PUSH: Sparse Representation Dynamics Enable Cost-Efficient Reinforcement Learning in LLMs

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning (RL) for large language models relies on costly on-policy sampling. Exploiting the representational trajectory traversed during RL is a promising direction for improving training efficiency and model performance. We use sparse autoencoders (SAEs) to decompose an LLM's residual-stream activations into high-dimensional feature coordinates to interpret the model state during the RL process. The token-aligned change in SAE feature activations from the pre-RL model to the current model forms the ΔSAE signal. We find that the direction along which the model improves during RL is low-dimensional in ΔSAE coordinates, with five features at the final transformer layer accounting for 96.83% of the aggregate squared norm in code generation. These directions persist throughout training so that the signal predicts where RL is heading despite non-monotonic training dynamics. We introduce Progress Utilization via Sparse Hidden Representations (PUSH). On a fixed set of guidance texts, PUSH resolves this signal into its dynamic features and extends their per-token targets along the directions these features have established over training. Each PUSH application fits these targets in a brief activation-matching update outside the RL reward channel and standard RL then resumes under its original objective. Experiments in code generation and mathematical reasoning show immediate performance gains from PUSH without additional RL rollouts, and these gains persist during continued RL. Applying PUSH repeatedly reduces the cumulative sampling cost of RL by 30% on average at matched accuracy. Our results show that PUSH uses interpretability to explain prior learning and direct subsequent training.

1 Introduction

Reinforcement learning (RL) with verifiable rewards has become a central tool for improving the reasoning abilities of large language models (LLMs) (Shao et al., 2024; Guo et al., 2025; Yu et al., 2026). Since each policy update must be estimated from responses freshly sampled and verified under the current policy, on-policy rollout generation constitutes the principal computational cost of RL training (Fu et al., 2026). The sample efficiency of RL further deteriorates as training progresses, since performance saturates with increasing compute (Khatri et al., 2026; Qi et al., 2025; Liu et al., 2026) and an increasing proportion of rollouts contributes negligibly to policy updates (Yu et al., 2026; Zheng et al., 2025b). Standard RL discards each batch of costly rollouts immediately after its on-policy updates and leaves the substantial computation invested in sampling largely underutilized. The parameter change accumulated through these rollouts nevertheless exhibits a highly structured and reproducible pattern (Mukherjee et al., 2026; Zhu et al., 2025). Such regularity suggests that the change already obtained from past rollouts can be reused to amortize past exploration and advance training without additional sampling. Weight-space extrapolation pursues this possibility by scaling the parameter difference between checkpoints (Zheng et al., 2025a) but operates on entire checkpoints and therefore cannot identify which representational components carry the progress or in which token contexts they should be extended.

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.

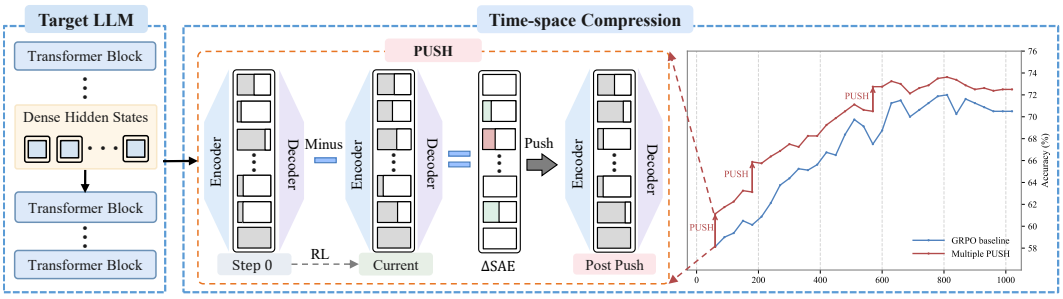


Figure 1: Conceptual illustration of PUSH. A frozen SAE maps the hidden states of the pre-RL and current models to shared sparse feature coordinates in which their token-aligned difference ΔSAE records the progress of RL.

Sparse autoencoders (SAEs) have emerged as a central technique in mechanistic interpretability, decomposing the hidden states of language models into sparsely active and largely monosemantic features (Huben et al., 2024; Bricken et al., 2023; Templeton et al., 2026; Gao et al., 2025). By resolving the superposition of more features than dimensions (Elhage et al., 2022), this decomposition localizes representational change to semantically meaningful coordinates. Because such dictionaries largely transfer from base models to their fine-tuned counterparts (Kissane et al., 2024), they provide a promising lens on training dynamics and have been used to characterize how fine-tuning alters a model (Lindsey et al., 2024; Minder et al., 2026a) and how features evolve over training (Xu et al., 2024; Bayazit et al., 2026; Cheng et al., 2026). These analyses nevertheless remain largely retrospective and seldom feed their interpretation back into optimization. We therefore interpret RL progress in fixed SAE coordinates and use this interpretation to direct subsequent training.

We define ΔSAE as the token-aligned change in frozen SAE feature activations from the pre-RL model to the current model and use it to interpret how RL reshapes model representations. This interpretation reveals across random seeds and RL algorithms that RL progress concentrates in a low-dimensional feature subspace whose early directions largely persist throughout training. These properties motivate Progress Utilization via Sparse Hidden Representations (PUSH), a brief intervention between RL phases that extends the historical change within this subspace. On guidance texts, PUSH compares current activations with cached pre-RL activations at matching token positions and fits the resulting token-specific feature targets through temporary low-rank adapters (Hu et al., 2021) before RL resumes under its original reward and objective. The procedure requires neither correctness labels nor additional rollouts and adds negligible cost to RL training.

Experiments on code generation and mathematical reasoning show that PUSH brings immediate gains that persist after RL resumes and consistently exceed those of supervised finetuning, representation-space, and parameter-extrapolation baselines. These gains hold across RL algorithms, base models, target layers, and guidance sets. Further analyses support this design by showing that PUSH moves the model toward representations that baseline RL reaches only later in training and that the gains drop substantially when the token-level correspondence of the historical change is disrupted. The same analyses reveal that representational progress along the historical direction persists after capability gains diminish as RL saturates. This decoupling suggests that adapting the timing and strength of PUSH to the training state is a promising direction for improving its efficiency.

