BEYOND FORGETTING: DIAGNOSING AND HARNESSING SHARED REASONING IN CONTINUAL RLVR

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) commonly post-trains reasoning models on multiple tasks, while rerunning multitask RLVR (MTRL) as new tasks are added makes capability expansion costly. We therefore study continual RLVR, which updates the existing model as each task arrives. The central question is whether a model updated this way can perform as well as a jointly trained model. To answer this question, we introduce **Continual Reasoning Gym**, a continual-RLVR environment that organizes text and visual reasoning tasks into five task sequences. In this setting, we identify two key observations: Continual RLVR exhibits modest forgetting, yet its final performance remains below that of MTRL. To understand the latter, we decompose final performance and show that forgetting accounts for only part of the gap. To explain the former, we identify *shared reasoning*: transferable reasoning structure allows training on one task to support others on average. We therefore introduce **Continual Prompt Replay (CPR)**, which harnesses shared reasoning to improve learning on the arriving and future tasks by replaying previous-task prompts and regenerating their responses with the current policy. On average, only CPR reaches MTRL-level performance.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) commonly post-trains reasoning models on task mixtures fixed before training (Stojanovski et al., 2025; Xu et al., 2026; Xi et al., 2026; Cheng et al., 2025). A deployed coding agent, for example, may later need to adapt to newly released APIs or changing user preferences. Rerunning multitask RLVR (MTRL) over the enlarged task mixture for every new requirement is computationally costly. We therefore study continual RLVR, which updates the existing model as each task arrives.

The central question is whether a model updated this way can perform as well as a jointly trained model (Momeni & Liu, 2025). Catastrophic forgetting is the standard explanation for why sequential learning trails joint training (Kirkpatrick et al., 2017; Wang et al., 2023). Yet reinforcement learning has recently been found to forget less than supervised fine-tuning (SFT), suggesting that continual RLVR may likewise exhibit less forgetting (Shenfeld et al., 2026; Zhang et al., 2026).

To test whether this lower forgetting allows a continually updated model to match a jointly trained one, we introduce Continual Reasoning Gym (CRG), which organizes text and visual reasoning tasks into task sequences. On CRG, we identify two key observations: a sequential RLVR (Seq. RLVR) baseline exhibits modest forgetting, yet its final performance remains below MTRL.

To understand the latter, we derive an exact decomposition of final performance and show that forgetting accounts for only part of the gap. The remaining gap reflects how well the model learns each arriving task and how earlier training benefits future tasks. To explain the former, we identify *shared reasoning*: transferable reasoning structure allows training on one task to benefit others on average. We provide empirical evidence for this pattern through a task-gradient alignment analysis and a behavioral case study. We therefore introduce Continual Prompt Replay (CPR), which harnesses shared reasoning to improve learning on the arriving and future tasks by bringing previous-task prompts into current-policy training (Figure 1). CPR is the only continual learning (CL) method to reach MTRL-level performance on average.

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.

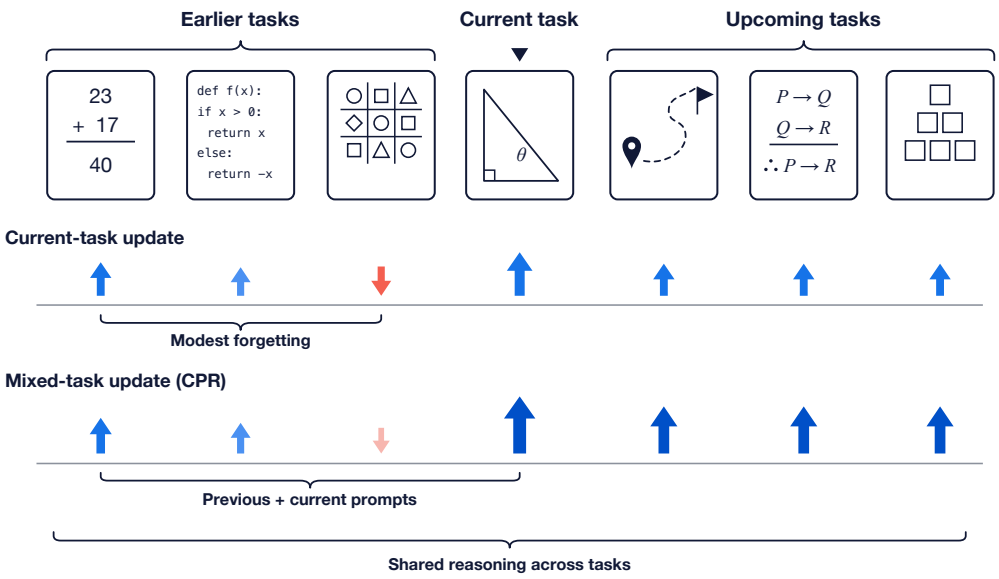


Figure 1: **Shared reasoning explains modest forgetting while CPR harnesses it to improve performance.** Later-task updates benefit earlier tasks on average, helping explain modest forgetting. CPR replaces some current-task prompts with previous-task prompts to harness shared reasoning and improve learning on the arriving and future tasks.

Our contributions are threefold:

1. **A continual-RLVR environment.** We introduce CRG with five task sequences spanning text and visual reasoning, together with implementations of diverse CL methods.
2. **A finding of shared reasoning.** We identify shared reasoning, whereby training on one task benefits others on average. It explains modest forgetting and can be harnessed to improve learning on the arriving and future tasks. We formalize this finding and provide empirical evidence from task-gradient measurements and a behavioral case study.
3. **A continual-RLVR method.** We introduce CPR, which harnesses shared reasoning by replaying previous-task prompts and regenerating their responses with the current policy. CPR is the only CL method to reach MTRL-level performance on average.

2 Related Work

Continual reinforcement-learning environments. Continual reinforcement-learning environments expose agents to ordered sequences of interactive tasks and evaluate how learning evolves across stages. Existing environments span robotic manipulation, embodied navigation, and cooperative multi-agent tasks (Wolczyk et al., 2021; Liu et al., 2023; Tomilin et al., 2023; 2026). CRG applies this staged protocol to reasoning models trained with RLVR, using programmatic verifiers to score generated responses.

CL benchmarks. CL benchmarks study sequential adaptation of large language models (LLMs), vision–language models (VLMs), and video–language models under stage-wise evaluation (Wang et al., 2023; Srinivasan et al., 2022; Tang et al., 2024). These benchmarks primarily use supervised updates on fixed input–target pairs. CRG instead evaluates stage-wise learning and transfer under continual RLVR, where rewards on policy-generated responses make both trajectories and training signals policy-dependent.

Table 1 summarizes these distinctions. Appendix E discusses additional related work.

Table 1: **Positioning CRG at the intersection of CL and RLVR.** CRG combines reasoning tasks, a task sequence, RLVR training, and multiple CL methods. ✓ = present, × = absent.

	Reasoning tasks	Task sequence	RLVR training	CL methods
TRACE (Wang et al., 2023)	✓	✓	×	✓
CLiMB (Srinivasan et al., 2022)	✓	✓	×	✓
Continual World (Wolczyk et al., 2021)	×	✓	×	✓
LIBERO (Liu et al., 2023)	×	✓	×	✓
Reasoning Gym (Stojanovski et al., 2025)	✓	×	✓	×
VisuLogic (Xu et al., 2026)	✓	×	✓	×
AgentGym-RL (Xi et al., 2026)	✓	×	✓	×
Continual Reasoning Gym (ours)	✓	✓	✓	✓

3 PRELIMINARIES

Continual RL. A continual RL problem chains an ordered sequence of tasks $\tau_1, \ldots, \tau_T$, each modeled as an MDP, that a single agent learns one at a time (Wolczyk et al., 2021). This paper studies the RLVR specialization, where each task carries a programmatic verifier and every training stage consists of RLVR updates. Starting from a base policy M_0, stage i trains on task τ_i and produces M_i from M_{i-1}, and the goal is a final policy with the highest average success across the T tasks.

Token-level Markov Decision Process. Task τ_i provides a prompt distribution $\mathcal{D}_i$ and a programmatic verifier $r_i(x, y) \in \{0, 1\}$ for a prompt $x \sim \mathcal{D}_i$ and completed response $y \in \mathcal{V}^*$. We model response generation on τ_i as a token-level MDP $(\mathcal{S}, \mathcal{A}, P, \bar{r}_i, \rho_i)$ (Sutton & Barto, 2018). Its state space $\mathcal{S}$ contains prompt-conditioned token prefixes, its action space $\mathcal{A} = \mathcal{V}$ is the vocabulary including the terminal token `eos`, and its initial-state distribution ρ_i draws a prompt $x \sim \mathcal{D}_i$ as $s_1 = x$. The deterministic transition appends the sampled token, $s_{t+1} = P(s_t, y_t) = \mathrm{concat}(s_t, y_t)$. The token-level reward $\bar{r}_i$ is zero before termination and equals $r_i(x, y)$ when `eos` completes response y. Thus an undiscounted rollout sampled from π_θ has return $r_i(x, y)$, and the RLVR objective is

$$J_i(\theta) = \mathbb{E}_{x \sim \mathcal{D}_i,\, y \sim \pi_\theta(\cdot|x)}[r_i(x, y)]. \tag{1}$$

4 CONTINUAL REASONING GYM

In this section, we instantiate the continual RLVR setting from Section 3 with five sequences of verifiable reasoning tasks.

4.1 TASK SEQUENCES

A task sequence turns a static task pool into a staged, non-stationary training stream. For each sequence, we select several related but distinct tasks and train them sequentially, allocating N/T steps to each under a total budget of N steps (Figure 2). We instantiate five task sequences across two modalities. The two text settings draw from Reasoning Gym's largest category `algorithmic` and its dedicated `algebra` category, covering procedural symbolic manipulation and symbolic mathematics (Stojanovski et al., 2025). The three visual settings take VisuLogic's three largest reasoning categories, namely quantitative, spatial, and positional (Xu et al., 2026). Within each category, we define the ordered stage-level subtypes that form the corresponding sequence. Appendix A.1 reports dataset statistics, and Appendix A.3 provides representative examples.

4.2 BASELINE ALGORITHMS

We use Seq. RLVR as the default baseline. At each stage, it continues training the policy with standard RLVR on the arriving task. MTRL jointly optimizes the same task set and serves as the matched multitask reference. Five CL interventions extend Seq. RLVR. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) penalizes changes to important weights. KL-to-old-policy penalizes divergence from the pre-task policy. Orthogonal Subspace Fine-Tuning (OSFT) (Nayak

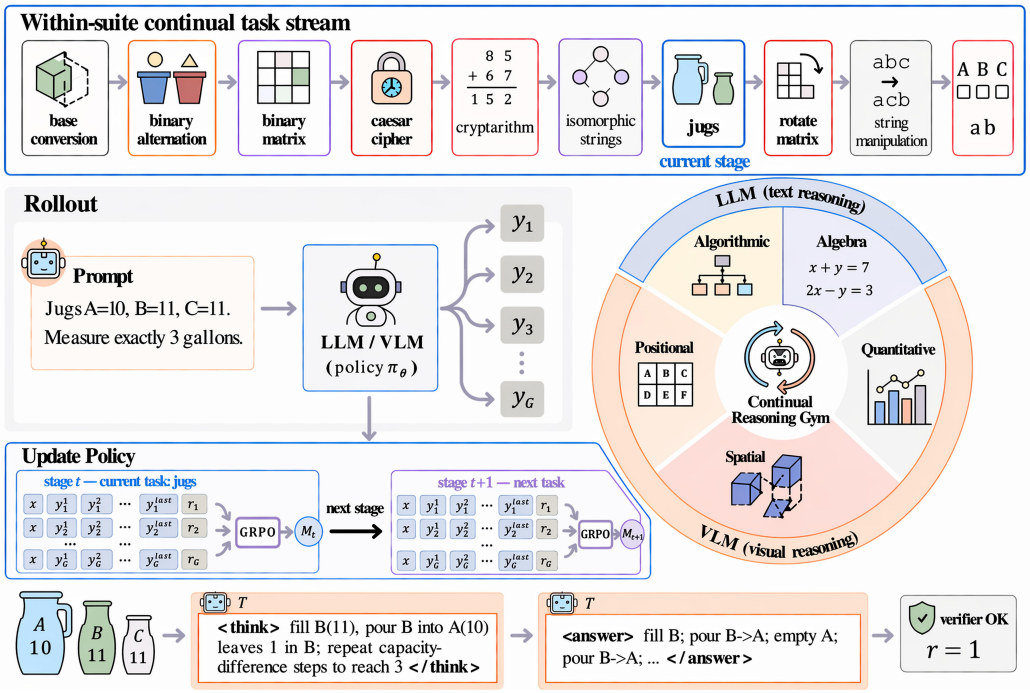


Figure 2: **Policy training proceeds stage by stage through a sequence of verifiable reasoning tasks.** *Top*: the LLM algorithmic sequence contains T=10 tasks, with jugs as the current task. *Left*: at each stage, the policy rolls out responses $y_1, \ldots, y_G$ to a prompt. A verifier scores the responses, and Group Relative Policy Optimization (GRPO) uses these scores to produce the stage policy M_t. *Wheel*: the five task sequences cover two text-reasoning settings and three visual-reasoning settings. *Bottom*: a jugs example shows the policy's reasoning and verifier-checked answer.

et al., 2026) constrains updates to the orthogonal complement of critical directions. ReDo (Sokar et al., 2023) reinitializes dormant units, while FIRE (Han et al., 2026) uses Frobenius-isometry reinitialization to balance stability and plasticity. We additionally include Muon (Liu et al., 2025), which orthogonalizes momentum updates, as an optimizer control.

4.3 Evaluation Quantities

The policy is evaluated on every task before training and after each task stage. From these evaluations, we report BaseAvg and FinalAvg for the two endpoints. To characterize what happens between them, we use forward transfer (FWT), task-learning gain (TLG), and backward transfer (BWT). FWT captures the effect of earlier-task training before a task's own stage, TLG the effect of direct training on that task, and BWT the effect of subsequent training on later tasks. To compare final performance with the matched multitask reference across task sequences, we report the continual-to-multitask ratio (CTM):

$$\mathrm{CTM} = \frac{\mathrm{FinalAvg}}{\mathrm{FinalAvg}_{\mathrm{MTRL}}}, \tag{2}$$

where $\mathrm{FinalAvg}_{\mathrm{MTRL}}$ is the multitask reference trained on the same task set. A CTM of 1 indicates MTRL-level performance. Further details are provided in Appendix A.2.

