 78,200	24.07–71.80 / 49.46–57.77
MATH500	72%	314159	80–100 / 120–140	53,470 / 65,594	25.74–38.60 / 33.67–42.68

Table 11: First independently observed target attainment. C denotes COBRA and R denotes random/local. Each update uses 768 prompt presentations. Distinct counts give the different training problems seen through the first observed above-target checkpoint. Token costs are millions of generated training response tokens. Each span joins the preceding observed below-target checkpoint to the first observed above-target checkpoint; it does not bound the first crossing. All listed targets are reached by these two arms.

The preserved seed-17 independent grid contains updates 80, 160, 180, and 200; earlier update-20/40/60 and update-100/120/140 weight exports are unavailable. Seed 314159 preserves every 20-update checkpoint. The full artifact target table also includes completion/pooled and the uninterrupted local control, with unreached targets retained. For example, completion/pooled in seed 17 does not reach DeepMath 50% or MATH500 70%/72% on its observed independent grid.

For the 70% MATH500 target, the completion checkpoints first above target have seen 45,882/45,847 distinct problems, versus 70,591/59,944 for either random comparator, reductions of 35.0%/23.5%. Their presentation counts are 61,440/61,440 versus 122,880/92,160. These checkpoints use 50.0%/33.3% fewer optimizer updates and 41.8%/23.6% fewer generated training tokens than random/local. Prompt-plus-response reductions at these observed checkpoints are 45.1%/27.3%. Against random/scheduled, the same update reductions accompany generated-token

Dataset	Target	Seed	Update span C / R	Distinct problems C / R	Response-token span C / R (M)
DeepMath	45%	17	0–80 / 80–160	45,882 / 70,591	0.00–24.07 / 15.09–42.78
DeepMath	45%	314159	40–60 / 80–100	36,886 / 53,470	7.26–14.94 / 15.15–21.24
DeepMath	48%	17	0–80 / 80–160	45,882 / 70,591	0.00–24.07 / 15.09–42.78
DeepMath	48%	314159	60–80 / 100–120	45,847 / 59,944	14.94–25.74 / 21.24–28.56
DeepMath	50%	17	80–160 / 80–160	70,591 / 70,591	24.07–71.80 / 15.09–42.78
DeepMath	50%	314159	100–120 / 120–140	59,944 / 65,594	38.60–51.30 / 28.56–37.03
MATH500	65%	17	0–80 / 80–160	45,882 / 70,591	0.00–24.07 / 15.09–42.78
MATH500	65%	314159	40–60 / 80–100	36,886 / 53,470	7.26–14.94 / 15.15–21.24
MATH500	70%	17	0–80 / 80–160	45,882 / 70,591	0.00–24.07 / 15.09–42.78
MATH500	70%	314159	60–80 / 100–120	45,847 / 59,944	14.94–25.74 / 21.24–28.56
MATH500	72%	17	80–160 / 160–180	70,591 / 74,664	24.07–71.80 / 42.78–50.52
MATH500	72%	314159	80–100 / 140–160	53,470 / 70,313	25.74–38.60 / 37.03–46.21

Table 12: First independently observed target attainment against the scheduled random comparator. C denotes COBRA and R denotes random/scheduled. Each update uses 768 prompt presentations. Distinct counts give the different training problems seen through the first observed above-target checkpoint. Token costs are millions of generated training response tokens. Each span joins the preceding observed below-target checkpoint to the first observed above-target checkpoint; it does not bound the first crossing. All listed targets are reached by both arms.

reductions of 43.7%/9.9%. The target concerns mean sampled correctness: completion's update-80 MATH500 accuracies are 70.625%/71.725%, with prompt standard errors 1.640/1.618 percentage points. For DeepMath 50%, completion and random/local first appear above target at the same updates, and completion uses 73.5%/52.3% more generated tokens. At MATH500 72%, seed-17 completion first appears above target at update 160 versus 200 for random/local, but uses 24.3% more generated tokens. Token costs depend on the accuracy target and response lengths.

At the seed-314159 completion/scheduled update-80 response budget of 25.74M tokens, random/local's latest evaluated checkpoint within budget is update 100, at 25.30M tokens. Its DeepMath accuracy is 49.44%, compared with 49.46% for completion/scheduled at update 80. Thus matching token cost can produce a different comparison from matching updates. The complete artifact table supplies these shared-budget comparisons for both datasets and both training-cost definitions.

The mathematical response-cap matrix contains 128 cells: completion/scheduled and random/local at updates 80, 160, and 200, and random/scheduled at updates 80 and 200, each across two seeds, two datasets, and four caps (512, 1,024, 2,048, and 4,096). The logic matrix contains 67 cells: the common start and all six trained arms at updates 80, 160, and 200 under cap 4,096, plus the three shorter caps for completion/scheduled and random/local at all three endpoints and for random/scheduled at 80 and 200. Every cap uses fresh generation under that stopping rule.

Table 13: Complete mathematical response-cap accuracy matrix (%). Each cap uses eight fresh draws per problem.