Contributions.

- Interpretation of RL training. We introduce ΔSAE to interpret RL training in sparse feature coordinates and reveal that RL progress follows low-dimensional persistent directions and decouples from performance as RL saturates.
- ΔSAE-guided RL optimization. We propose PUSH, an adaptable method that fits token-level feature targets from ΔSAE through a brief update outside the reward channel and reduces cumulative rollouts at matched performance by 30% on average.
- Interpretability as a bridge from analysis to training. Consistent gains across diverse training settings indicate that interpretability can compress training in both space and time by identifying effective feature changes to boost subsequent training.

2 Related Work

Learning signals in LLM RL. Reward shaping and process supervision modify or enrich the feedback used by LLM RL (Yu et al., 2026; Hou et al., 2025); ReMax reduces optimization overhead by dispensing with a learned value model (Li et al., 2024). Reward-based guidance can introduce noise or suffer from proxy overoptimization when its signal is misaligned (Yu et al., 2026; Gao et al., 2023). PUSH draws an additional learning direction from the model's own representation history in a brief parameter update, after which RL resumes with its original reward and objective.

SAE-based representation control and training. SAEs provide fine-grained feature coordinates for representation analysis and control (Huben et al., 2024; Gao et al., 2025). Representation steering changes hidden states during inference (Singh et al., 2024); SAE-based steering and dynamic guardrails also apply their controls at inference time (Fang et al., 2026; Muhamed et al., 2025). SAE-LoRA and SASFT show that SAE features can guide parameter training, respectively to accommodate an inserted SAE and to maintain selected task features (Chen et al., 2025; Deng et al., 2026).

Parameter extrapolation. Task arithmetic edits models by combining parameter-difference vectors (Ilharco et al., 2023), while ExPO scales the difference between initial and partially aligned checkpoints with one coefficient (Zheng et al., 2025a). These weight-level operations apply their chosen coefficients across the entire difference, without selecting feature changes for individual token contexts. PUSH selects historical SAE feature changes at each token and fits a new parameter update around those targets; direct comparisons appear in Section 5.2.

3 Sparse Representation Dynamics during RL

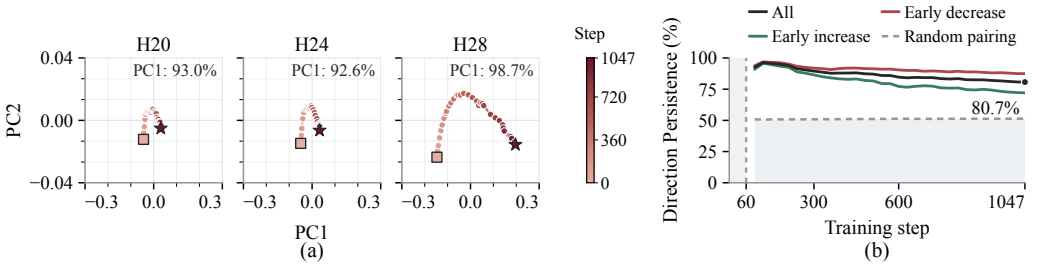


Figure 2: Persistent progress of RL in sparse feature coordinates. (a) Position-averaged SAE activations at H20, H24, and H28 across RL checkpoints projected onto their two leading principal components and colored by training step. (b) Fraction of features selected within the first 60 steps whose later changes continue their early increase or decrease compared with random pairing.

Sparse feature coordinates. We interpret RL progress in the feature coordinates of a TopK SAE fitted to the residual stream of the Transformer blocks of the LLM before training (Huben et al., 2024; Gao et al., 2025). For the hidden state $h_{k,t}$ at the ℓ-th layer Hℓ and token position t of checkpoint k, the SAE produces pre-TopK activations $r_{k,t}$ and sparse feature activations $z_{k,t}$

$$r_{k,t} = \mathrm{ReLU}\big(W_{\mathrm{enc}}^{\top}(h_{k,t}/\nu - b_{\mathrm{dec}}) + b_{\mathrm{enc}}\big), \qquad z_{k,t} = \mathrm{TopK}_K(r_{k,t}), \tag{1}$$

where ν normalizes the hidden-state scale and the nonzero entries of $z_{k,t}$ form the active support at position t. The SAE remains frozen throughout RL so that each coordinate denotes the same feature at every checkpoint, and base-model dictionaries largely transfer to fine-tuned models (Kissane et al., 2024) in agreement with the reconstruction quality of our SAEs after RL (Appendix A.1). Taking $k = 0$ as the model before training, we define the token-aligned historical feature change ΔSAE and its average over positions $\mathcal{T}$ as

$$\Delta z_{k,t} = z_{k,t} - z_{0,t}, \qquad \overline{\Delta z}_k = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \Delta z_{k,t}. \tag{2}$$

Positive and negative entries of $\Delta z_{k,t}$ record features that RL respectively strengthens and weakens at position t, and $\overline{\Delta z}_k$ traces the overall movement of the representation along the features that carry it.

Geometry of RL progress. Decomposed in these feature coordinates, the change learned during RL concentrates in a low-dimensional subspace spanned by a small number of interpretable features that consistently account for most of $\|\overline{\Delta z}_k\|_2^2$ (Section 6.1). This low-dimensional structure parallels the sparse localization of RL updates in parameter space (Mukherjee et al., 2026; Zhu et al., 2025) and agrees with evidence that post-training mainly reinforces mechanisms already present in the base model (Prakash et al., 2024; Ward et al., 2025). Beyond being low-dimensional, this change accumulates along a persistent direction. The position-averaged trajectories advance along a sustained principal direction at every examined layer with only local fluctuations (Figure 2a), and disjoint problem sets recover nearly the same direction (Appendix A.4). This direction is identifiable from early training history since 80.7% of the features whose average activations rise or fall within the first 60 steps still move in their early direction at the final checkpoint compared with 51.4% under random pairing (Figure 2b and Appendix A.5.4). RL progress therefore follows a low-dimensional and temporally consistent trajectory in feature space, and the ΔSAE accumulated by an intermediate checkpoint serves as an informative predictor of the subsequent update direction and a natural basis for extrapolation.