5 Continual Prompt Replay

In this section, we first decompose final performance to show why the MTRL gap remains despite modest forgetting. We then formalize shared reasoning to explain modest forgetting and motivate CPR, which brings previous-task prompts into current-policy training to improve FWT and TLG.

5.1 Decomposing the MTRL Gap

Seq. RLVR remains below MTRL despite modest forgetting. We first test whether forgetting accounts for the gap. Under Seq. RLVR, training on later tasks reduces performance on previously learned tasks by only 2.47 percentage points on average across the five task sequences, yet final performance reaches only 88% of MTRL (Figure 3). To determine what accounts for the gap beyond this modest forgetting, we decompose $\mathrm{FinalAvg}$ as follows.

Proposition 1 (Final Performance Beyond Forgetting)**.**

$$\mathrm{FinalAvg} = \mathrm{BaseAvg} + \tfrac{T-1}{T}\,\mathrm{FWT} + \mathrm{TLG} + \tfrac{T-1}{T}\,\mathrm{BWT}. \tag{3}$$

Proof. See Appendix C.1. □

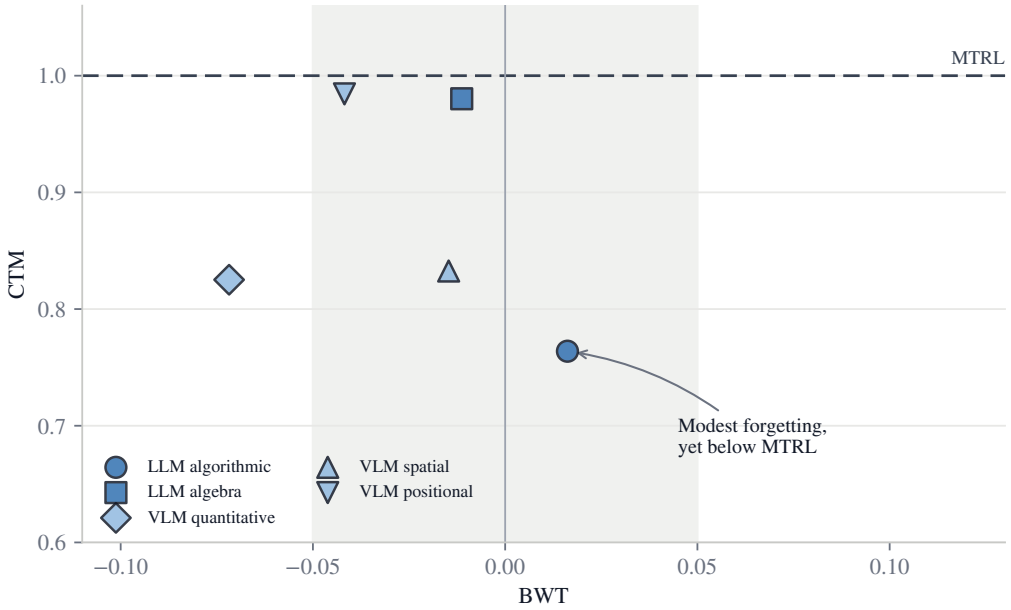


Figure 3: **Seq. RLVR remains below MTRL despite modest forgetting.** Each marker represents one task sequence. Across task sequences, performance on earlier tasks falls by only 2.47 percentage points on average, yet final performance reaches 88% of MTRL.

Since all methods within a task sequence start from the same initial policy, $\Delta\mathrm{BaseAvg} = 0$. Thus,

$$\Delta\mathrm{FinalAvg} = \tfrac{T-1}{T}\Delta\mathrm{FWT} + \Delta\mathrm{TLG} + \tfrac{T-1}{T}\Delta\mathrm{BWT}. \tag{4}$$

Even if the weighted BWT contribution were zero, the gap to MTRL would remain. We therefore ask two questions: why Seq. RLVR exhibits modest forgetting, and how the remaining FWT and TLG gaps can be reduced.

5.2 Shared Reasoning Explains Modest Forgetting

To address the first question, we analyze how later-task updates affect earlier-task objectives. Consider an update at stage t with parameters θ. Let J_i denote the objective for task i, with gradient $g_i^{(t)} = \nabla_\theta J_i(\theta)$, and let $g_k^{(t)}$ denote the update direction estimated from task k. After an update of size η along this direction,

$$J_i(\theta + \eta g_k^{(t)}) - J_i(\theta) = \eta \langle g_i^{(t)}, g_k^{(t)} \rangle + O(\eta^2).$$

The sign of the inner product indicates whether an update from task k locally benefits or harms task i. The modest forgetting observed above suggests that the cumulative effect of later-task updates is

not strongly harmful to earlier tasks on average. One possible explanation is that reasoning tasks share reasoning structure, thus allowing updates on one task to benefit others on average. We call this pattern shared reasoning and formalize it as follows.

Assumption 1 (Mean task-gradient alignment)**.** *Across reasoning tasks,*

$$\mathbb{E}_{t,i,k}\left[\langle g_i^{(t)}, g_k^{(t)}\rangle\right] \geq 0. \tag{5}$$

For Seq. RLVR, every update is drawn from the current task, so $k = t$. Restricting to earlier tasks $i < t$ and averaging over BWT's stages and tasks gives the following corollary.

Corollary 1 (Backward Transfer under Mean Alignment)**.** *Under Assumption 1, later-stage Seq. RLVR updates have a non-negative expected first-order contribution to earlier-task objectives.*

Proof. See Appendix C.3. □

Corollary 1 connects mean alignment to modest forgetting: later-task updates can benefit earlier-task objectives on average. We next ask whether shared reasoning can also improve FWT and TLG. Since mean alignment does not require every task pair to align, the current-task direction may harm some task objectives, while previous-task directions may complement it. This motivates CPR.

5.3 Harnessing shared reasoning with CPR

To bring previous-task directions into the current update, CPR replays previous-task prompts and regenerates their responses with the current policy. CPR stores each prompt with its task identity and latest pass rate. During later stages, it selects previous-task prompts with intermediate pass rates to replace a fraction of current-task prompts, changing the task-sampling distribution without increasing the number of prompts or rollouts per update.

Let $\widehat{\rho}_t$ be the fraction of replayed prompts in an update at stage t, and let $q_{t,k}$ be the fraction of those prompts drawn from task $k < t$. The mixed-batch update direction is

$$g_t^{\mathrm{CPR}} = (1 - \widehat{\rho}_t) g_t^{(t)} + \widehat{\rho}_t \sum_{k<t} q_{t,k} g_k^{(t)}. \tag{6}$$

Applying Assumption 1 to this mixture yields the following corollary.

Corollary 2 (CPR under Mean Alignment)**.** *Under Assumption 1, CPR updates have a non-negative expected first-order contribution when averaged over the relevant stages and task objectives.*

Proof. See Appendix C.3. □

6 Experiments

We first test the shared-reasoning explanation for modest forgetting, then evaluate whether CPR harnesses shared reasoning to improve FWT and TLG.

6.1 Setup

We instantiate CRG with Qwen3-4B in two text settings and Qwen2.5-VL-7B in three visual settings. For each setting, every method uses a budget of $N = 500$ optimizer steps. Appendix B.1 provides further experimental details.

6.2 Evidence for Shared Reasoning

To evaluate shared reasoning as an explanation for modest forgetting, we measure task-gradient alignment and trace changes in reasoning behavior across stages.

Task gradients are positively aligned on average. On the ten-task LLM algorithmic sequence, we compute pairwise cosine similarities between task gradients under Seq. RLVR. The mean cosine is 0.345, and 82.2% of task pairs have positive cosine (Figure 4). This positive mean alignment is consistent with the shared-reasoning explanation for modest forgetting. Additional gradient diagnostics are provided in Appendix D.5.1.

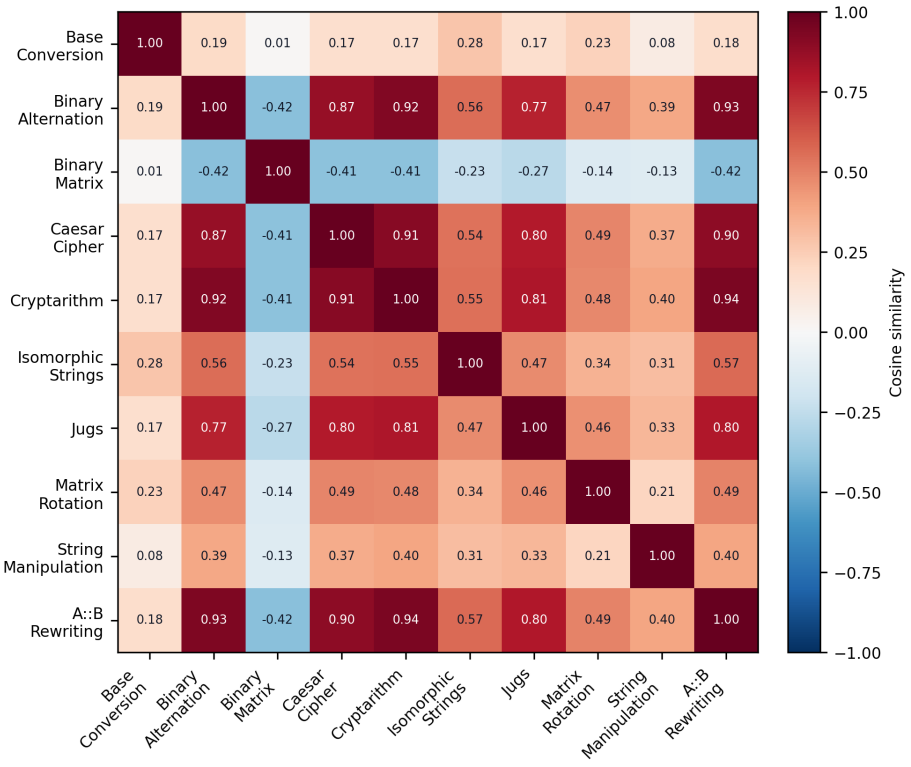


Figure 4: **Task gradients are positively aligned on average under Seq. RLVR.** The heatmap shows pairwise cosine similarities between task gradients on the ten-task LLM algorithmic sequence. The mean cosine is 0.345, and 82.2% of task pairs have positive cosine.

Shared reasoning appears in model behavior. Having found positive mean alignment between task gradients, we next examine whether training on other tasks produces corresponding changes in performance and reasoning. Figure 5 tracks jugs success and representative reasoning excerpts throughout Seq. RLVR. Jugs performance improves both before its own stage and during later-task training, with the later-task gain directly consistent with shared reasoning as an explanation for modest forgetting. The excerpts show a parallel progression. Initially, the model finds an optimal route but repeatedly second-guesses it and emits no answer. Before jugs training, it takes a longer route without answering. After direct training, it returns a valid answer after rechecking. At the end of the sequence, it reaches the same answer through shorter, more direct reasoning. Full transcripts are provided in Appendix D.5.2.

We therefore observe shared reasoning at both the update and behavioral levels. We next test whether CPR can harness it to improve learning on the arriving and future tasks.

6.3 CPR Results

CPR reaches MTRL-level performance on average. CPR achieves a mean CTM of 1.03, up from 0.88 under Seq. RLVR, and is the only evaluated CL method to reach MTRL-level performance on average (Table 2).

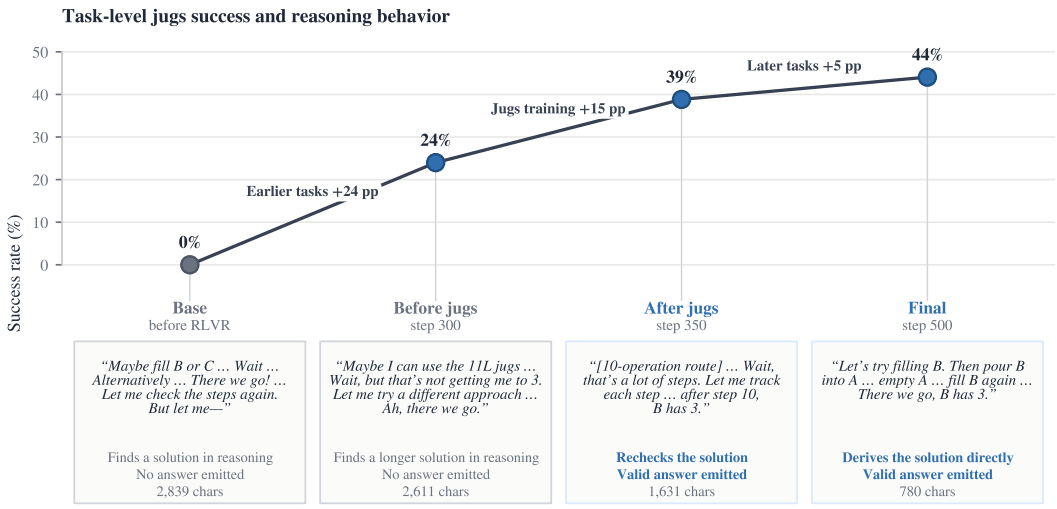


Figure 5: **A jugs case illustrates shared reasoning in both performance and reasoning behavior.** Jugs performance improves before its own stage and again during later-task training. The aligned excerpts show a progression from unresolved reasoning to valid and more direct solutions.

CPR improves forward transfer and direct task learning. Relative to Seq. RLVR, CPR raises mean FinalAvg by 6.4 percentage points. Of this gain, FWT accounts for $+2.4$ points and TLG for $+4.9$ points, while BWT accounts for -0.9 points. CPR's gain therefore comes from improved learning on arriving and future tasks (Table 2).

Table 2: **CPR is the only method to reach MTRL-level performance on average.** For each method, FWT, TLG, and BWT report their percentage-point contributions to the change in $\mathrm{FinalAvg}$ relative to Seq. RLVR. Total is their sum.

Metric	Seq. RLVR	Reset (ReDo)	Reset (FIRE)	Regularization (EWC)	Optimizer (Muon)	Isolation (OSFT)	Regularization (KL)	**Replay (CPR)**
FWT	0.0	-1.5	-5.8	-1.7	+1.3	-3.4	+0.6	**+2.4**
TLG	0.0	-7.8	-4.0	+0.4	-0.8	-7.9	-5.0	**+4.9**
BWT	0.0	+3.0	-14.5	+2.7	+2.5	+2.7	**+5.4**	-0.9
Total	0.0	-6.3	-24.4	+1.4	+3.0	-8.5	+1.1	**+6.4**
Mean CTM	0.88	0.78	0.29	0.87	0.97	0.65	0.95	**1.03**

Previous-task gradients better align updates with the full task set. To examine their role in CPR's gains, we evaluate Qwen3-4B task gradients and compare each current-task gradient with an equally weighted combination of the current-task gradient and the mean previous-task gradient. This combination increases cosine similarity with the mean gradient across all tasks from 0.43 to 0.79 on average and yields higher alignment at all nine stages (Figure 6). Additional gradient diagnostics are provided in Appendix D.5.1.