Arm	Seed	Cap 512	Cap 1,024	Cap 2,048	Cap 4,096
DeepMath development; update 80					
Random/local	17	38.31	44.95	45.21	45.85
Random/local	314159	29.64	45.12	47.14	45.97
Random/scheduled	17	35.89	44.65	44.60	44.58
Random/scheduled	314159	34.64	44.14	44.09	44.26
COBRA	17	13.62	43.48	48.93	49.00
COBRA	314159	9.28	39.50	49.32	49.46
DeepMath development; update 160					
Random/local	17	32.13	50.12	51.27	51.71
Random/local	314159	20.73	46.68	51.76	51.59
COBRA	17	11.94	44.78	50.93	51.64
COBRA	314159	14.11	43.14	50.46	51.46

Arm	Seed	Cap 512	Cap 1,024	Cap 2,048	Cap 4,096
DeepMath development; update 200					
Random/local	17	30.03	50.61	52.78	53.00
Random/local	314159	17.92	47.17	52.93	52.03
Random/scheduled	17	29.69	50.56	52.98	53.47
Random/scheduled	314159	19.12	46.34	52.25	52.44
COBRA	17	17.58	46.90	51.88	51.81
COBRA	314159	18.90	47.41	52.49	52.42
MATH500; update 80					
Random/local	17	60.55	63.20	63.98	63.45
Random/local	314159	58.88	67.03	67.97	67.53
Random/scheduled	17	60.15	64.70	64.48	64.28
Random/scheduled	314159	59.05	62.80	63.60	62.68
COBRA	17	51.18	69.20	70.73	70.62
COBRA	314159	45.62	68.53	71.43	71.73
MATH500; update 160					
Random/local	17	59.50	71.10	71.10	71.10
Random/local	314159	54.27	71.95	72.45	72.62
COBRA	17	49.50	70.93	72.78	73.02
COBRA	314159	48.65	70.28	74.05	72.90
MATH500; update 200					
Random/local	17	59.40	71.33	72.22	72.55
Random/local	314159	53.02	71.45	73.72	73.83
Random/scheduled	17	58.27	72.00	72.32	73.12
Random/scheduled	314159	55.15	71.53	73.22	73.58
COBRA	17	53.20	70.58	72.22	73.55
COBRA	314159	51.35	71.20	73.88	72.62

Table 14: Held-out logic accuracy (%) across evaluated caps. Dashes denote cap cells outside the evaluation matrix. The common starting policy scores 9.60% at cap 4,096.

Arm	Seed	Update	Cap 512	Cap 1,024	Cap 2,048	Cap 4,096
Random/local	17	80	24.73	28.27	28.43	28.82
Random/local	17	160	32.85	40.42	40.95	41.42
Random/local	17	200	31.23	45.48	45.35	46.15
Random/local	314159	80	23.75	27.07	27.45	27.50
Random/local	314159	160	38.45	41.83	41.83	42.73
Random/local	314159	200	41.05	46.20	45.52	46.23
Random/scheduled	17	80	22.57	26.15	26.30	26.60
Random/scheduled	17	160	—	—	—	41.52
Random/scheduled	17	200	38.17	45.07	45.42	45.25
Random/scheduled	314159	80	21.02	29.20	29.68	29.85
Random/scheduled	314159	160	—	—	—	44.17
Random/scheduled	314159	200	17.32	49.65	51.08	51.50
COBRA	17	80	3.55	22.62	29.10	28.88
COBRA	17	160	8.65	34.55	43.77	44.80
COBRA	17	200	9.20	41.02	50.30	51.38
COBRA	314159	80	2.57	21.60	31.82	32.88
COBRA	314159	160	2.25	24.27	44.85	47.02
COBRA	314159	200	2.38	28.07	51.30	53.85

Comparison	Seed 17	Seed 314159
Official-base completion/scheduled minus random/local, points [95% interval]		
DeepMath development, update 80	+0.39 [-1.17, +1.95]	+0.95 [-0.61, +2.54]
MATH500, update 80	+0.92 [-0.40, +2.25]	+0.83 [-0.55, +2.23]
DeepMath development, update 200	-0.81 [-2.44, +0.83]	-1.15 [-2.71, +0.42]
MATH500, update 200	+0.27 [-1.10, +1.65]	+0.57 [-0.70, +1.88]
Format-calibrated update-400 DeepMath, accuracy (truncation), %		
Random/local	57.10 (0.71)	57.62 (2.05)
Random/scheduled	58.45 (0.63)	56.64 (1.64)
COBRA	55.40 (5.03)	31.64 (52.25)

Table 15: Starting-policy and horizon comparisons. Top: official-base completion/scheduled minus random/local accuracy in percentage points, with paired prompt-bootstrap 95% intervals. Bottom: calibrated-start DeepMath accuracy at update 400, with truncation percentages in parentheses. Columns are fixed training seeds.

D Starting-Policy Comparison

D.1 Official-base start-policy comparison

A matched two-seed comparison trains random/local and completion/scheduled directly from the official Qwen2.5-7B base policy. It uses the same prompts, 80/160/200-update checkpoints, eight draws per prompt, and 4,096-token cap as the calibrated-start comparison. Before task training, official minus calibrated is -2.17 points on DeepMath, with paired prompt-bootstrap 95% interval $[-5.08, 0.68]$, and $+17.90$ points on MATH500, interval $[14.85, 21.00]$.