4 Method

PUSH turns the extrapolation suggested by Section 3 into a brief parameter update inserted between RL phases. It constructs token-aligned historical targets on a fixed set of guidance texts, restricts each target to the features with the largest historical change, and fits suppression and continuation targets in separate branches whose updates are merged before RL resumes.

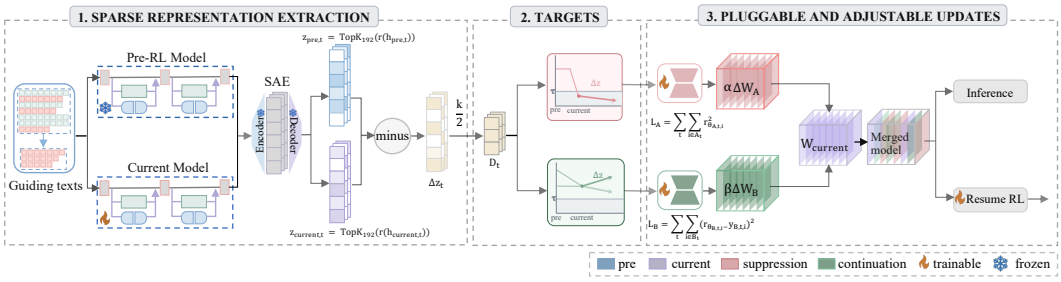


Figure 3: Overview of the PUSH method. The procedure extracts sparse representations from the pre-RL and current models, constructs token-level targets, and applies separate trainable updates before RL resumes.

4.1 Sparse Representation Extraction

PUSH operates on a small random sample $\mathcal{G}$ of guidance texts with answer-token positions $\mathcal{T}(\mathcal{G})$. The sparse feature activations $z_{\mathrm{pre},t}$ of the model before training are cached once on $\mathcal{G}$. Each application of PUSH encodes the current model on the same texts and obtains $r_{\mathrm{now},t}$ and $z_{\mathrm{now},t}$ from Equation (1) together with the token-level ΔSAE $\Delta z_t = z_{\mathrm{now},t} - z_{\mathrm{pre},t}$ from Equation (2) entirely offline. PUSH retains ΔSAE at the token level because RL-induced change depends on context (Wang et al., 2026a; Akgül et al., 2026) and averaging over positions would discard this structure. Disrupting this token-level correspondence substantially reduces the gains of PUSH (Section 6.3).

4.2 Constructing Token-Level Targets

Since the average change in Equation (2) is the mean of the token-level changes, targets that extend each token along its own ΔSAE jointly extend this average along the low-

dimensional and persistent direction identified in Section 3, subject to the feature selection and nonnegativity constraints described below. At each token, PUSH selects at most $D = K/2$ features with the largest nonzero absolute historical changes and thereby retains the largest share of squared historical change under this budget (Appendix B.1). Among the selected features, the suppression set A_t contains those present in the pre-RL TopK support but absent from the current support, and the continuation set B_t contains features that either enter or remain in the current support. The corresponding targets are

$$y_{A,t,i} = 0 \ (i \in A_t), \qquad y_{B,t,i} = [z_{\text{now},t,i} + \gamma(z_{\text{now},t,i} - z_{\text{pre},t,i})]_+ \ (i \in B_t), \qquad \gamma \geq 0, \quad (3)$$

For A_t, the pre-TopK activation $r_{\text{now},t,i}$ can remain positive after the feature leaves the TopK support, so the zero target suppresses this residual response. For B_t, the target continues the historical increase or decrease from the current activation and clips negative values to zero. The selected sets and their targets remain fixed throughout a PUSH.

4.3 Pluggable and Adjustable Updates

PUSH pauses RL and fits the suppression (A) and continuation (B) targets in separate temporary low-rank adaptation (LoRA) branches initialized from the current model W_{now} (Hu et al., 2021). Only the branch LoRA parameters are optimized, with gradients propagated through the frozen SAE. On a guidance batch $G_b \subseteq \mathcal{G}$, the two branches minimize

$$\mathcal{L}_A(\theta; G_b) = \sum_{t \in \mathcal{T}(G_b)} \|r_{\theta,t}[A_t]\|_2^2, \qquad \mathcal{L}_B(\theta; G_b) = \sum_{t \in \mathcal{T}(G_b)} \|r_{\theta,t}[B_t] - y_{B,t}\|_2^2, \quad (4)$$

where $r_{\theta,t}$ is the pre-TopK activation obtained by applying Equation (1) under trainable parameters θ, $v[S]$ denotes the subvector of v indexed by S, and $y_{B,t} = (y_{B,t,i})_{i \in B_t}$ (Appendix B.2). After fitting, PUSH merges the branch changes $\Delta W_A = W_A - W_{\text{now}}$ and $\Delta W_B = W_B - W_{\text{now}}$ as

$$\Delta W_{\text{PUSH}} = s\,(\alpha \Delta W_A + \beta \Delta W_B), \qquad W_{\text{PUSH}} = W_{\text{now}} + \Delta W_{\text{PUSH}}, \quad s \geq 0. \quad (5)$$

PUSH uses the fixed ratio $(\alpha, \beta) = (2, 1)$, and the overall strength s can be rescaled from the saved update without another training run. RL then resumes from W_{PUSH} with its saved training state and original objective and reward, and inference requires no SAE (Appendices B and C.1). Multiple PUSH repeats this procedure at later checkpoints with the pre-RL model as the fixed historical reference.

5 Experiments

5.1 Experimental Setup

We evaluate immediate gains and continued RL in code generation with Qwen2.5-Coder-7B-Instruct (Hui et al., 2024) under GRPO (Shao et al., 2024) and DAPO (Yu et al., 2026) on AceCode (Zeng et al., 2025), and in mathematical reasoning with Qwen2.5-Math-7B Base (Yang et al., 2024) and DeepSeekMath-Instruct 7B (Shao et al., 2024) under GRPO on GSM8K (Cobbe et al., 2021). The coding runs use the public verl training framework with its vLLM rollout backend; the exact training and evaluation settings are listed in Appendix C. We report greedy pass@1 accuracy on held-out problems and compare each application of PUSH with the baseline checkpoint at the same training step using $s = 1$ (Appendix E.2). Configuration analyses use a code validation split disjoint from the test split, and further details appear in Appendices C and B.