To determine whether CPR benefits from replay itself or from regenerating responses with the current policy, we compare it with no replay and sample replay.

Current-policy regeneration drives the replay gain. We replace CPR's freshly generated responses with trajectories from an earlier policy. On LLM algorithmic reasoning, this sample-replay variant reaches a FinalAvg of 47.5%, below no replay at 49.9%, whereas CPR reaches 63.3% and approaches MTRL at 65.3% (Figure 7). The gain therefore depends on regenerating responses with the current policy.

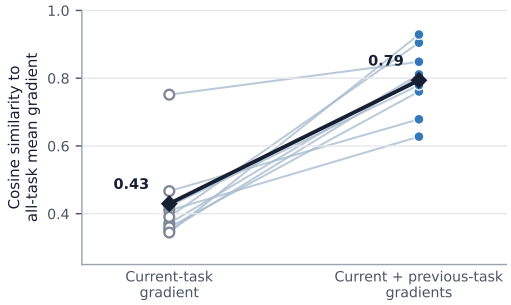


Figure 6: **Previous-task gradients better align updates with the full task set.** For Qwen3-4B, each line compares a current-task gradient with an equal-weight combination of that gradient and the mean previous-task gradient. Dark diamonds show the averages across the nine stages.

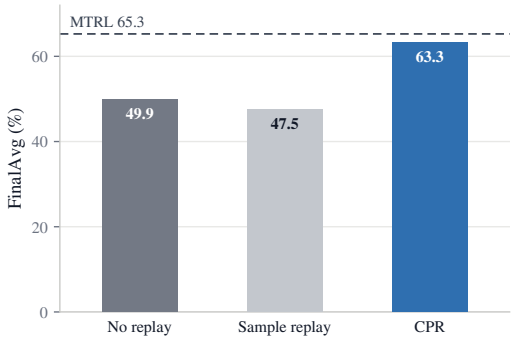


Figure 7: **Current-policy regeneration drives CPR's replay gain.** On LLM algorithmic reasoning, CPR approaches MTRL. Sample replay reuses trajectories generated by an earlier policy and underperforms no replay.

This comparison shows that CPR's gain depends on both previous-task prompts and current-policy regeneration. Appendix D.2 reports additional replay-pool comparisons, and Appendix D.3 reports replay-ratio robustness.

7 Conclusion

In this paper, we introduce CRG to evaluate whether continual RLVR can reach MTRL-level performance across text and visual reasoning task sequences. Using CRG, we identify two key observations: Seq. RLVR forgets modestly, yet its performance remains below MTRL. To understand the latter, we decompose final performance and show that preserving earlier-task performance alone cannot close the gap. To understand the former, we identify shared reasoning, whereby training on one task benefits others on average. We then instantiate this insight as CPR, which replays previous-task prompts and regenerates responses with the current policy to harness shared reasoning and improve learning on arriving and future tasks. On CRG, CPR is the only evaluated CL method to reach MTRL-level performance on average.

Limitations. Our experiments focus on five task sequences using Qwen3-4B and Qwen2.5-VL-7B. Extending CRG to agentic settings and larger models remains future work. Our analysis characterizes local update effects and does not capture nonlinear dynamics over extended training.

AI use statement

In this work, we use AI for formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, refining hypotheses, implementing methods, and reformatting datasets. We have not used AI for developing theoretical models or con-

ceptual frameworks, designing research methodology or experiments, generating synthetic datasets, assisting with translation, supporting qualitative and thematic data analysis, or interpreting results, and the remaining required disclosure tasks are not applicable to this work. Additionally, we use AI for creating and modifying scientific figures, creating and editing software code, drafting and editing parts of the paper, summarizing existing literature, sourcing information, identifying relevant literature, and suggesting the paper's structure. We have reviewed all AI-assisted work. We checked mathematical arguments, verified citations against their sources, tested AI-assisted code through the reported workflows, and reviewed AI-assisted figures and manuscript revisions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of AI.

Ethics statement

This work does not involve private data or raise ethical concerns.

Reproducibility statement

Section 4 defines the task sequences and stage protocol, while Section 6.1 specifies training and evaluation. Appendix A.1 lists the task sequences, dataset statistics, task construction, and VLM provenance. The rest of the appendix provides exact metric definitions and boundary conventions, complete hyperparameters, full results, proofs, CPR details and ablations, gradient diagnostics, and raw case-study transcripts. Code, configurations, evaluation records, and scripts for regenerating the reported tables and figures will accompany the paper release.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, Taylor W. Killian, Mikhail Yurochkin, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. Revisiting reinforcement learning for LLM reasoning from a cross-domain perspective. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

Isaac Han, Sangyeon Park, Seungwon Oh, Donghu Kim, Hojoon Lee, and Kyung-Joong Kim. FIRE: Frobenius-isometry reinitialization for balancing the stability-plasticity tradeoff. In *International Conference on Learning Representations*, 2026.

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

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017.

Lirui Luo, Guoxi Zhang, Hongming Xu, Cong Fang, and Qing Li. SPHERE: Mitigating the loss of spectral plasticity in mixture-of-experts for deep reinforcement learning. In *International Conference on Machine Learning*, 2026.

Saleh Momeni and Bing Liu. Achieving upper bound accuracy of joint training in continual learning. *arXiv preprint arXiv:2502.12388*, 2025.

Nikhil Shivakumar Nayak, Krishnateja Killamsetty, Ligong Han, Abhishek Bhandwaldar, Prateek Chanda, Kai Xu, Hao Wang, Aldo Pareja, Oleg Silkin, Mustafa Eyceoz, and Akash Srivastava. Sculpting subspaces: Constrained full fine-tuning in LLMs for continual learning. In *International Conference on Learning Representations*, 2026.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's razor: Why online reinforcement learning forgets less. In *International Conference on Learning Representations*, 2026.

Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, 2023.

Tejas Srinivasan, Ting-Yun Chang, Leticia Leonor Pinto Alva, Georgios Chochlakis, Mohammad Rostami, and Jesse Thomason. CLiMB: A continual learning benchmark for vision-and-language tasks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2022.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. REASONING GYM: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

Tianqi Tang, Shohreh Deldari, Hao Xue, Celso De Melo, and Flora D. Salim. ViLCo-Bench: VIdeo language COntinual learning benchmark. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024.

Tristan Tomilin, Meng Fang, Yudi Zhang, and Mykola Pechenizkiy. COOM: A game benchmark for continual reinforcement learning. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023.

Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, Constantin Ruhdorfer, Bram Grooten, Fabrice Kusters, Yali Du, Andreas Bulling, Mykola Pechenizkiy, and Meng Fang. MEAL: A benchmark for continual multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2026.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, and Xuanjing Huang. TRACE: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*, 2023.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base LLMs. In *International Conference on Learning Representations*, 2026.

Maciej Wolczyk, Michal Zajac, Razvan Pascanu, Lukasz Kucinski, and Piotr Milos. Continual world: A robotic benchmark for continual reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: Training LLM agents for long-horizon decision making through multi-turn reinforcement learning. In *International Conference on Learning Representations*, 2026.

Weiye Xu, Jiahao Wang, Weiyun Wang, Zhe Chen, Wengang Zhou, Aijun Yang, Lewei Lu, Houqiang Li, Xiaohua Wang, Xizhou Zhu, Wenhai Wang, Jifeng Dai, and Jinguo Zhu. VisuLogic: A benchmark for evaluating visual reasoning in multi-modal large language models. In *International Conference on Learning Representations*, 2026.

Zhihao Zhang, Qiaole Dong, Qi Zhang, Jun Zhao, Enyu Zhou, Zhiheng Xi, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Mingqi Wu, Yanwei Fu, Tao Ji, Tao Gui, Xuanjing Huang, and Kai Chen. Why reinforcement fine-tuning enables MLLMs preserve prior knowledge better: A data perspective. In *International Conference on Learning Representations*, 2026.

APPENDIX

APPENDIX CONTENTS

A Continual Reasoning Gym Details **13**
A.1 Task Sequences . . . 13
A.2 Evaluation Metrics . . . 16
A.3 Representative Task Examples . . . 16

B Experimental Details **20**
B.1 Training Configuration . . . 20
B.2 CPR Implementation . . . 22
B.3 CPR Resource Overhead . . . 23

C Theoretical Analysis **24**
C.1 Proof of Proposition 1 . . . 24
C.2 First-Order CPR Analysis . . . 25
C.3 Corollary Proofs . . . 26

D Additional Results **27**
D.1 Full Method Results . . . 27
D.2 Additional CPR Ablation Results . . . 31
D.3 Robustness to Replay Ratio ρ . . . 32
D.4 CPR Gains Accompany Both Entropy Increases and Decreases . . . 33
D.5 Additional Evidence for Shared Reasoning . . . 34
D.6 Pass@K Comparison with the Base Model . . . 39
D.7 Trained Small Models versus Larger Zero-Shot Models . . . 40

E Extended Related Work **41**

A CONTINUAL REASONING GYM DETAILS

A.1 TASK SEQUENCES

CRG instantiates five task sequences comprising 30 stages. The text tasks draw from ReasoningGym (Stojanovski et al., 2025), and the visual tasks draw from VisuLogic (Xu et al., 2026).

VisuLogic defines the top-level quantitative, spatial, and positional domains. We partition each domain into project-defined stage subtypes and fix their training order in Table 3. GPT-5.5 using the `xhigh` reasoning-effort setting generated the initial subtype assignments. Professional researchers then manually reviewed every assignment. Three quantitative examples remained ambiguous after review and are excluded from the four stage pools. The researcher-reviewed labels define the final spatial and positional subtypes.

A.1.1 LLM REASONING

Algorithmic Reasoning (LLM). This setting covers ten procedural tasks whose symbolic answers require explicit multi-step computation. Their order is:

Table 3: **Five task sequences define the Continual Reasoning Gym streams.** Arrows show the training sequence. The LLM algorithmic, LLM algebra, VLM quantitative, VLM spatial, and VLM positional streams contain 10, 6, 4, 6, and 4 stages, respectively.

Modality	Stream	Ordered stages
LLM	Algorithmic Reasoning	Base Conversion → Binary Alternation → Binary Matrix → Caesar Cipher → Cryptarithm → Isomorphic Strings → Jugs → Matrix Rotation → String Manipulation → A::B Rewriting
LLM	Algebra Reasoning	Complex Arithmetic → Intermediate Integration → Polynomial Equations → Polynomial Multiplication → Simple Equations → Simple Integration
VLM	Quantitative Reasoning	Linear Quantity → Prime Quantity → Point Quantity → Counter Quantity
VLM	Spatial Reasoning	Assembly → Cross-sectional → Hexahedron → Three Views → Spatial Orders → Polyhedron Rotation
VLM	Positional Reasoning	Translation → Rotation → Comparative → Flip

Table 4: **Continual Reasoning Gym contains 30 stages across five task sequences.** Each setting is split 8:2 into training and test sets. LLM examples are generated procedurally, with at most 20,000 total examples sampled for each text setting. Symbolic and four-way answers are scored by the corresponding task verifier.

Modality	Setting	Stages	Total examples	Answer format
LLM	Algorithmic	10	20,000	Symbolic
LLM	Algebra	6	20,000	Symbolic
VLM	Quantitative	4	1,386	Four-way choice
VLM	Spatial	6	1,043	Four-way choice
VLM	Positional	4	743	Four-way choice
Total		**30**	—	—

Base Conversion rewrites a number between numeral bases.
Binary Alternation returns the minimum number of swaps that make a binary string alternate, or reports impossibility.
Binary Matrix computes, for every cell of a binary matrix, the Manhattan distance to the nearest zero.
Caesar Cipher recovers the plaintext of a shift-ciphered message whose shift is unknown.
Cryptarithm solves an addition puzzle for the unique letter-to-digit assignment.
Isomorphic Strings decides whether two strings admit a one-to-one character mapping that preserves structure.
Jugs produces a sequence of fill, pour, and empty operations that measures a target volume.
Matrix Rotation rotates a matrix by a specified angle.
String Manipulation applies a described sequence of string edits.
A::B Rewriting reduces a program under the A::B rewriting system to its normal form.

Each sub-task carries an exact-match verifier over its symbolic answer.

Algebra Reasoning (LLM). This setting covers six symbolic-mathematics tasks. Their order is:

Complex Arithmetic evaluates arithmetic on complex numbers.
Intermediate Integration evaluates integrals at a harder difficulty level.
Polynomial Equations solves a polynomial for its roots.
Polynomial Multiplication expands a product of polynomials.
Simple Equations solves a basic algebraic equation for its unknown.
Simple Integration evaluates integrals at an easier difficulty level.

Answers are checked symbolically, so numerically equivalent forms are accepted.

A.1.2 VLM VISUAL REASONING

Quantitative Reasoning (VLM). This setting tests changes in graphical quantities, including counts of points, lines, and angles and arithmetic relations among shapes. Its four stages are project-defined subtypes of VisuLogic's Quantitative domain:

Linear Quantity element counts change linearly along a row, column, or sequence, by a constant or position-indexed step.
Prime Quantity counts follow a number-theoretic rule such as primality, parity, multiples, or divisibility.
Point Quantity the changing quantity is points, intersections, endpoints, or nodes.
Counter Quantity other count patterns over elements, blocks, edges, segments, or black and white cells.

Spatial Reasoning (VLM). This setting requires reconstructing three-dimensional structure from two-dimensional figures, including folding surfaces and integrating solid shapes. Its six stages are project-defined subtypes of VisuLogic's Spatial domain:

Assembly assembling, completing, or decomposing 3D parts into matching polyhedron pieces.
Cross-sectional identifying the cross-section of a 3D solid cut by a plane.
Hexahedron reconstructing a cube or cuboid surface from its net or folded box.
Three Views matching a 3D object to its orthographic projection views.
Spatial Orders a sequence, classification, or ordering pattern over spatial figures.
Polyhedron Rotation comparing a solid under 3D rotations, viewing orientations, or symmetry.

Positional Reasoning (VLM). This setting examines transformations that relocate objects while preserving their constituent elements. Its four stages are project-defined subtypes of VisuLogic's Positional domain:

Translation marked elements shift, swap, or reorder positions while keeping their orientation.
Rotation elements or orientations rotate or cycle around a center or ordered positions.
Comparative a mixed positional relationship in which no single translation, rotation, or flip dominates.
Flip the pattern uses mirror reflection across an axis.

All three VLM settings use four-way multiple-choice instances scored by exact match.