Arm	Seed	Set	Official	Calibrated	Difference
Random/local	17	DeepMath	56.62	53.00	+3.61
Random/local	17	MATH500	76.33	72.55	+3.77
Random/local	314159	DeepMath	56.03	52.03	+4.00
Random/local	314159	MATH500	75.60	73.83	+1.77
Completion/scheduled	17	DeepMath	55.81	51.81	+4.00
Completion/scheduled	17	MATH500	76.60	73.55	+3.05
Completion/scheduled	314159	DeepMath	54.88	52.42	+2.47
Completion/scheduled	314159	MATH500	76.17	72.62	+3.55

Table 16: Update-200 accuracy (%) from official-base and matched format-calibrated starts. Difference is official minus calibrated. Each cell averages eight draws per prompt.

Within official-start runs, completion/scheduled minus random/local at update 80 is $+0.39$ $[-1.17, 1.95]$ and $+0.95$ $[-0.61, 2.54]$ points on DeepMath, and $+0.92$ $[-0.40, 2.25]$ and $+0.83$ $[-0.55, 2.23]$ points on MATH500 for seeds 17 and 314159. At update 200, DeepMath differences are -0.81 $[-2.44, 0.83]$ and -1.15 $[-2.71, 0.42]$ points. Across all 24 trained arm, seed, checkpoint, and dataset combinations, official-start accuracy exceeds matched calibrated-start accuracy by 1.78–9.78 points, and every paired 95% interval lies above zero. These across-start differences measure policy initialization, not the batching algorithm.

Each update-200 task run samples 153,600 responses. Format calibration is a separate shared 30-update stage containing 15,360 responses, 6,339,644 generated response tokens, and 9,827,756 prompt-plus-response tokens. Endpoint data retain task-training response counts and both token costs in separate columns; calibration cost is not folded into task-training cost. Paired 20,000-draw bootstrap intervals resample prompts while retaining all eight matched draws and condition on the fixed training seeds.

E EARLY AND FINAL ENDPOINT TABLES

Model and evaluation set	Update	Random/local	Random/sched.	COBRA
Qwen2.5-7B: DeepMath	80	45.85 / 45.97	44.58 / 44.26	49.00 / 49.46
	200	53.00 / 52.03	53.47 / 52.44	51.81 / 52.42
Qwen2.5-7B: MATH500	80	63.45 / 67.53	64.28 / 62.68	70.62 / 71.73
	200	72.55 / 73.83	73.12 / 73.58	73.55 / 72.62
Qwen2.5-7B: logic	80	28.82 / 27.50	26.60 / 29.85	28.88 / 32.88
	200	46.15 / 46.23	45.25 / 51.50	51.38 / 53.85
Qwen3-4B: DeepMath	80	55.13 / 55.98	54.96 / 55.37	54.10 / 54.52
	200	59.08 / 58.23	57.52 / 58.81	58.57 / 58.52
Qwen3-4B: MATH500	80	76.22 / 76.08	75.88 / 76.00	75.33 / 76.12
	200	76.68 / 77.78	76.60 / 76.98	76.88 / 77.55

Table 17: Accuracy (%) for scheduled completion grouping and both random controls. Each cell shows seeds 17 / 314159. All cells use eight responses per prompt and a 4,096-token cap. Qwen2.5 uses the calibrated start; Qwen3 uses its official base. Logic contains 500 held-out three-to-seven-person puzzles.

Scheduled arms switch after update 80. Qwen3's update-80 DeepMath completion/scheduled-minus-random/local differences are -1.03/-1.46 percentage points. Final mathematical accuracy is similar between scheduled completion and random controls on both models. Table 18 supplies their intermediate stability endpoints.

Dataset	Arm	Seed	Accuracy	Length	Truncated
Qwen2.5-7B					
DeepMath	COBRA	17	51.64	901	2.98
DeepMath	Completion/local	17	38.57	2732	40.60
DeepMath	COBRA	314159	51.46	902	3.10
DeepMath	Completion/local	314159	16.33	3770	74.90
MATH500	COBRA	17	73.02	626	2.00
MATH500	Completion/local	17	63.20	1875	26.00
MATH500	COBRA	314159	72.90	635	1.90
MATH500	Completion/local	314159	32.02	3272	63.45
Qwen3-4B					
DeepMath	COBRA	17	56.86	910	3.44
DeepMath	Completion/local	17	55.00	1753	13.31
DeepMath	COBRA	314159	57.71	960	3.56
DeepMath	Completion/local	314159	55.98	1554	13.23
MATH500	COBRA	17	76.92	655	2.88
MATH500	Completion/local	17	75.80	1229	8.60
MATH500	COBRA	314159	77.28	658	2.23
MATH500	Completion/local	314159	75.92	1043	8.25

Table 18: Intermediate completion-grouping endpoints at update 160 for both models. Accuracy and truncation are percentages; length is mean generated response tokens. Both seeds and both normalization choices are evaluated with eight draws per problem and a 4,096-token cap.