5.2 Immediate Capability Gains

We evaluate a single PUSH before RL resumes at eight checkpoints per algorithm across three training seeds against four alternatives that reuse the same guidance texts or the same training history (Appendix D.2.1). SFT fits the guidance responses as in self-training and supervised updates interleaved with RL (Zelikman et al., 2022; Ma et al., 2026), while ADL extrapolates dense hidden states along their historical change following the activation

difference lens (Minder et al., 2026b) and activation-difference steering (Rimsky et al., 2024). RepE restricts this extrapolation to the leading principal subspace of the change (Zou et al., 2023), and ExPO extends the weight difference from the model before training without fitting (Zheng et al., 2025a; Wang et al., 2026b).

Table 1: Immediate accuracy gains in percentage points over the baseline checkpoint at the same step, reported as means over three training seeds with standard deviations as subscripts. Δ denotes the margin of PUSH over the strongest alternative.

	Step	SFT	ADL	RepE	ExPO	PUSH$_{(ours)}$	Δ
GRPO	60	$0.58_{\pm 0.36}$	$0.33_{\pm 0.44}$	$\underline{0.71}_{\pm 0.19}$	$0.38_{\pm 0.00}$	$\mathbf{2.50}_{\pm 0.45}$	+1.79
	150	$0.88_{\pm 0.25}$	$0.46_{\pm 0.26}$	$0.21_{\pm 0.31}$	$\underline{0.92}_{\pm 0.19}$	$\mathbf{2.96}_{\pm 0.07}$	+2.04
	240	$1.50_{\pm 0.25}$	$1.29_{\pm 0.19}$	$\underline{1.79}_{\pm 0.26}$	$1.71_{\pm 0.14}$	$\mathbf{4.13}_{\pm 0.82}$	+2.33
	330	$-0.21_{\pm 0.31}$	$0.29_{\pm 0.40}$	$0.33_{\pm 0.26}$	$\underline{2.08}_{\pm 0.52}$	$\mathbf{3.38}_{\pm 0.22}$	+1.29
	420	$-0.71_{\pm 0.38}$	$\underline{2.17}_{\pm 0.38}$	$2.08_{\pm 0.38}$	$1.17_{\pm 0.52}$	$\mathbf{3.71}_{\pm 0.14}$	+1.54
	510	$-2.17_{\pm 0.63}$	$-0.38_{\pm 0.13}$	$\underline{-0.08}_{\pm 0.19}$	$-1.83_{\pm 0.19}$	$\mathbf{3.50}_{\pm 0.25}$	+3.58
	600	$-1.38_{\pm 0.25}$	$1.79_{\pm 0.47}$	$\underline{2.13}_{\pm 0.13}$	$-1.29_{\pm 0.40}$	$\mathbf{3.33}_{\pm 0.56}$	+1.21
	690	$-1.71_{\pm 0.26}$	$\underline{1.04}_{\pm 0.19}$	$-1.54_{\pm 0.26}$	$-1.29_{\pm 0.31}$	$\mathbf{1.46}_{\pm 0.26}$	+0.42
DAPO	60	$\underline{1.04}_{\pm 0.31}$	$-0.54_{\pm 0.51}$	$0.33_{\pm 0.19}$	$0.92_{\pm 0.38}$	$\mathbf{3.58}_{\pm 0.63}$	+2.54
	150	$\underline{2.17}_{\pm 0.19}$	$1.54_{\pm 0.19}$	$1.29_{\pm 0.44}$	$1.46_{\pm 0.47}$	$\mathbf{3.96}_{\pm 0.59}$	+1.79
	240	$2.29_{\pm 0.19}$	$1.63_{\pm 0.13}$	$\underline{2.46}_{\pm 0.56}$	$2.21_{\pm 0.26}$	$\mathbf{5.33}_{\pm 0.07}$	+2.88
	330	$0.38_{\pm 0.13}$	$2.04_{\pm 0.26}$	$1.50_{\pm 0.13}$	$\mathbf{5.42}_{\pm 0.26}$	$\underline{2.75}_{\pm 0.22}$	−2.67
	420	$-0.13_{\pm 0.25}$	$\underline{1.17}_{\pm 0.51}$	$0.54_{\pm 0.26}$	$0.29_{\pm 0.31}$	$\mathbf{3.08}_{\pm 0.44}$	+1.92
	510	$0.04_{\pm 0.14}$	$-0.08_{\pm 0.36}$	$0.29_{\pm 0.36}$	$\underline{1.04}_{\pm 0.14}$	$\mathbf{2.29}_{\pm 0.56}$	+1.25
	600	$-1.67_{\pm 0.26}$	$\underline{1.04}_{\pm 0.26}$	$-0.17_{\pm 0.44}$	$-3.29_{\pm 0.44}$	$\mathbf{1.63}_{\pm 0.57}$	+0.58
	690	$-1.04_{\pm 0.38}$	$\underline{-0.79}_{\pm 0.26}$	$-1.17_{\pm 0.26}$	$-7.17_{\pm 0.26}$	$\mathbf{0.58}_{\pm 0.62}$	+1.38

PUSH attains the largest immediate improvement at all but one starting point and remains beneficial throughout early and middle training under both RL algorithms (Table 1). The guidance data alone cannot account for these gains because supervised fitting of the same responses helps only in early training and turns detrimental as RL progresses. ADL and RepE extract smaller and less consistent gains from the same history, indicating that this history becomes substantially more useful once resolved into sparse feature coordinates. ExPO remains competitive in early training but deteriorates sharply as training proceeds because scaling the entire weight difference likely amplifies changes that no longer drive progress, and a displacement-matched variant excludes update magnitude as the source of the advantage of PUSH (Appendix D.2.2). Together with its consistent gains in mathematical reasoning (Appendix D.1), these comparisons attribute the immediate improvement of PUSH to the selective extrapolation of token-level history in sparse feature coordinates.