A.2 Evaluation Metrics

Stage-wise evaluation records one score for every stage policy and task. Let $R_{j,i} = S(M_j, \tau_i)$ denote the mean verifier success rate of stage-j policy M_j on the fixed evaluation examples for task τ_i. Evaluating $j = 0, \ldots, T$ on every $i = 1, \ldots, T$ gives a $(T+1) \times T$ matrix R for each method and task sequence. This evaluation follows GEM (Lopez-Paz & Ranzato, 2017), its continual-RL use in Continual World (Wolczyk et al., 2021), and its large-model use in TRACE (Wang et al., 2023).

Four entries identify the task-level performance values used by these metrics:

$$B_i = R_{0,i}, \qquad P_i = R_{i-1,i}, \qquad A_i = R_{i,i}, \qquad F_i = R_{T,i}. \tag{7}$$

They denote performance before any training, immediately before and after direct training on task i, and after all stages. The boundary conventions are $P_1 := B_1$ and $A_T := F_T$.

The metrics are

$$\begin{aligned} \text{BaseAvg} &= \frac{1}{T}\sum_{i=1}^{T} B_i, & \text{FinalAvg} &= \frac{1}{T}\sum_{i=1}^{T} F_i, \\ \text{FWT} &= \frac{1}{T-1}\sum_{i=2}^{T}(P_i - B_i), & \text{TLG} &= \frac{1}{T}\sum_{i=1}^{T}(A_i - P_i), \\ \text{BWT} &= \frac{1}{T-1}\sum_{i=1}^{T-1}(F_i - A_i). \end{aligned} \tag{8}$$

FWT and BWT follow Lopez-Paz & Ranzato (2017), while TLG is our addition. FWT and BWT average over $T-1$ applicable tasks because forward transfer is undefined for the first task and backward transfer is undefined for the last task. TLG averages direct learning over all T tasks. Means across the five settings weight each setting equally.

A.3 Representative Task Examples

Figure 8 shows one task from each setting.

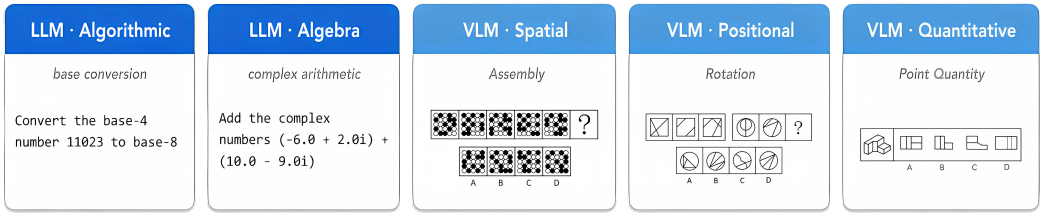


Figure 8: **Representative CRG tasks.** One example is shown for each setting.

The transcripts below show one evaluation prompt and a model response in the format consumed by the corresponding verifier. They include the model's `<think>` reasoning and `<answer>` span. The latter supplies the answer scored by the task verifier.

A.3.1 LLM Reasoning

Base Conversion *LLM algorithmic*

```
[PROMPT]
Your task is to convert a number between two different bases.

If the target base is > 10, use lowercase letters a-z for digits above 9.

Now, convert the base-4 number 11023 to base-8
```

[MODEL RESPONSE]

```
<think>
Okay, let's see. I need to convert the base-4 number 11023 to base-8. Hmm, how do I do that?
     Well, I remember that converting between bases can be done by first converting the
     number to decimal (base 10) and then to the target base. Let me try that.

First, let's convert 11023 (base 4) to decimal. Each digit represents a power of 4. Starting
     from the right, the digits are positions 0 to 4. So the number is 1 1 0 2 3.

So, the digits are 1, 1, 0, 2, 3. Let me calculate each digit's value:

1 * 4^4 = 1 * 256 = 256
1 * 4^3 = 1 * 64 = 64
0 * 4^2 = 0 * 16 = 0
2 * 4^1 = 2 * 4 = 8
3 * 4^0 = 3 * 1 = 3

Now add them all up: 256 + 64 is 320, plus 0 is 320, plus 8 is 328, plus 3 is 331. So the
     decimal equivalent is 331.

Now, I need to convert 331 (base 10) to base 8. To do that, I divide the number by 8 and keep
      track of the remainders.

331 divided by 8 is 41 with a remainder of 3.
41 divided by 8 is 5 with a remainder of 1.
5 divided by 8 is 0 with a remainder of 5.

So the remainders, from last to first, are 5, 1, 3. Therefore, the base 8 number is 513.

Let me check that. 5*8^2 + 1*8 + 3 = 5*64 + 8 +3 = 320 + 11 = 331. Yes, that's correct. So
     the answer should be 513.
</think>

<answer>513</answer>
```

Complex Arithmetic *LLM algebra*

[PROMPT]

```
Add the complex numbers: (-6.0 + 2.0i) + (10.0 - 9.0i)
```

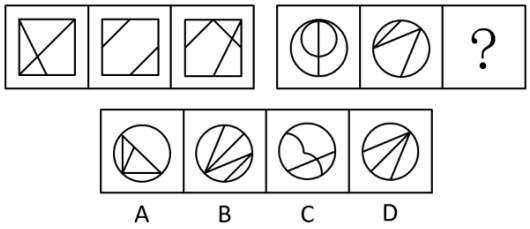
[MODEL RESPONSE]

```
<think>
Okay, let's see. I need to add two complex numbers: (-6.0 + 2.0i) and (10.0 - 9.0i). Hmm, how
      do you add complex numbers again? Oh right, you just add the real parts together and
     the imaginary parts together.

So the first number is -6.0 plus 2.0i. The second one is 10.0 minus 9.0i. Let me break it
     down. The real parts are -6.0 and 10.0. Adding those together: -6.0 + 10.0. That should
     be 4.0, right? Because 10 minus 6 is 4.

Now the imaginary parts: 2.0i and -9.0i. Adding those gives 2.0 - 9.0, which is -7.0i. So
     combining those two results, the sum should be 4.0 - 7.0i. Let me check again to make
     sure I didn't mix up any signs. The second number's imaginary part is negative, so
     subtracting 9.0i from 2.0i would be -7.0i. Yeah, that seems right. So the final answer
     is 4.0 - 7.0i.
</think>

<answer>4.0 - 7.0i</answer>
```

A.3.2 VLM Visual Reasoning

Each visual example is a four-way VisuLogic-derived problem whose image contains the answer options. We report the image, shared textual prompt, and model response.

Linear Quantity *VLM quantitative*



[PROMPT]
From the four given options, select the most suitable one to fill in the question mark, so that a certain regularity is presented:

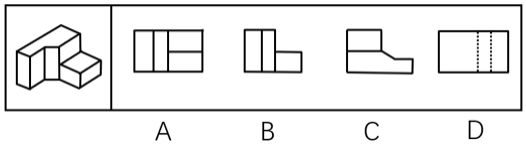
A: A
B: B
C: C
D: D
[four answer options A--D are shown in the image above]

[MODEL RESPONSE]
<think>The pattern involves dividing shapes into sections and rotating these sections within the shape.

1. In the first row, the square is divided diagonally into two triangles, which are then rotated and shifted within the square.
2. Applying this logic to the circles:
   - The circle is divided into three equal parts (sectors).
   - These sectors are then rotated and shifted within the circle to form the pattern seen in the last column of the first row.

The correct option should reflect this rotation and shifting pattern within the circle. Option A matches this description as it shows the sectors rotated and shifted similarly to the pattern observed in the first row's last column.**</think>**
<answer>A</answer>

Polyhedron Rotation *VLM spatial*



[PROMPT]
From any angle, which one on the right is not a view of the three-dimensional shape given on the left?

A: A
B: B
C: C
D: D
[four answer options A--D are shown in the image above]

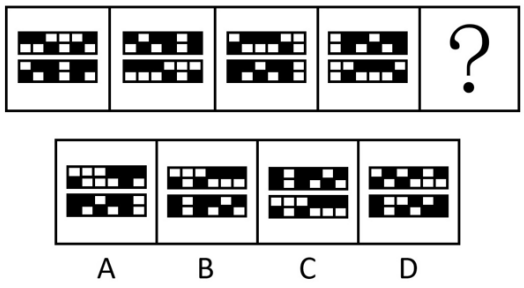
[MODEL RESPONSE]
<think>Observation:

1. Shape left:
   - Block structure vertically stacked horizontally.
   - Visible rectangular planes intersecting.

2. Options:
   - A: Horizontal split rectangles aligned vertically.
   - B: Vertical split rectangles aligned horizontally.
   - C: Horizontal split rectangles aligned horizontally.
   - D: Rectangular dashed lines vertically aligned.

3. Analysis:
   - Option A matches horizontal splits vertically aligned.
   - Option B matches vertical splits horizontally aligned.
   - Option C matches horizontal splits horizontally aligned.
   - Option D dashed vertical rectangles align incorrectly.**</think>**
<answer>D</answer>

Rotation *VLM positional*




```
[PROMPT]
From the four given options, choose the most appropriate one to fill in the question mark, so
      that a certain pattern is formed:

A: A
B: B
C: C
D: D
[four answer options A--D are shown in the image above]

[MODEL RESPONSE]
<think>
The arrangement of squares within each grid seems to alternate between two distinct patterns:

1. Grids with horizontal alignment.
2. Grids with vertical alignment.

Observing closely:
- Option A mirrors horizontally aligned squares.
- Option B mirrors vertically aligned squares.
- Option C mirrors horizontally aligned squares.
- Option D mirrors vertically aligned squares.

Given the alternating pattern:
Option B matches vertically aligned squares.

</think>
<answer>B</answer>
```

B EXPERIMENTAL DETAILS

B.1 TRAINING CONFIGURATION

Table 5: **All runs use the same core RLVR configuration.** The text experiments use Qwen3-4B, and the visual experiments use Qwen2.5-VL-7B. Training updates all model parameters, including the VLM vision tower.

Group	Hyperparameter	Value
Model	LLM backbone	Qwen3-4B
	VLM backbone	Qwen2.5-VL-7B
RLVR	Advantage estimator	GRPO
	Entropy coefficient	1×10^{-3}
	Rollouts per prompt	5
	PPO clip range	0.2
	PPO epochs	1
	Loss reduction	Token mean
Optimization	Optimizer	AdamW
	Learning rate	1×10^{-6}
	Adam β values	$(0.9, 0.999)$
	Weight decay	0.01
	Learning-rate schedule	Constant
	Warmup steps	0
	Gradient clipping	1.0
	Train batch size	512
	PPO mini-batch size	256
	PPO micro-batch per GPU	10 for LLM, 4 for VLM
	Total optimizer steps	500
Generation	Temperature	1.0
	Top-p	1.0
	Top-k	-1
Sequence length	Maximum prompt length for LLM	1024
	Maximum response length for LLM	1024
	Maximum prompt length for VLM	1024
	Maximum response length for VLM	1024
	Maximum model length for VLM	3072
Parameter update	LoRA rank	0, corresponding to full-parameter updates
	VLM vision tower	Unfrozen

Table 6: **MTRL accesses the complete task pool from the first update, whereas CPR replays previous-task prompts.** Both use the shared configuration in Table 5.

Method	Setting	Value
MTRL	Task access	Complete task pool available from the first update
	LLM task sampling	$P_{\mathrm{MTRL}}(k) = 1/T$
CPR	Replay pool	Prompts from previous tasks
	Replay fraction	$\rho = 0.5$
	Maximum replayed prompts per batch	$\min\{\lfloor \rho b \rfloor, \lfloor b/2 \rfloor\}$ for batch size b
	Store capacity	One record per observed prompt with task identity and verifier metadata. Footprint reported in Appendix B.3
	Eligibility	Latest pass rate in $[0.24, 0.70]$
	Priority	Smallest distance from pass rate 0.5 first
	Cooldown	5 updates

Table 7: **Each CL baseline changes a specific part of the shared training protocol.** The table lists every baseline-specific override. All remaining settings follow Table 5.

Method	Setting	Value
EWC	Penalty coefficient	1.0
	Fisher decay	1.0
	Numerical stabilizer ϵ	10^{-8}
	Fisher samples	All available samples
FIRE	Newton–Schulz steps	5
	Reset modules	Attention query and key projections
	Task-boundary handling	Reset the optimizer and synchronize rollout weights after reset
ReDo	Dormancy threshold	$\tau = 0.1$
	Calibration set	8 examples
	LeCun initialization	Disabled
Muon	Momentum	0.95
	Newton–Schulz steps	5
	Nesterov momentum	Enabled
	AdamW-side β values	$(0.9, 0.999)$
	AdamW-side ϵ	10^{-8}
	Learning-rate scaling	RMS-matched AdamW
	Weight-decay scaling	1.0
	Maximum matrix aspect ratio	8
	Maximum matrix dimension	65,536
OSFT	Rank ratio	0.5
	Initialization	At the start of training and after every task switch
	Task-boundary handling	Reset the optimizer and synchronize rollout weights after reinitialization
	Numerical precision	FP32 upcast with BF16 output
KL-to-old-policy	KL coefficient	0.001
	Old-prompt fraction	0.5
	Old-policy refresh	At every task switch
	Checkpoint requirement	Task-boundary checkpoint

B.2 CPR Implementation

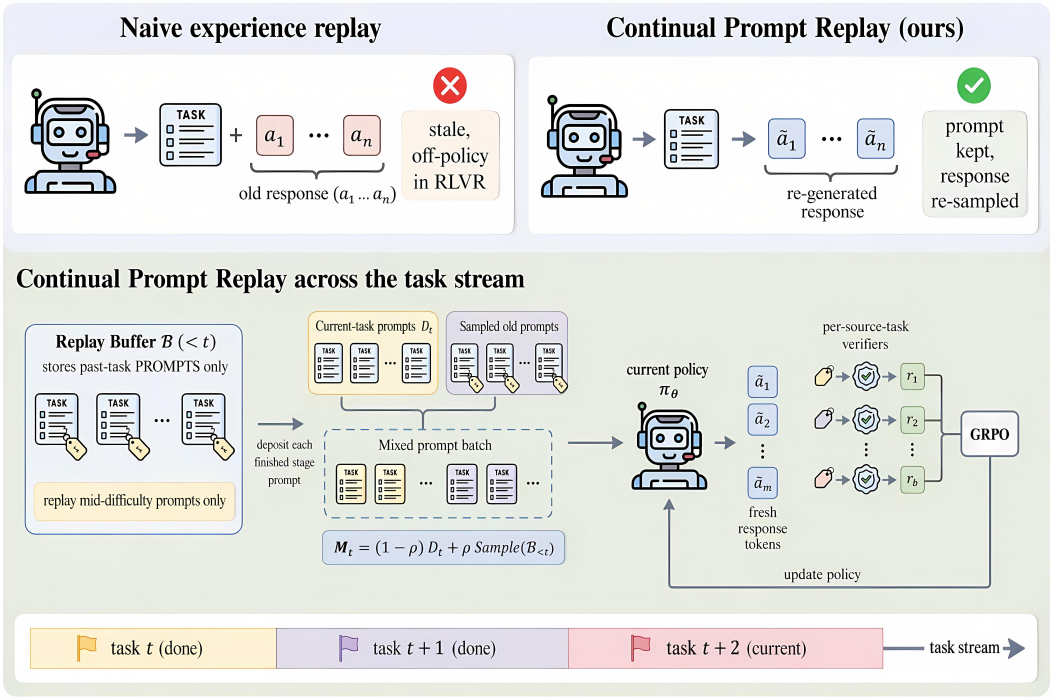


Figure 9: **CPR turns previous-task prompts into current-policy RLVR updates.** The top row contrasts stale trajectory replay with CPR's response regeneration. The lower panel shows one CPR update during a later stage. CPR selects previous-task prompts, mixes them with current-task prompts, generates every response with the current policy, and scores each response with its task verifier before the GRPO update.