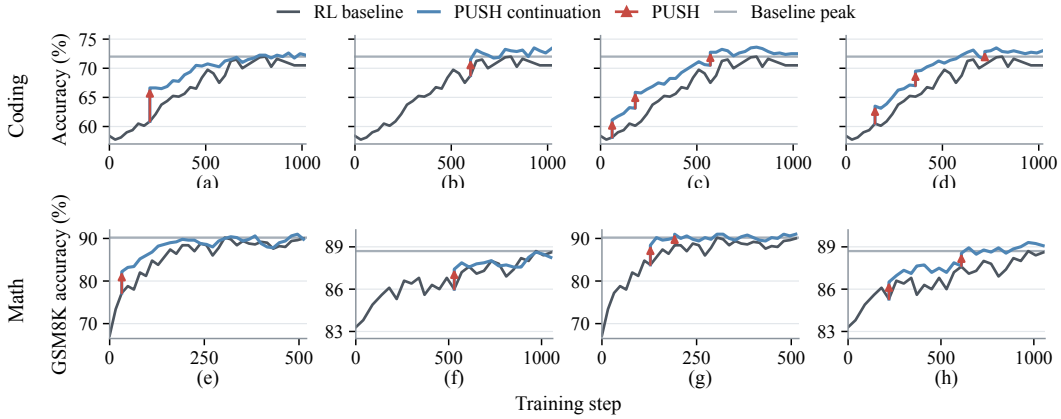


Figure 4: Persistence of PUSH gains under continued GRPO. Panels (a)–(d) follow one training run of Qwen2.5-Coder-7B-Instruct on code generation and panels (e)–(h) follow Qwen2.5-Math-7B Base in (e) and (g) and DeepSeekMath-Instruct 7B in (f) and (h) on mathematical reasoning, with Single PUSH in the first two panels and Multiple PUSH in the last two panels of each row.

5.3 Continued RL with Single and Multiple PUSH

Figure 4 compares continued training after Single and Multiple PUSH with the RL baseline at matched training steps under the schedules in Appendix D.3, and the gains of PUSH persist after RL resumes in both domains. In code generation every illustrated branch remains above the baseline throughout the hundreds of updates after the last intervention. In mathematical reasoning the advantage of Single PUSH fades as the baseline plateaus, whereas Multiple PUSH keeps both models above the baseline until training ends. Continued RL therefore retains the capability gained through PUSH, and repeated application sustains this advantage where training saturates early.

5.4 Rollout Savings and Computational Cost

Table 2: Rollout savings of Single and Multiple PUSH at matched test accuracy reported as means over three training seeds with standard deviations.

	Rollouts saved (k)		Post-PUSH reduction (%)		Total reduction (%)	
	Single	Multiple	Single	Multiple	Single	Multiple
GRPO	$31.4_{\pm 6.7}$	$42.9_{\pm 9.2}$	$42.57_{\pm 9.27}$	$51.86_{\pm 8.56}$	$22.29_{\pm 4.88}$	$29.09_{\pm 5.67}$
DAPO	$78.4_{\pm 25.2}$	$140.8_{\pm 10.9}$	$33.74_{\pm 9.75}$	$52.94_{\pm 4.65}$	$18.97_{\pm 5.79}$	$31.61_{\pm 2.82}$

Rollout savings at matched accuracy. A complete PUSH workflow lasts approximately one minute on a single NVIDIA H200. The mean end-to-end time over three runs ranges from 52.67 seconds for Qwen2.5-Coder-7B-Instruct to 77.07 seconds for Qwen2.5-Math-7B Base (Table 23, Appendix D.4.3). Brief PUSH updates deliver substantial savings in subsequent on-policy sampling. We report both post-PUSH reduction, measured relative to the baseline rollouts needed after the first intervention, and total reduction, measured from the beginning of RL. For each training seed, we average savings over shared accuracy thresholds and six starting points; Table 2 reports the mean and sample standard deviation across three seeds per algorithm. The full calculation and per-seed results appear in Appendix D.4.1.

Multiple PUSH cuts the remaining rollout budget by 51.86–52.94%, saving approximately half of the sampling required after the first intervention. These gains translate into total rollout reductions of 29.09% and 31.61% respectively. The improvement can bring high-performing models forward by hundreds of training steps: a GRPO Multiple PUSH branch starting at step 240 reaches 73.25% accuracy at step 480, whereas its baseline first reaches the same accuracy at step 990, an advance of 510 steps and 130,560 rollouts. By turning accumulated training progress into a brief parameter update, PUSH reduces the demand for costly on-policy sampling and advances capability with little intervention time.

6 Discussion

6.1 Concentration of Representation Changes

The low-dimensional trajectory of RL progress in Section 3 admits a sparse description in feature space, since a compact set of features shared across training runs and algorithms carries most of the historical feature change. Across three seeds of both GRPO and DAPO the same five H28 features rank highest in the same order, and on problems held out from feature selection these five of the 28,672 coordinates account for 93.43–96.83% of $\|\overline{\Delta z}_k\|_2^2$ at the end of training (Appendix A.3). Although these features approach their final change at different rates during training (Figure 5), the change decoded from them alone closely tracks the full representation change throughout training (Appendix A.3.1). This sparsity therefore supports the extrapolation of RL progress in sparse feature coordinates.

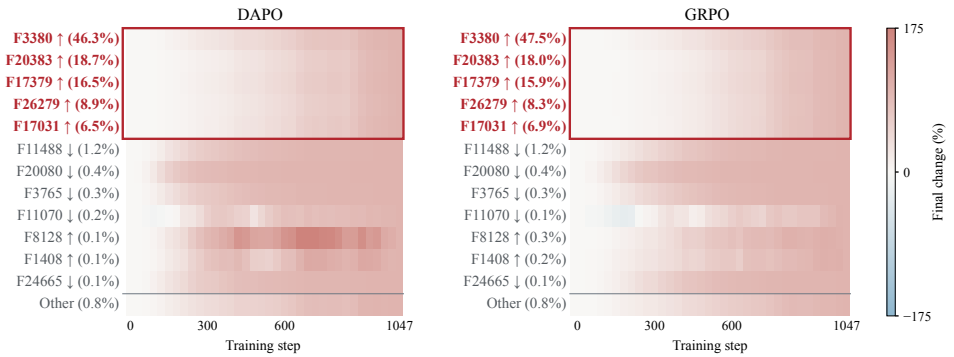


Figure 5: Five sparse features dominate the representation change during DAPO (left) and GRPO (right) training. Colors show the fraction of the final change that each feature has completed, and percentages give its share of the squared change at the end of training.