Algorithm 1 specifies one CPR update during stage t.

Algorithm 1: One CPR update during stage t

Input: current-task prompt distribution $\mathcal{D}_t$, replay store $\mathcal{B}_{<t}$, task verifiers $\{r_k\}_{k\le t}$, current policy π_θ, batch size b, and replay fraction ρ.

1. set the maximum number of replayed prompts to $b_R = \min\{\lfloor \rho b \rfloor, \lfloor b/2 \rfloor\}$.
2. construct a batch of b prompts by replacing at most b_R current-task prompts with eligible prompts from $\mathcal{B}_{<t}$. Let C and R denote the resulting current-task and replay subsets, and set $\widehat{\rho}_t = |R|/b$ for this update.
3. form the mixed prompt batch $C \cup R$ with each prompt's task identity $m(x)$.
4. generate a fresh response $y \sim \pi_\theta(\cdot \mid x)$ for every prompt $x \in C \cup R$.
5. score each response with its task verifier, $r_{m(x)}(x, y)$.
6. apply one RLVR update to the mixed batch.
7. refresh the stored record and latest pass rate for each observed prompt, retaining at most one record per prompt.

Output: updated policy $\pi_{\theta'}$ and replay store $\mathcal{B}_{\le t}$.

Replay store and selector. CPR stores one record per prompt and refreshes its pass rate whenever the prompt reappears. The pass rate is the mean verifier score over the five responses generated for that prompt in its latest update.

The default selector admits previous-task prompts whose latest pass rate lies in $[0.24, 0.70]$. A selected prompt becomes eligible again after five updates. Among eligible prompts, CPR prioritizes those closest to pass rate 0.5.

For one update during stage t, let $n_{t,k}$ be the number of replayed prompts drawn from task $k < t$, and let $n_t = \sum_{\ell<t} n_{t,\ell}$. When $n_t > 0$, $q_{t,k} = n_{t,k}/n_t$ and $\sum_{k<t} q_{t,k} = 1$, so $q_{t,k}$ is the fraction of replayed prompts in that update drawn from task k. When $n_t = 0$, $\widehat{\rho}_t = 0$ and the replay term is omitted. If fewer than b_R prompts are eligible, then $0 < \widehat{\rho}_t < \rho$ whenever at least one prompt is replayed.

B.3 CPR Resource Overhead

At each CPR update, previous-task prompts replace current-task prompts within the same 512-prompt batch used by Seq. RLVR, and the current policy draws five responses per prompt. Both methods therefore generate 2,560 rollouts per update. CPR changes the task-sampling distribution without increasing the number of sampled prompts or current-policy rollouts. Its method-specific work is replay selection and CPU-side store access.

Replay-store footprint. CPR stores one record per observed prompt containing the prompt, task and verifier metadata, latest pass rate, and selector state. The serialized footprint is 1.24 KiB per prompt on average across the five CRG settings (Table 8).

Table 8: **Average serialized footprint per CPR replay record.** Each row reports the mean over prompts in one CRG setting. The final row weights the five settings equally.

Setting	Per prompt (KiB)
LLM Algorithmic	1.02
LLM Algebra	1.01
VLM Quantitative	1.42
VLM Spatial	1.39
VLM Positional	1.34
Mean	1.24

Runtime. On eight NVIDIA B200 GPUs, CPR averages 14.8 hours per 500-update run, compared with 14.6 hours for Seq. RLVR (Table 9).

Table 9: **Average wall-clock time per 500-update run on eight NVIDIA B200 GPUs.**

Method	Mean runtime (h)
Seq. RLVR	14.6
CPR	14.8

C Theoretical Analysis

C.1 Proof of Proposition 1

Adding and subtracting P_i and A_i writes each per-task final score as

$$F_i = B_i + (P_i - B_i) + (A_i - P_i) + (F_i - A_i), \tag{9}$$

and averaging over the T tasks gives

$$\begin{aligned} \mathrm{FinalAvg} &= \mathrm{BaseAvg} + \frac{1}{T}\sum_{i=1}^{T}(P_i - B_i) + \mathrm{TLG} + \frac{1}{T}\sum_{i=1}^{T}(F_i - A_i) \\ &= \mathrm{BaseAvg} + \frac{1}{T}\sum_{i=2}^{T}(P_i - B_i) + \mathrm{TLG} + \frac{1}{T}\sum_{i=1}^{T-1}(F_i - A_i), \end{aligned} \tag{10}$$

where the boundary terms $P_1 - B_1$ and $F_T - A_T$ vanish by the conventions of Appendix A.2. Matching the $\frac{1}{T}$ averaging to the $\frac{1}{T-1}$ normalization of the transfer terms,

$$\frac{1}{T}\sum_{i=2}^{T}(P_i - B_i) = \frac{T-1}{T}\,\mathrm{FWT}, \qquad \frac{1}{T}\sum_{i=1}^{T-1}(F_i - A_i) = \frac{T-1}{T}\,\mathrm{BWT}, \tag{11}$$

which yields the identity of Proposition 1.

C.2 First-Order CPR Analysis

By Proposition 1, any FinalAvg difference between methods with the same BaseAvg must pass through FWT, TLG, or BWT. We therefore compare one CPR update with a compute-matched baseline update and ask when replay changes these terms favorably.

Consider one update in stage t at pre-update parameters θ. Let $g_i^{(t)} = \nabla_\theta J_i(\theta)$ be the objective gradient for task i, and let $g_k^{(t)}$ be the update direction induced by prompts from task k. CPR combines prompts from the current task t with prompts from previous tasks $k < t$ through the direction g_t^{CPR} in Eq. (6). A first-order expansion of task i under CPR gives

$$\begin{aligned}\Delta_{i,\mathrm{CPR}}^{(t)} &:= J_i(\theta + \eta g_t^{\mathrm{CPR}}) - J_i(\theta) \\ &= \eta(1-\widehat{\rho}_t)\langle g_i^{(t)}, g_t^{(t)}\rangle + \eta\widehat{\rho}_t \sum_{k<t} q_{t,k}\langle g_i^{(t)}, g_k^{(t)}\rangle + O(\eta^2).\end{aligned} \tag{12}$$

The Seq. RLVR baseline uses only the current-task direction, so

$$\Delta_{i,\mathrm{seq}}^{(t)} = \eta\langle g_i^{(t)}, g_t^{(t)}\rangle + O(\eta^2). \tag{13}$$

Subtracting Eq. (13) from Eq. (12) yields the central comparison:

$$\Delta_{i,\mathrm{CPR}}^{(t)} - \Delta_{i,\mathrm{seq}}^{(t)} = \eta\widehat{\rho}_t \left[\sum_{k<t} q_{t,k}\langle g_i^{(t)}, g_k^{(t)}\rangle - \langle g_i^{(t)}, g_t^{(t)}\rangle\right] + O(\eta^2). \tag{14}$$

The bracket compares the average direction recovered from previous-task prompts with the current-task direction that those prompts replace. CPR improves task i to first order when this difference is positive after the relevant updates are averaged.

If every task pair had non-negative alignment, the current-task direction would already be non-harmful to every objective at first order. Mean alignment is weaker. The current direction may interfere with specific objectives even when the average over previous-task directions and relevant objectives is positive. CPR realizes that average at the current parameters by regenerating responses to previous-task prompts. Assumption 1 gives a non-negative expected first-order contribution under a reported term's weighting. Equation (14) gives the corresponding CPR–Seq. RLVR difference.

The position of task i relative to stage t determines which decomposition term receives the local effect. Setting $i = t$ gives the contribution during direct training and hence TLG. Tasks with $i < t$ give the contribution to BWT, while tasks with $i > t$ give the contribution to FWT. The replay pool controls the task mixture $q_{t,k}$. The replay ratio controls the scale $\widehat{\rho}_t$. The replay-pool ablation probes $q_{t,k}$. The ratio sweep in Appendix D.3 tests robustness to the configured replay scale ρ.

C.3 COROLLARY PROOFS

We first expand the expectation in Assumption 1. For one reported term, let α_t denote its non-negative stage weights, $p_{t,i}$ its non-negative task-objective weights, and $w_{t,k}$ the realized sampling frequencies of prompts from task k. Each set of weights is normalized over its support. For an update mixture with these weights, define $\Delta_i^{(t)} = J_i(\theta + \eta \sum_k w_{t,k} g_k^{(t)}) - J_i(\theta)$. Then

$$\mathbb{E}_{t,i,k}\left[\langle g_i^{(t)}, g_k^{(t)} \rangle\right] := \sum_t \alpha_t \sum_i p_{t,i} \sum_k w_{t,k} \mathbb{E}_{\text{upd}}\left[\langle g_i^{(t)}, g_k^{(t)} \rangle\right], \tag{15}$$

where $\mathbb{E}_{\text{upd}}$ averages prompt, selector, rollout, and update stochasticity. Averaging the corresponding first-order task changes gives

$$\sum_t \alpha_t \sum_i p_{t,i} \mathbb{E}_{\text{upd}}\left[\Delta_i^{(t)}\right] = \eta \sum_t \alpha_t \sum_i p_{t,i} \sum_k w_{t,k} \mathbb{E}_{\text{upd}}\left[\langle g_i^{(t)}, g_k^{(t)} \rangle\right] + O(\eta^2). \tag{16}$$

For Corollary 1, the Seq. RLVR baseline has $w_{t,t} = 1$. BWT uses later stages and earlier-task objectives, so its first-order contribution is proportional to

$$\sum_t \alpha_t \sum_{i<t} p_{t,i} \mathbb{E}_{\text{upd}}\left[\langle g_i^{(t)}, g_t^{(t)} \rangle\right]. \tag{17}$$

This is Assumption 1 under BWT's stage and task weighting with $k = t$. Its non-negativity proves a non-negative expected first-order contribution to earlier-task objectives even when individual (i, t) pairs are negative.

For Corollary 2, CPR assigns $w_{t,t} = 1 - \widehat{\rho}_t$ and $w_{t,k} = \widehat{\rho}_t q_{t,k}$ for $k < t$. Applying Eq. (16) with $i = t$ gives the first-order contribution to TLG. Restricting the task weights to $i < t$ gives BWT, while restricting them to $i > t$ gives FWT. Whenever Assumption 1 holds under one of these weightings, the corresponding right-hand side is non-negative. This proves the corollary for that term. Equation (14) gives the corresponding CPR–Seq. RLVR difference.

D Additional Results

D.1 Full Method Results

Method rankings vary substantially across task sequences. Figure 10 compares CTM for all eight sequential methods. Figure 11 compares the exact FinalAvg decompositions of the Seq. RLVR baseline versus CPR for each task sequence. Tables 10–14 report the complete results.

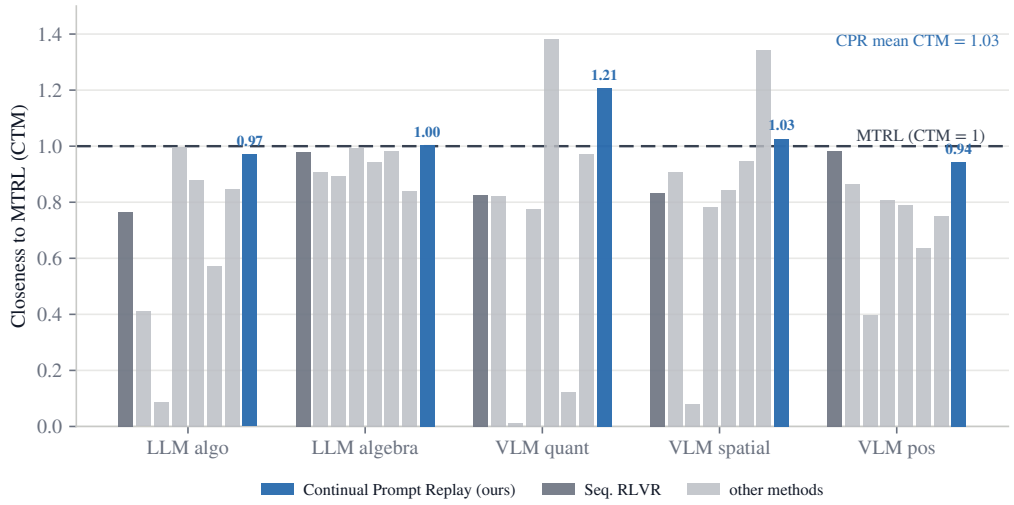


Figure 10: **CPR is the only evaluated CL method that reaches MTRL-level performance on average.** Bars show CTM for eight sequential methods in each setting. Seq. RLVR is dark gray. The six other interventions are light gray. CPR is blue. This level is reached or exceeded for LLM algebra, VLM quantitative reasoning, and VLM spatial reasoning. Its mean CTM is 1.03, compared with 0.88 for Seq. RLVR.

The same FinalAvg difference can reflect different changes in FWT, TLG, and BWT. Figure 11 compares the exact FinalAvg decomposition for the Seq. RLVR baseline and CPR for each task sequence.



Figure 11: **CPR's FinalAvg gains arise from FWT, TLG, or both.** Each pair compares Seq. RLVR with CPR. Stacked bars show the four terms BaseAvg, $\frac{T-1}{T}$FWT, TLG, $\frac{T-1}{T}$BWT. Black markers show FinalAvg. The BWT effect varies across task sequences. On VLM quantitative reasoning, larger FWT and TLG outweigh a more negative BWT contribution.