6.2 Representation Changes after PUSH

Table 3: Representation measurements and test-accuracy gains after PUSH in one training run of each algorithm. Representation entries report raw hidden states and SAE features in that order, and accuracy gains are in percentage points over the same-step baseline.

	GRPO				DAPO			
Step	Displacement	Nearest	Cosine	Gain	Displacement	Nearest	Cosine	Gain
60	.0558/.0531	120/120	.975/.974	+3.00	.0489/.0456	120/150	.981/.978	+3.50
150	.1553/.1466	270/270	.951/.917	+3.00	.1054/.0988	240/240	.943/.908	+4.63
210	.2092/.2080	330/360	.971/.981	+5.75	.1517/.1511	360/360	.971/.973	+4.63
240	.2417/.2491	390/420	.979/.986	+5.00	.1773/.1825	420/420	.971/.978	+5.38
270	.2789/.2961	480/480	.971/.980	+2.25	.1913/.2004	450/450	.947/.946	+4.00
330	.3218/.3481	600/600	.954/.964	+3.50	.2086/.2215	510/510	.977/.983	+2.88
420	.3462/.3802	690/690	.947/.958	+3.63	.2324/.2499	540/540	.980/.987	+2.63
510	.3770/.4144	720/720	.951/.962	+3.50	.2551/.2708	690/690	.969/.983	+2.25
600	.4186/.4669	930/930	.948/.966	+2.75	.2915/.3066	810/810	.972/.989	+2.13

PUSH moves representations toward states reached later by baseline RL. Table 3 compares the representation after PUSH with the baseline trajectory at nine starting points from steps 60 through 600 in one training run of each algorithm. For each starting point we report the normalized displacement induced by PUSH, the baseline checkpoint nearest to the post-PUSH representation, and the cosine between the PUSH displacement and the natural displacement toward that checkpoint. Across all point comparisons both representation views move toward a baseline checkpoint 60 to 330 steps ahead in a direction closely aligned with natural training, and accuracy improves at every starting point (Appendix E.1).

6.3 Ablation Studies

We organize the comparisons into three groups: Historical targets, Feature selection, and Branch combination. The first group tests whether each token keeps its own historical information; the second tests which changed features are retained and how many; the third tests how the suppression and continuation branches are combined. Table 4 follows this hierarchy at seven starting points, using the fixed test-set strength $s = 1$. Every entry summarizes all three seeds. Replacing the guidance texts with either of two disjoint sets leaves the gains nearly unchanged, and the remaining sensitivity analyses appear in Appendix E.

Every ablated variant falls below Full PUSH at all starting points, and the size of the loss distinguishes the components that determine the gain of PUSH from those that refine it. Shuffling pre-RL features across tokens or replacing token-level directions with their mean removes most of the gain, and selecting the same number of features at random eliminates it entirely. Retaining all changed features or removing either branch causes smaller losses, and doubling the budget to 192 features leaves the gain nearly unchanged (Appendix E.3). PUSH therefore derives its gain mainly from extrapolating the token-aligned history of the

most changed features, while the feature budget and the complementary suppression and continuation branches refine this gain.

Table 4: Ablations of PUSH with accuracy gains in percentage points over the same-step baseline. The variants shuffle pre-RL features across tokens (w/o align.), replace token-level directions with their mean (w/o token), select D features at random (w/o Top-D), retain all changed features (w/o budget), and remove the continuation or suppression branch (w/o cont., w/o supp.).

		Historical targets		Feature selection		Branch combination	
Step	Full PUSH	w/o align.	w/o token	w/o Top-D	w/o budget	w/o cont.	w/o supp.
150	$2.96_{\pm 0.07}$	$0.50_{\pm 0.13}$	$0.83_{\pm 0.07}$	$-0.25_{\pm 0.13}$	$2.67_{\pm 0.26}$	$2.29_{\pm 0.14}$	$1.75_{\pm 0.22}$
210	$5.96_{\pm 0.19}$	$1.25_{\pm 0.13}$	$1.63_{\pm 0.13}$	$0.25_{\pm 0.13}$	$5.83_{\pm 0.29}$	$4.33_{\pm 0.14}$	$3.63_{\pm 0.13}$
240	$4.13_{\pm 0.82}$	$-0.13_{\pm 0.13}$	$1.08_{\pm 0.19}$	$0.00_{\pm 0.13}$	$2.67_{\pm 0.19}$	$3.58_{\pm 0.40}$	$2.79_{\pm 0.26}$
420	$3.71_{\pm 0.14}$	$1.67_{\pm 0.07}$	$2.13_{\pm 0.13}$	$-0.50_{\pm 0.13}$	$1.63_{\pm 0.13}$	$1.88_{\pm 0.13}$	$2.25_{\pm 0.13}$
510	$3.50_{\pm 0.25}$	$-0.75_{\pm 0.13}$	$0.17_{\pm 0.07}$	$0.29_{\pm 0.07}$	$1.17_{\pm 0.07}$	$2.63_{\pm 0.13}$	$1.75_{\pm 0.13}$
570	$2.96_{\pm 0.14}$	$0.42_{\pm 0.07}$	$0.50_{\pm 0.13}$	$-0.08_{\pm 0.07}$	$1.75_{\pm 0.13}$	$1.38_{\pm 0.13}$	$0.92_{\pm 0.07}$
600	$3.33_{\pm 0.56}$	$0.38_{\pm 0.13}$	$0.75_{\pm 0.13}$	$0.13_{\pm 0.13}$	$2.92_{\pm 0.07}$	$3.13_{\pm 0.13}$	$1.79_{\pm 0.19}$

6.4 Training Stage and PUSH Strength

Figure 6 compares update strengths and suppression-to-continuation ratios across GRPO training stages on the validation split. Across the seven early- and middle-stage checkpoints from steps 60 to 420, all 70 evaluated coefficient settings yield positive validation gains over the same-step baseline. At step 210, doubling the update strength increases the gain from 3.13 to 7.38 percentage points. Related stage dependence also appears in model extrapolation (Zheng et al., 2025a). All test-set results use the fixed strength $s = 1$ without stage-specific tuning, and adapting the strength to the training stage remains a direction for further gains (Appendix E.2).

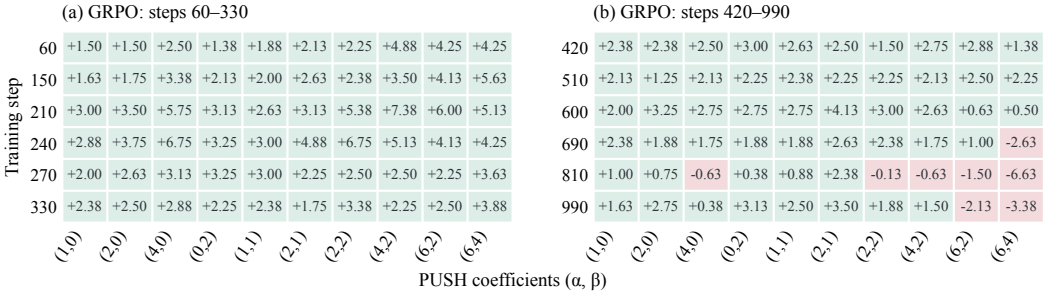


Figure 6: Validation accuracy gains of PUSH in percentage points over the same-step baseline under different coefficient pairs (α, β) across GRPO training stages.

7 Conclusion

We introduced PUSH, which selects dynamic features from the historical change at each token, constructs context-specific progression targets, and writes them into model parameters through brief LoRA updates. In the main code experiments, PUSH reaches matched accuracy with fewer cumulative rollouts during early and middle training, and part of the capability gain persists after RL resumes. Comparisons with direct parameter extrapolation show that selective progression is more effective than extrapolating the full update, while experiments in mathematical reasoning provide validation in a second task domain. These results support the view that RL learning history, once acquired through costly sampling, can provide additional training supervision and improve training efficiency through a brief auxiliary update.

AI use statement

In this work, we used generative AI tools to assist with manuscript language editing, the organization of text and figures, code documentation, formatting, literature organization, and checking reported numbers. The authors developed the research method, formulated the hypotheses and claims, designed the experiments, and wrote the experimental code; the data were generated and analyzed by the authors. We did not use generative AI tools to propose the research method, generate data, or replace the authors' judgment of the experimental results. All AI-assisted text and code were reviewed and tested by the authors, and all citations and reported numbers were checked against their original sources or experimental records. The authors take full responsibility for the final manuscript, code, and other research artifacts.

References

Ömer Faruk Akgül, Rajgopal Kannan, Willie Neiswanger, and Viktor Prasanna. Rethinking RL for LLM reasoning: It's sparse policy selection, not capability learning. arXiv preprint arXiv:2605.06241, 2026.

Deniz Bayazit, Aaron Mueller, and Antoine Bosselut. Crosscoding through time: Tracking emergence & consolidation of linguistic representations throughout LLM pretraining. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1353–1377, 2026.

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

Matthew Chen, Joshua Engels, and Max Tegmark. Low-rank adapting models for sparse autoencoders. arXiv preprint arXiv:2501.19406, 2025.

Yue Cheng, Jiajun Zhang, Xiaohui Gao, Weiwei Xing, Zheng Wang, and Zhanxing Zhu. Mechanistically interpreting the role of sample difficulty in RLVR for LLMs. arXiv preprint arXiv:2605.28388, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.

Boyi Deng, Yu Wan, Baosong Yang, Fei Huang, Wenjie Wang, and Fuli Feng. SASFT: Sparse autoencoder-guided supervised finetuning to mitigate unexpected code-switching in LLMs. In International Conference on Learning Representations, volume 2026, pp. 14084–14109, 2026.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. arXiv preprint arXiv:2209.10652, 2022.

Yi Fang, Wenjie Wang, Mingfeng Xue, Boyi Deng, Fengli Xu, Dayiheng Liu, and Fuli Feng. Controllable LLM reasoning via sparse autoencoder-based steering. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 21290–21305, 2026.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning. Advances in Neural Information Processing Systems, 38:36256–36282, 2026.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Proceedings of the 40th International Conference on Machine Learning, volume 202, pp. 10835–10866, 2023.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In International Conference on Learning Representations, volume 2025, pp. 26721–26754, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 12355–12369, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In International Conference on Learning Representations, volume 2024, pp. 7827–7845, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-Coder technical report. arXiv preprint arXiv:2409.12186, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In International Conference on Learning Representations, 2023.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Venkata Sai Surya Subramanyam Duvvuri, Manzil Zaheer, Inderjit Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for LLMs. In International Conference on Learning Representations, volume 2026, pp. 72438–72467, 2026.

Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. SAEs (usually) transfer between base and chat models. AI Alignment Forum, 2024. URL https://www.alignmentforum.org/posts/fmwk6qxrpW8d4jvbd/saes-usually-transfer-between-base-and-chat-models.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. ReMax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In Proceedings of the 41st International Conference on Machine Learning, volume 235, pp. 29128–29163, 2024.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. Transformer Circuits Thread, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. Advances in Neural Information Processing Systems, 38:17998–18031, 2026.

Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Wong, Junbo Niu, Chengyu Shen, Runming He, Yanhao Li, et al. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. In International Conference on Learning Representations, volume 2026, pp. 80802–80821, 2026.

Julian Minder, Clément Dumas, Caden Juang, Bilal Chughtai, and Neel Nanda. Overcoming sparsity artifacts in crosscoders to interpret chat-tuning. Advances in Neural Information Processing Systems, 38:106423–106474, 2026a.