Table 10: **Complete results for LLM algorithmic reasoning.** Base reports pre-training accuracy. Final reports endpoint accuracy. Final−Base, FWT, TLG, BWT use percentage points. The largest reported value in each performance column is bold.

Role	Method	Model	Steps	Base (%)	Final (%)	Final−Base (%)	FWT (%)	TLG (%)	BWT (%)
baseline	Seq. RLVR	Qwen3-4B	500	17.5	49.9	+32.3	+17.2	+15.4	**+1.6**
MTRL reference	MTRL	Qwen3-4B	500	17.5	**65.3**	**+47.7**	–	–	–
CL method	FIRE	Qwen3-4B	500	17.5	5.6	-11.9	+10.0	+12.9	-37.5
CL method	ReDo	Qwen3-4B	500	17.5	26.8	+9.2	+9.5	+4.6	-4.4
CL method	EWC	Qwen3-4B	500	17.5	65.0	+47.5	+19.9	**+30.1**	-0.6
CL method	Muon	Qwen3-4B	500	17.5	57.4	+39.9	+20.0	+23.7	-2.0
CL method	OSFT	Qwen3-4B	500	17.5	37.3	+19.8	+12.5	+18.9	-11.5
CL method	CPR	Qwen3-4B	500	17.5	63.3	+45.8	**+20.7**	+26.0	+1.2
CL method	KL-to-old-policy	Qwen3-4B	500	17.5	55.4	+37.8	+20.1	+23.4	-4.0

Table 11: **Complete results for LLM algebra.** Base reports pre-training accuracy. Final reports endpoint accuracy. Final−Base, FWT, TLG, BWT use percentage points. The largest reported value in each performance column is bold.

Role	Method	Model	Steps	Base (%)	Final (%)	Final−Base (%)	FWT (%)	TLG (%)	BWT (%)
baseline	Seq. RLVR	Qwen3-4B	500	43.2	88.2	+45.1	+20.3	+29.1	-1.1
MTRL reference	MTRL	Qwen3-4B	500	43.2	90.0	+46.8	–	–	–
CL method	CPR	Qwen3-4B	500	43.2	**90.4**	**+47.2**	+19.6	+30.9	+0.1
CL method	OSFT	Qwen3-4B	500	43.2	88.4	+45.2	+23.6	+21.9	+4.3
CL method	FIRE	Qwen3-4B	500	43.2	80.3	+37.1	+15.0	**+33.0**	-10.0
CL method	ReDo	Qwen3-4B	500	43.2	81.7	+38.5	+19.1	+23.2	-0.6
CL method	EWC	Qwen3-4B	500	43.2	89.3	+46.2	+19.7	+30.9	-1.3
CL method	KL-to-old-policy	Qwen3-4B	500	43.2	75.7	+32.6	+16.1	+4.3	**+17.7**
CL method	Muon	Qwen3-4B	500	43.2	84.8	+41.6	**+25.0**	+23.4	-3.2

Table 12: **Complete results for VLM quantitative reasoning.** Base reports pre-training accuracy. Final reports endpoint accuracy. Final−Base, FWT, TLG, BWT use percentage points. The largest reported value in each performance column is bold.

Role	Method	Model	Steps	Base (%)	Final (%)	Final−Base (%)	FWT (%)	TLG (%)	BWT (%)
baseline	Seq. RLVR	Qwen2.5-VL-7B	500	24.8	25.5	+0.7	+1.4	+5.1	-7.2
MTRL reference	MTRL	Qwen2.5-VL-7B	500	24.8	30.9	+6.1	–	–	–
CL method	CPR	Qwen2.5-VL-7B	500	24.8	37.2	+12.5	+13.2	**+16.0**	-18.0
CL method	ReDo	Qwen2.5-VL-7B	500	24.8	25.4	+0.6	-1.4	-3.0	+6.2
CL method	FIRE	Qwen2.5-VL-7B	500	24.8	0.4	-24.4	+0.1	-3.2	-28.3
CL method	EWC	Qwen2.5-VL-7B	500	24.8	24.0	-0.8	+2.9	-10.7	+10.3
CL method	Muon	Qwen2.5-VL-7B	500	24.8	**42.7**	**+17.9**	**+13.9**	+6.3	+1.6
CL method	OSFT	Qwen2.5-VL-7B	500	24.8	3.8	-21.0	-17.0	-11.3	+3.9
CL method	KL-to-old-policy	Qwen2.5-VL-7B	500	24.8	30.0	+5.3	+7.2	-15.2	**+20.0**

Table 13: **Complete results for VLM spatial reasoning.** Base reports pre-training accuracy. Final reports endpoint accuracy. Final−Base, FWT, TLG, BWT use percentage points. The largest reported value in each performance column is bold.

Role	Method	Model	Steps	Base (%)	Final (%)	Final−Base (%)	FWT (%)	TLG (%)	BWT (%)
baseline	Seq. RLVR	Qwen2.5-VL-7B	500	24.1	26.3	+2.2	+7.4	-2.7	-1.5
MTRL reference	MTRL	Qwen2.5-VL-7B	500	24.1	31.6	+7.5	–	–	–
CL method	CPR	Qwen2.5-VL-7B	500	24.1	32.5	+8.4	**+9.9**	+4.0	-4.6
CL method	ReDo	Qwen2.5-VL-7B	500	24.1	28.7	+4.7	+1.9	-3.7	**+8.1**
CL method	FIRE	Qwen2.5-VL-7B	500	24.1	2.5	-21.6	-1.7	-6.5	-16.4
CL method	EWC	Qwen2.5-VL-7B	500	24.1	24.7	+0.6	-7.7	+8.7	-2.0
CL method	Muon	Qwen2.5-VL-7B	500	24.1	26.7	+2.6	-4.2	+1.4	+5.7
CL method	OSFT	Qwen2.5-VL-7B	500	24.1	30.0	+5.9	+1.2	-1.4	+7.6
CL method	KL-to-old-policy	Qwen2.5-VL-7B	500	24.1	**42.4**	**+18.4**	+7.7	**+14.5**	-3.1

Table 14: **Complete results for VLM positional reasoning.** Base reports pre-training accuracy. Final reports endpoint accuracy. Final−Base, FWT, TLG, BWT use percentage points. The largest reported value in each performance column is bold.

Role	Method	Model	Steps	Base (%)	Final (%)	Final−Base (%)	FWT (%)	TLG (%)	BWT (%)
baseline	Seq. RLVR	Qwen2.5-VL-7B	500	28.7	34.7	+6.0	-8.9	**+15.8**	-4.2
MTRL reference	MTRL	Qwen2.5-VL-7B	500	28.7	**35.3**	**+6.6**	–	–	–
CL method	CPR	Qwen2.5-VL-7B	500	28.7	33.2	+4.5	-11.3	+10.4	**+3.4**
CL method	ReDo	Qwen2.5-VL-7B	500	28.7	30.5	+1.7	**+0.7**	+2.3	-1.6
CL method	FIRE	Qwen2.5-VL-7B	500	28.7	14.0	-14.8	-21.8	+6.5	-6.6
CL method	Muon	Qwen2.5-VL-7B	500	28.7	27.9	-0.8	-8.2	+3.6	+2.4
CL method	EWC	Qwen2.5-VL-7B	500	28.7	28.5	-0.2	-7.6	+5.5	+0.0
CL method	OSFT	Qwen2.5-VL-7B	500	28.7	22.5	-6.2	-4.0	-5.2	+2.6
CL method	KL-to-old-policy	Qwen2.5-VL-7B	500	28.7	26.5	-2.2	-9.6	+10.8	-7.8

D.2 Additional CPR Ablation Results

We use two ablations to isolate CPR's replay mechanism. Table 15 tests current-policy regeneration. Table 16 changes the replay pool.

Table 15: **Current-policy regeneration drives CPR's replay gain.** On LLM algorithmic reasoning at 500 steps and $\rho = 0.5$, sample replay reuses earlier-policy trajectories with importance sampling. CPR regenerates responses with the current policy. Delta rows are relative to no replay and are computed from unrounded values. The best value among the three sequential variants in each metric is bold.

Variant	FinalAvg (%)	FWT (%)	TLG (%)	BWT (%)
No replay (Seq. RLVR)	49.9	+17.2	+15.4	+1.6
Sample replay	47.5	+19.6	+10.3	**+2.3**
CPR	**63.3**	**+20.7**	**+26.0**	+1.2
MTRL reference	65.3	–	–	–
Δ Sample replay $-$ no replay	-2.4	+2.4	-5.1	+0.6
Δ CPR $-$ no replay	+13.5	+3.5	+10.7	-0.4

Table 16: **Replaying prompts from all previous tasks gives the highest mean FinalAvg.** No replay corresponds to Seq. RLVR. Replay from only the immediately previous task is reported for LLM algorithmic reasoning. Bold values mark the maximum within each setting and among the mean rows.

Setting	Replay pool	FinalAvg (%)	CTM	FWT (%)	TLG (%)	BWT (%)
LLM algorithmic	No replay	49.9	0.76	+17.2	+15.4	**+1.6**
LLM algorithmic	Immediately previous task	**66.3**	**1.02**	+20.8	**+30.1**	+0.0
LLM algorithmic	All previous tasks (CPR)	63.3	0.97	+20.7	+26.0	+1.2
LLM algorithmic	Previous and current tasks	64.3	0.98	**+20.9**	+29.1	−1.3
LLM algebra	No replay	88.2	0.98	+20.3	+29.1	−1.1
LLM algebra	All previous tasks (CPR)	90.4	1.00	+19.6	+30.9	**+0.1**
LLM algebra	Previous and current tasks	**91.2**	**1.01**	**+20.8**	**+31.3**	−0.8
VLM quantitative	No replay	25.5	0.83	+1.4	+5.1	−7.2
VLM quantitative	All previous tasks (CPR)	**37.2**	**1.21**	**+13.2**	**+16.0**	−18.0
VLM quantitative	Previous and current tasks	36.5	1.18	+10.9	+3.4	**+0.1**
VLM spatial	No replay	26.3	0.83	+7.4	−2.7	−1.5
VLM spatial	All previous tasks (CPR)	**32.5**	**1.03**	**+9.9**	**+4.0**	−4.6
VLM spatial	Previous and current tasks	29.8	0.94	+6.1	−0.1	**+0.9**
VLM positional	No replay	**34.7**	**0.98**	−8.9	**+15.8**	−4.2
VLM positional	All previous tasks (CPR)	33.2	0.94	−11.3	+10.4	**+3.4**
VLM positional	Previous and current tasks	31.1	0.88	**−4.3**	+6.7	−1.4
Mean	No replay	44.9	0.88	+7.5	+12.5	−2.5
Mean	All previous tasks (CPR)	**51.3**	**1.03**	+10.4	**+17.5**	−3.6
Mean	Previous and current tasks	50.6	1.00	**+10.9**	+14.1	**−0.5**

Replaying prompts from all previous tasks raises mean FinalAvg from 44.9% to 51.3%. Allowing current-task prompts reaches 50.6%.

Success-rate filtering does not explain CPR's gain in this setting. Figure 12 compares CPR with a matched variant that removes the success-rate threshold from prompt eligibility. The variant reaches 63.2% FinalAvg, essentially matching CPR at 63.3% and leaving a difference of only 0.1 percentage points. Under this comparison, the CPR improvement cannot be attributed to success-rate filtering alone.

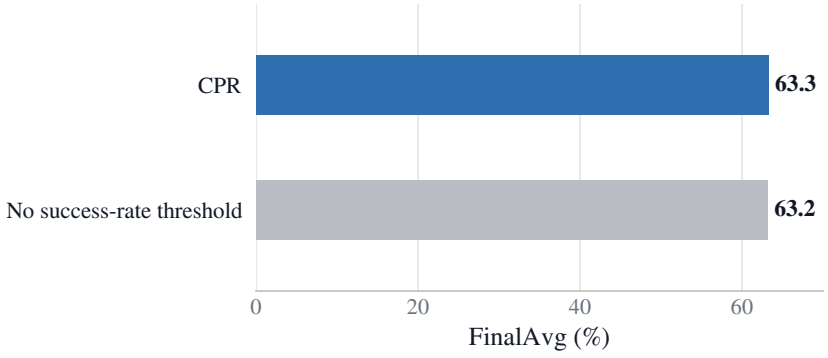


Figure 12: **Removing the success-rate threshold leaves CPR's endpoint nearly unchanged in the matched ablation.** FinalAvg on LLM algorithmic reasoning is 63.2% without the success-rate threshold and 63.3% with CPR.

D.3 Robustness to Replay Ratio ρ

CPR remains near MTRL across a tenfold range of nonzero replay ratios. Holding all other settings fixed on LLM algorithmic reasoning, $\rho \in \{0.05, 0.20, 0.50\}$ gives CTM between 0.97 and 0.99.

Table 17: **CPR is robust to replay ratio ρ on LLM algorithmic reasoning.** With Qwen3-4B, every tested nonzero ratio reaches CTM 0.97–0.99, compared with 0.76 without replay. FinalAvg varies by only 1.3 percentage points across the nonzero ratios. The best sequential value in each metric is bold.

Replay ratio ρ	FinalAvg (%)	CTM	FWT (%)	TLG (%)	BWT (%)
0.00 (Seq. RLVR)	49.9	0.76	+17.2	+15.4	**+1.6**
0.05	**64.5**	**0.99**	**+22.7**	+28.9	−2.6
0.20	63.2	0.97	+20.1	**+29.2**	−1.8
0.50	63.3	0.97	+20.7	+26.0	+1.2
MTRL reference	65.3	1.00	–	–	–

D.4 CPR Gains Accompany Both Entropy Increases and Decreases

We test whether lower final-policy entropy consistently accompanies CPR's performance gains. For each method's final policy, we measure generated-token predictive entropy on a fixed prompt set under greedy decoding. The reported value is the token-weighted mean over generated response tokens, in nats. Figure 13 compares CPR with the Seq. RLVR baseline within each setting.

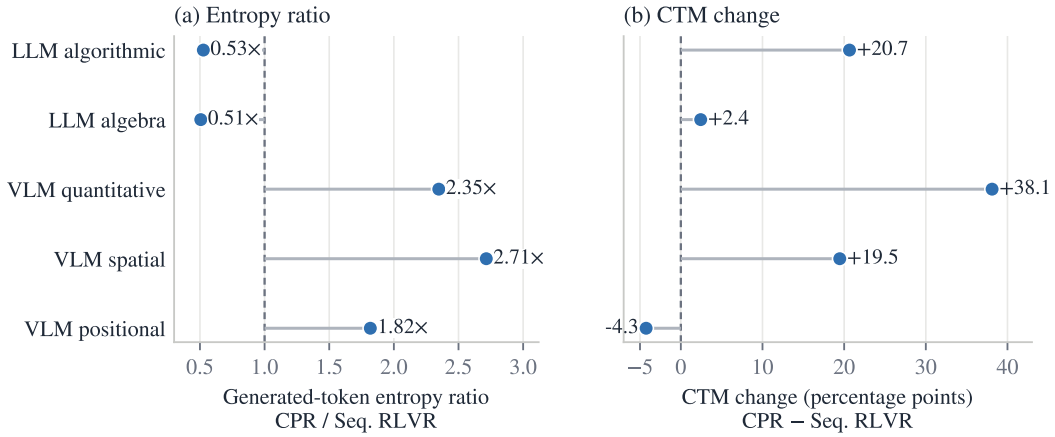


Figure 13: **CPR gains accompany both entropy increases and decreases.** Each row pairs CPR with Seq. RLVR in the same setting. The left panel reports the ratio of final-policy generated-token entropy, measured on fixed prompts under greedy decoding. The right panel reports the corresponding CTM change. Dashed lines mark no change. Among the four settings where CPR improves CTM, entropy decreases in both LLM settings and increases in VLM quantitative and spatial reasoning.

CPR gains occur under opposite entropy changes. Entropy falls by 47–49% in the two LLM settings, where CTM rises by 20.7 and 2.4 percentage points. In VLM quantitative and spatial reasoning, entropy rises to 2.35 and 2.71 times the corresponding baseline level while CTM rises by 38.1 and 19.5 points. CPR's gains therefore do not consistently coincide with lower final-policy entropy.

D.5 Additional Evidence for Shared Reasoning

We provide additional diagnostics for both roles of shared reasoning. We compare the current-task direction with a mixture that includes previous-task directions to test the mechanism CPR uses to improve FWT and TLG. We then examine task-gradient alignment across domains. With the jugs trajectory, we connect later-task support to modest forgetting and earlier-task support to forward transfer at the behavioral level.

D.5.1 Gradient Alignment Diagnostics

We use two diagnostics. First, at the Qwen3-4B base model, we compare the current-task gradient with a mixture that includes previous-task gradients. Second, we compare task-gradient alignment across domains.

Adding previous-task gradients improves alignment with the all-task mean. At the Qwen3-4B base model $\theta^\star$, we evaluate all ten LLM algorithmic task gradients and write

$$g_k^\star = \nabla J_k(\theta^\star), \qquad \bar{g}^\star = \frac{1}{10}\sum_{i=1}^{10} g_i^\star. \tag{18}$$

Each $g_k^\star$ averages two mini-batch gradients from the language-model head and final two transformer blocks. Combining all eight FSDP shards gives $201{,}861{,}632$ coordinates. The parameters remain fixed throughout collection.

For each stage $t \in \{2, \ldots, 10\}$ in the LLM algorithmic sequence, we compare the current-task direction with an equal-weight previous-task mixture:

$$g_t^{\text{current}} = g_t^\star, \qquad g_t^{\text{mix}} = 0.5 g_t^\star + \frac{0.5}{t-1}\sum_{k<t} g_k^\star. \tag{19}$$

Only current-task and previous-task gradients enter g_t^{mix}. We evaluate both directions by their alignment with the all-task mean $\bar{g}^\star$. We use equal weighting to isolate the contribution of previous-task directions from CPR's prompt selector.

Table 18 reports cosine and inner product with $\bar{g}^\star$. Across stages 2 through 10, mean cosine rises from 0.43 to 0.79. Mean inner product rises by 18.2%, while its standard deviation falls by 55.3%. The mixture has the higher cosine in all nine comparisons and the higher raw inner product in seven.

Table 18: **Adding previous-task gradients improves alignment with the all-task mean at the Qwen3-4B base model.** Each row uses the same fixed set of ten task gradients. For each stage, the mixture assigns half of its weight to the current task and distributes the other half equally across previous tasks. The all-task mean $\bar{g}^\star$ serves only as the evaluation direction. Inner products are reported in units of 10^{-5}.

		Cosine to $\bar{g}^\star$		Inner product (10^{-5})	
Stage	Current task	Current	Mixture	Current	Mixture
2	Binary Alternation	0.410	0.628	1.122	5.421
3	Binary Matrix	0.467	0.679	7.811	6.616
4	Caesar Cipher	0.351	0.804	1.384	3.801
5	Cryptarithm	0.372	0.811	0.964	2.987
6	Isomorphic Strings	0.751	0.849	17.877	11.039
7	Jugs	0.391	0.905	1.619	4.049
8	Matrix Rotation	0.360	0.761	3.394	4.590
9	String Manipulation	0.422	0.780	4.125	4.806
10	A::B Rewriting	0.344	0.929	1.051	3.193
Mean		0.430	0.794	4.372	5.167
Std.		0.120	0.092	5.217	2.330

Shared LLM geometry weakens across domains. Figure 14 combines separately collected CPR diagnostics for the ten algorithmic tasks and six algebra tasks using the same model and parameter subset. Across the 60 cross-domain task pairs, mean cosine is -0.01. The cosine between the two domain-average gradient vectors is 0.006. Both are close to zero, placing the measured algorithmic–algebra geometry near orthogonality. This panel characterizes cross-domain geometry. The comparison above instead measures how adding previous-task gradients at the Qwen3-4B base model changes alignment with the all-task mean.

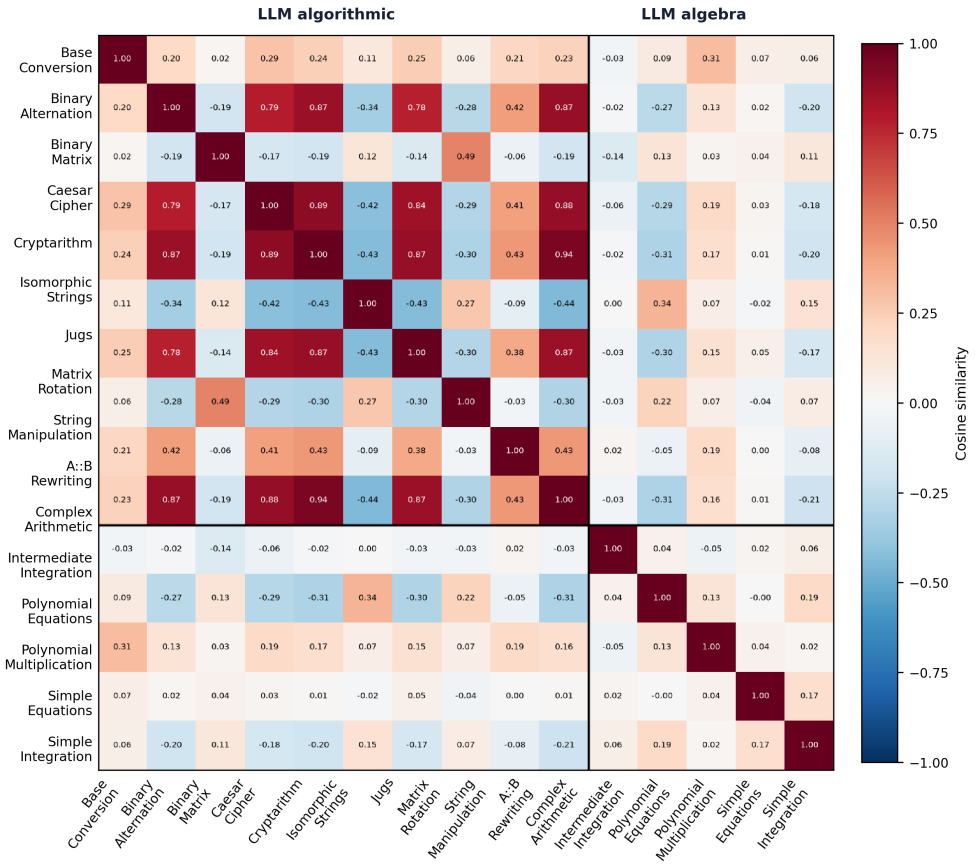


Figure 14: **LLM gradient alignment weakens across the algorithmic–algebra boundary.** Pairwise cosines are shown for the ten algorithmic tasks and six algebra tasks. Black lines mark the domain boundary. Cross-domain task pairs have mean cosine -0.01, while the cosine between the two domain-average vectors is 0.006. The panel combines separately collected CPR diagnostics.

Measured VLM gradients remain positively aligned across domains. Figure 15 combines CPR diagnostics collected separately on the quantitative, spatial, and positional domains. Every displayed off-diagonal cosine is positive, including all cross-domain blocks. Comparing these positive VLM cosines with the near-zero LLM geometry, we find that cross-task alignment varies with the reasoning domain.

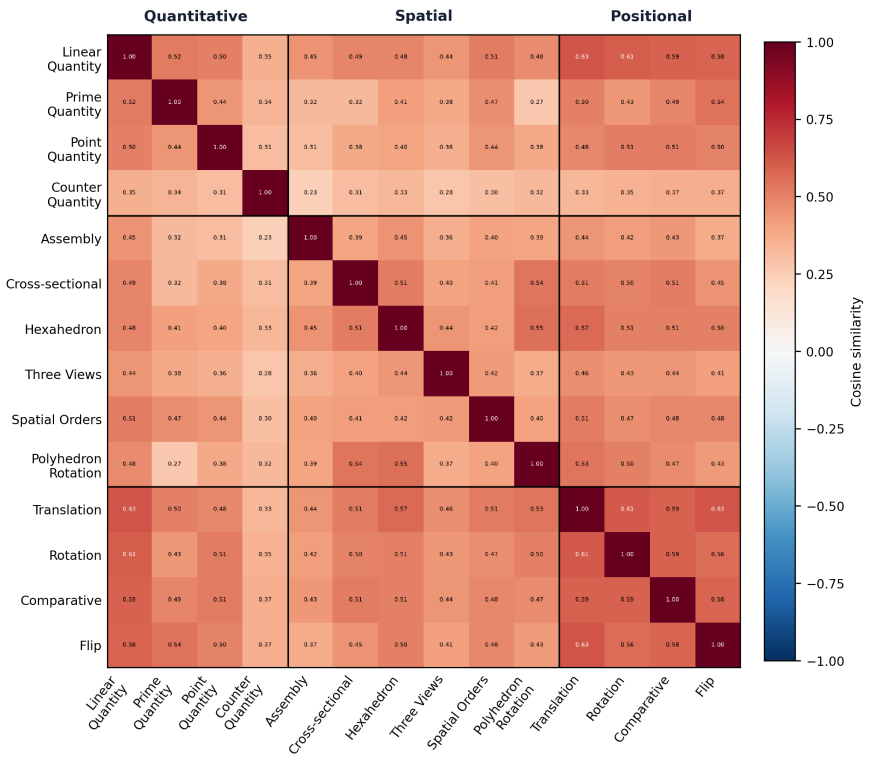


Figure 15: **The measured VLM task gradients are positively aligned across all three domains.** The matrix combines separately collected CPR diagnostics for quantitative, spatial, and positional reasoning. Every displayed off-diagonal task pair has positive cosine, including pairs that cross domain boundaries. Black lines mark those boundaries.

At the frozen LLM checkpoint, adding previous-task directions increases average alignment to the all-task direction while reducing its variation across stages 2 through 10. Across domains, measured task-gradient alignment ranges from near-orthogonal algorithmic–algebra geometry to uniformly positive VLM geometry. We use this contrast to characterize the strength of shared reasoning as domain-dependent.

D.5.2 JUGS REASONING TRAJECTORY

We use the jugs trajectory to distinguish procedural reasoning from verifier-valid answer generation across four checkpoints from the same Seq. RLVR run on the LLM algorithmic stream (Figure 5). Before any RLVR, the base model explores alternative routes, derives the optimal 10-operation B-to-A route, and then continues to recheck it until generation ends without an answer. The pre-jugs policy abandons an initial direction, reaches A=3 through a longer 28-operation route, and likewise emits no answer. After direct jugs training, the model derives the optimal route, explicitly checks all 10 states, and emits a valid answer. The final policy produces the same answer after later-task training while reducing the reasoning trace from 1,631 to 780 characters and omitting the separate rechecking pass.

Raw transcripts. The four outputs summarized in Figure 5 appear below verbatim.

Base model / before RLVR *jugs, LLM algorithmic*

```
<think>
Okay, let's see. I need to figure out how to get exactly 3 liters in any of the jugs using
     the given moves. The jugs are A (10L), B (11L), and C (11L). The target is 3 liters. All
      start empty.

First, I remember that these types of puzzles usually involve filling, emptying, and pouring
     between jugs to reach the desired amount. Since the target is 3, which is smaller than
```

```
     the capacities of all jugs, maybe I need to find a way to measure it through some
     combination of pours.

Let me think about possible steps. Let's start by trying to fill the largest jugs first.
     Maybe fill B or C first. But since they are 11L, which is more than 3, maybe that's not
     directly helpful. Wait, but if I can pour from a full jug into another, maybe I can get
     the right amount.

Wait, the target is 3. Let me think of possible combinations. For example, if I can get 11 -
     8 = 3, but how to get 8? Alternatively, maybe using the 10L jug. If I fill A (10L) and
     pour into B (11L), but B is empty. So pour A into B. Then B has 10L, A is empty. Then
     fill A again, pour into B. B can take 1L more (since it's 11L). So after pouring, B is
     full (11L), and A has 10 - 1 = 9L left. Then empty B. Now, pour the 9L from A into B.
     Now B has 9L, A is empty. Then fill A again (10L), pour into B. B can take 2 more liters
      (11-9=2). So after pouring, B is 11L, A has 10-2=8L. Then empty B again. Now pour A's 8
     L into B. B has 8L. Fill A again (10L), pour into B. B can take 3 more liters (11-8=3).
     So after pouring, A has 10-3=7L, and B is full. But that's not helpful. Wait, but maybe
     I can use another jug. Alternatively, maybe using C instead of B.

Alternatively, maybe using the 10L and 11L jugs. Let me think. If I fill B (11L) and pour
     into A (10L). Then B has 1L left. Then empty A. Now pour the 1L from B into A. Now A has
      1L. Then fill B again (11L), pour into A until A is full. A can take 9 more liters. So
     B would have 11 - 9 = 2L left. Then empty A again. Pour the 2L from B into A. Now A has
     2L. Fill B again (11L), pour into A. A can take 8 more liters, so B would have 11 - 8 =
     3L. There we go! So B would have 3L. That's the target. Let me check the steps again.

So steps would be:

1. Fill B (11L) → B=11, A=0, C=0.
2. Pour B into A until A is full. A can take 10L, so B has 11-10=1L left. Now A=10, B=1, C=0.
3. Empty A → A=0, B=1, C=0.
4. Pour B into A. Now A=1, B=0, C=0.
5. Fill B again → B=11, A=1, C=0.
6. Pour B into A until A is full. A can take 9 more liters. So B becomes 11-9=2L. Now A=10, B
     =2, C=0.
7. Empty A → A=0, B=2, C=0.
8. Pour B into A. Now A=2, B=0, C=0.
9. Fill B again → B=11, A=2, C=0.
10. Pour B into A until A is full. A can take 8 liters, so B becomes 11-8=3L. Now B=3, which
     is the target.

So the moves would be:

fill B, pour B->A, empty A, pour B->A, fill B, pour B->A, empty A, pour B->A, fill B, pour B
     ->A.

But let me
```

Before jugs training / step 300 *jugs, LLM algorithmic*

```
<think>
Okay, let's see. I need to figure out how to get exactly 3 units of water in any of the jugs
     using the given operations. The jugs are A (10L), B (11L), and C (11L). The target is 3L
     . All start empty.

First, I remember that these types of puzzles usually involve filling and transferring water
     between jugs to reach the desired amount. Since the target is 3, which is less than the
     capacities of all jugs, maybe I can use the smaller jugs to measure it.

Let me think. The jugs are 10, 11, and 11. The target is 3. So maybe I can use the 11L jugs
     to measure. For example, if I fill one of the 11L jugs and pour into the 10L jug, that
     would leave 1L in the 11L jug. Then, if I pour that 1L into the other 11L jug, maybe I
     can do something else. Wait, but that's not getting me to 3.

Wait, maybe I need to use the 10L jug. Let me try a different approach. Let's think of
     possible steps:

1. Fill the 10L jug (A) to 10L.
2. Pour from A to B (11L). Since B is empty, it can take all 10L. Now A is empty, B has 10L.
3. Fill A again (10L).
4. Pour from A to B. B can take 1L more (since it has 10L, and B is 11L). So after pouring, B
      is full (11L), and A has 10 - 1 = 9L left.
5. Now empty B (11L) to 0L.
6. Pour the 9L from A into B. Now B has 9L, A is empty.
7. Fill A again (10L).
8. Pour from A to B. B can take 2L (since it has 9L, needs 2 more to reach 11). So after
     pouring, B is 11L, A has 10 - 2 = 8L.
9. Empty B again.
10. Pour the 8L from A into B. Now B has 8L, A is empty.
```

```
11. Fill A again (10L).
12. Pour from A to B. B can take 3L (since it has 8L, needs 3 more to reach 11). So after
     pouring, B is 11L, A has 10 - 3 = 7L.
13. Empty B again.
14. Pour the 7L from A into B. Now B has 7L, A is empty.
15. Fill A again (10L).
16. Pour from A to B. B can take 4L (7 + 4 = 11). So A has 10 - 4 = 6L.
17. Empty B again.
18. Pour the 6L from A into B. Now B has 6L, A is empty.
19. Fill A again (10L).
20. Pour from A to B. B can take 5L (6 + 5 = 11). A has 10 - 5 = 5L.
21. Empty B again.
22. Pour the 5L from A into B. Now B has 5L, A is empty.
23. Fill A again (10L).
24. Pour from A to B. B can take 6L (5 + 6 = 11). A has 10 - 6 = 4L.
25. Empty B again.
26. Pour the 4L from A into B. Now B has 4L, A is empty.
27. Fill A again (10L).
28. Pour from A to B. B can take 7L (4 + 7 = 11). A has 10 - 7 = 3L.

Ah, there we go. Now A has 3L. So the steps would be:

Fill A, pour to B, fill A, pour to B, empty B, pour A to B, fill A, pour to B, empty B, pour
     A to B, fill A, pour to B, empty B, pour A to B, fill A, pour to B, empty B, pour A to B
     , fill A, pour to B, empty B, pour A to B, fill A, pour to B.
```

After jugs training / step 350 *jugs, LLM algorithmic*

```
<think>
Okay, let's see. I need to figure out how to get exactly 3 gallons in any of the jugs. The
     jugs are A (10), B (11), and C (11). The target is 3. So, the possible moves are fill,
     empty, and pour.

First, I should think about the possible combinations. Since the jugs are 10, 11, and 11,
     maybe I can use the 11-liter jugs to measure out the 3. Let me try to think of steps.

Let me start by filling the 11-liter jug (B) first. Then, pour it into the 10-liter jug (A).
     That would leave 1 liter in B. Then, empty A. Now, pour the 1 liter from B into A. Now A
      has 1 liter, B is empty. Then, fill B again. Now, pour B into A until A is full. A can
     take 9 more, so B would have 11 - 9 = 2 left. Then empty A again. Now pour the 2 from B
     into A. Now A has 2. Fill B again. Now pour B into A. A can take 8 more, so B would have
      11 - 8 = 3. There we go. So the steps would be:

Fill B, pour B into A. Empty A, pour B into A. Fill B, pour B into A. Empty A, pour B into A.
      Then fill B again and pour into A. That would leave 3 in B. So the moves would be:

fill B
pour B->A
empty A
pour B->A
fill B
pour B->A
empty A
pour B->A
fill B
pour B->A

Wait, but that's a lot of steps. Let me check again. Let me track each step:

1. Fill B (11)
2. Pour B into A (A is 10, B is 1)
3. Empty A (A is 0, B is 1)
4. Pour B into A (A is 1, B is 0)
5. Fill B (11)
6. Pour B into A (A can take 9, so B has 2 left)
7. Empty A (A is 0, B is 2)
8. Pour B into A (A is 2, B is 0)
9. Fill B (11)
10. Pour B into A (A can take 8, so B has 3 left)

So after step 10, B has 3. So the moves would be the list of these steps. So the answer is
     the list of these moves.
</think>

<answer>["fill B", "pour B->A", "empty A", "pour B->A", "fill B", "pour B->A", "empty A", "
     pour B->A", "fill B", "pour B->A"]</answer>
```

Final policy / step 500 *jugs, LLM algorithmic*

```
<think>
Okay, let's see. I need to get exactly 3 gallons in any of the jugs. The jugs are A (10), B
     (11), and C (11). The moves are fill, empty, and pour. Let me think.

First, maybe start by filling the largest jug, like B or C. Let's try filling B. Then pour
     from B to A. Since A is 10, and B is 11, pouring B into A would fill A to 10, leaving 1
     in B. Then empty A. Now, pour the 1 from B to A. Now A has 1. Then fill B again. Now
     pour B into A. A can take 9 more, so B would have 11-9=2 left. Now B has 2. Then empty A
     . Pour B's 2 into A. Now A has 2. Fill B again. Pour B into A. A can take 8 more, so B
     would have 11-8=3. There we go, B has 3. So the moves would be: fill B, pour B->A, empty
      A, pour B->A, fill B, pour B->A, empty A, pour B->A, fill B, pour B->A. That gives B as
      3.
</think>

<answer>["fill B", "pour B->A", "empty A", "pour B->A", "fill B", "pour B->A", "empty A", "
     pour B->A", "fill B", "pour B->A"]</answer>
```

Together, we use these diagnostics to distinguish the two roles of shared reasoning. Positive mean gradient alignment persists under both Seq. RLVR and CPR, while its magnitude varies across domains. In the jugs trajectory, the learned solution persists through later training, and the final policy produces it with a shorter reasoning trace, consistent with modest forgetting. Success also improves before direct jugs training, illustrating how shared reasoning can benefit forward transfer.

D.6 PASS@K COMPARISON WITH THE BASE MODEL

We examine whether CPR improves both single-sample accuracy and solution coverage as the sampling budget increases. We compare Qwen3-4B before RLVR with the final 500-step CPR policy with current-task replay. Each policy produces 16 responses at temperature 0.7 for the same 64 held-out prompts per LLM algorithmic task. We estimate pass@1, pass@4, pass@8, and pass@16 from these responses following Chen et al. (2021).

Table 19: **CPR improves both pass@1 and higher-K success rates.** CPR with current-task replay outperforms the base model throughout the sampling curve. Both policies use the same 64 held-out prompts per task and sample 16 responses at temperature 0.7. The higher value within each task and K is bold.

Task	Policy	pass@1	pass@4	pass@8	pass@16
Base Conversion	Base	57.7	81.0	87.5	90.6
	CPR + current	**98.1**	**99.8**	**100.0**	**100.0**
Binary Alternation	Base	0.6	1.8	2.7	3.1
	CPR + current	**43.1**	**55.4**	**60.1**	**64.1**
Binary Matrix	Base	17.9	27.7	31.7	34.4
	CPR + current	**51.0**	**56.6**	**58.9**	**60.9**
Caesar Cipher	Base	0.2	0.8	1.6	3.1
	CPR + current	**5.8**	**9.5**	**10.2**	**10.9**
Cryptarithm	Base	0.0	0.0	0.0	0.0
	CPR + current	0.0	0.0	0.0	0.0
Isomorphic Strings	Base	59.4	88.7	95.0	98.4
	CPR + current	**99.9**	**100.0**	**100.0**	**100.0**
Jugs	Base	0.0	0.0	0.0	0.0
	CPR + current	**51.5**	**53.1**	**53.1**	**53.1**
Matrix Rotation	Base	33.2	38.9	42.1	45.3
	CPR + current	**97.6**	**99.6**	**99.9**	**100.0**
String Manipulation	Base	4.9	11.4	15.1	18.8
	CPR + current	**72.9**	**84.2**	**87.9**	**90.6**
A::B Rewriting	Base	0.0	0.0	0.0	0.0
	CPR + current	**36.0**	**55.1**	**62.7**	**68.8**
Macro-average	Base	17.4	25.0	27.6	29.4
	CPR + current	**55.6**	**61.3**	**63.3**	**64.8**

CPR raises macro pass@1 from 17.4% to 55.6% and macro pass@16 from 29.4% to 64.8%. Its gains remain large throughout the sampling curve: 38.2, 36.3, 35.7, and 35.5 points at $K \in \{1, 4, 8, 16\}$. At $K = 16$, CPR improves nine of the ten tasks, including jugs from 0.0% to 53.1% and A::B rewriting from 0.0% to 68.8%.

D.7 Trained Small Models versus Larger Zero-Shot Models

How does training a small model compare with relying on a larger model's zero-shot performance? We compare the trained Qwen3-4B and Qwen2.5-VL-7B policies with Claude Opus 4.8, GPT-5.5, and larger same-family Qwen backbones. The Qwen references are Qwen3-32B for text reasoning and Qwen2.5-VL-32B-Instruct and Qwen2.5-VL-72B-Instruct for visual reasoning. Every model receives the same prompt, verifier, fixed evaluation examples, and 1024-token response budget.

Table 20: **CPR outperforms larger same-family zero-shot models across all five settings.** Scores are mean verifier accuracy on CRG's fixed evaluation sets with a 1024-token response budget. Claude Opus 4.8, GPT-5.5, and the larger Qwen backbones are evaluated zero-shot. Base is the untrained small model. The Seq. RLVR, CPR, and MTRL columns report trained policies. The best score in each row is bold.

Setting	Opus 4.8	GPT-5.5	Qwen-32B	Qwen-72B	Base	Seq. RLVR	CPR	MTRL
LLM algorithmic	53.7	**77.7**	29.3	–	17.5	49.9	63.3	65.3
LLM algebra	84.0	**94.8**	66.6	–	43.2	88.2	90.4	90.0
VLM quantitative	10.5	5.5	27.7	26.6	24.8	25.5	**37.2**	30.9
VLM spatial	19.9	0.8	27.7	28.9	24.1	26.3	**32.5**	31.6
VLM positional	19.1	8.6	27.7	25.8	28.7	34.7	33.2	**35.3**

CPR exceeds the larger same-family Qwen references in every setting. The trained 4B text policy outperforms Qwen3-32B, and the trained 7B visual policy outperforms both Qwen2.5-VL-32B-Instruct and Qwen2.5-VL-72B-Instruct. The frontier models show a modality-dependent pattern. GPT-5.5 remains strongest in both text settings, while CPR exceeds Claude Opus 4.8 and GPT-5.5 in all three visual settings.

E Extended Related Work

RLVR task suites. RLVR task suites supply verifier-scored reasoning problems. Reasoning Gym (Stojanovski et al., 2025) supplies 100+ procedural generators with algorithmic verifiers and parametric difficulty. VisuLogic (Xu et al., 2026) is a human-verified visual-reasoning benchmark with a rule-based RL baseline. AgentGym-RL (Xi et al., 2026) trains multi-turn LLM agents from environment outcome rewards. Verifiable rewards can also incentivize correct reasoning in base LLMs (Wen et al., 2026). These works establish task-generation and reward interfaces for RLVR. Their training protocols expose a fixed task inventory, with some curricula varying difficulty or interaction horizon. CRG orders task identities into stages and evaluates one policy after each arrival.

Forgetting in CL. Catastrophic forgetting describes the loss of earlier-task performance after training on later tasks and remains a central failure mode in CL (Kirkpatrick et al., 2017). Continual World measures this behavior in reinforcement-learning task sequences (Wolczyk et al., 2021), while TRACE treats earlier-task performance as a primary endpoint for sequentially fine-tuned language models (Wang et al., 2023). Reinforcement and supervised updates exhibit different forgetting behavior. RL's Razor finds that online reinforcement learning forgets less than supervised fine-tuning when the two are matched on new-task performance (Shenfeld et al., 2026). Zhang et al. report that reinforcement fine-tuning causes less forgetting of prior knowledge than SFT in multimodal language models (Zhang et al., 2026). We take these results as evidence that reinforcement-learning updates may cause less forgetting of earlier capabilities than supervised updates. Low forgetting alone does not determine whether sequential training reaches joint training on the final task set. Final performance also depends on how well arriving tasks are learned and how training transfers across tasks. This distinction motivates our comparison between continual RLVR and MTRL.

CL methods. These methods alter how a learner updates across task stages. Their mechanisms include regularization, plasticity restoration, parameter isolation, replay, and optimization. EWC (Kirkpatrick et al., 2017) regularizes parameters by Fisher information, ReDo (Sokar et al., 2023) reactivates dormant units, FIRE (Han et al., 2026) controls re-initialization geometry, SPHERE (Luo et al., 2026) targets spectral plasticity, OSFT (Nayak et al., 2026) constrains updates to a low-rank subspace, and Muon (Liu et al., 2025) changes the optimizer geometry. Their original evaluations span supervised CL, deep RL, and large-model optimization. Our comparison places representatives of these mechanisms under one continual-RLVR protocol and measures whether they narrow the endpoint gap to MTRL.