Julian Minder, Clément Dumas, Stewart Slocum, Helena Casademunt, Cameron Holmes, Robert West, and Neel Nanda. Narrow finetuning leaves clearly readable traces in activation differences. In International Conference on Learning Representations, volume 2026, pp. 35340–35406, 2026b.

Aashiq Muhamed, Jacopo Bonato, Mona Diab, and Virginia Smith. SAEs can improve unlearning: Dynamic sparse autoencoder guardrails for precision unlearning in LLMs. In Conference on Language Modeling, 2025.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tur, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. Advances in Neural Information Processing Systems, 38:132119–132138, 2026.

Nikhil Prakash, Tamar Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In International Conference on Learning Representations, volume 2024, pp. 8057–8082, 2024.

Zhenting Qi, Fan Nie, Alexandre Alahi, James Zou, Himabindu Lakkaraju, Yilun Du, Eric Xing, Sham Kakade, and Hanlin Zhang. EvoLM: In search of lost language model training dynamics. arXiv preprint arXiv:2506.16029, 2025.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 15504–15522, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.

Shashwat Singh, Shauli Ravfogel, Jonathan Herzig, Roee Aharoni, Ryan Cotterell, and Ponnurangam Kumaraguru. Representation surgery: Theory and practice of affine steering. In Proceedings of the 41st International Conference on Machine Learning, 2024.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, et al. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. arXiv preprint arXiv:2605.29358, 2026.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. Advances in Neural Information Processing Systems, 38:115452–115486, 2026a.

Tianle Wang, Jiayu Liu, Zhongyuan Wu, Shenghao Jin, Wei Chen, Hao Xu, and Ning Miao. Linear dynamics in the RLVR training of large language models. arXiv preprint arXiv:2601.04537, 2026b.

Jake Ward, Chuqiao Lin, Constantin Venhoff, and Neel Nanda. Reasoning-finetuning repurposes latent representations in base models. arXiv preprint arXiv:2507.12638, 2025.

Yang Xu, Yi Wang, Hengguan Huang, and Hao Wang. Tracking the feature dynamics in LLM training: A mechanistic study. arXiv preprint arXiv:2412.17626, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving interference when merging models. In Advances in Neural Information Processing Systems, volume 36, 2023.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. arXiv preprint arXiv:2409.12122, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. Advances in Neural Information Processing Systems, 38:113222–113244, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. Advances in Neural Information Processing Systems, 35:15476–15488, 2022.

Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. AceCoder: Acing coder RL via automated test-case synthesis. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 12023–12040, 2025.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Model extrapolation expedites alignment. In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1025–1041, 2025a.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts, 2025b. URL https://arxiv.org/abs/2506.02177.

Hanqing Zhu, Zhenyu Zhang, Hanxian Huang, DiJia Su, Zechun Liu, Jiawei Zhao, Igor Fedorov, Hamed Pirsiavash, Zhizhou Sha, Jinwon Lee, et al. The path not taken: RLVR provably learns off the principals. arXiv preprint arXiv:2511.08567, 2025.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to AI transparency. arXiv preprint arXiv:2310.01405, 2023.

Appendix overview.

Appendix	Contents
A	Representation Dynamics: Properties motivating PUSH
B	PUSH Implementation: Target selection, frozen loss, optimization, and merge
C	Experimental Setup: Models, evaluation, SAE training, data, and configurations
D	Main Experimental Results: Immediate gains, external benchmarks, baseline comparisons, continued RL, rollout savings, and PUSH cost
E	Supplementary Discussion and Analyses: Joint representation and capability progress, training stage and PUSH strength, and additional ablations
F	Limitations and Future Work

A Representation Dynamics

This appendix examines SAE reconstruction, training directions across runs and problem sets, and the features carrying these changes. We first establish the quality of the recovered coordinates, then trace how they move during training, identify the features that carry the movement, and finally examine their directions and code effects.

A.1 Reconstruction

We evaluate the three layer-specific SAEs from Table 15 before RL and at step 1047 across three seeds of DAPO and GRPO, using the reconstruction data in Appendix C.3. Table 1 reports the shared pre-RL model and each algorithm's post-RL mean and sample standard deviation. Lower FVU and higher reconstruction cosine indicate better reconstruction.

Table 1: Reconstruction quality before and after RL on the same 450 problems. The first three columns in each block report the shared pre-RL value and the three-seed post-RL means ± sample standard deviations; the last two columns report the corresponding change from pre-RL, computed before rounding.

FVU ↓					
Layer	Pre-RL	DAPO	GRPO	DAPO increase	GRPO increase
H20	0.0207	$0.0240_{\pm 0.0013}$	$0.0245_{\pm 0.0006}$	0.0033	0.0038
H24	0.0622	$0.0755_{\pm 0.0032}$	$0.0738_{\pm 0.0020}$	0.0133	0.0116
H28	0.1391	$0.1632_{\pm 0.0077}$	$0.1612_{\pm 0.0066}$	0.0241	0.0221
Reconstruction cosine ↑					
Layer	Pre-RL	DAPO	GRPO	DAPO decrease	GRPO decrease
H20	0.9459	$0.9300_{\pm 0.0064}$	$0.9303_{\pm 0.0028}$	0.0159	0.0157
H24	0.9506	$0.9299_{\pm 0.0065}$	$0.9357_{\pm 0.0029}$	0.0207	0.0149
H28	0.9705	$0.9296_{\pm 0.0040}$	$0.9378_{\pm 0.0096}$	0.0409	0.0327

FVU increases and reconstruction cosine decreases after RL in every run. GRPO varies less across seeds at H20 and H24; at H28, DAPO has larger FVU variation but smaller cosine variation. The final two columns of each block make these changes directly comparable across algorithms.

A.2 Trajectories

All six runs use the trajectory data in Appendix C.3 and the three layer-specific SAEs in Table 15. At each saved checkpoint from pre-RL to step 1047, we average post-TopK activations within each answer, then equally across problems, excluding prompt and padding positions. PCA and t-SNE are fitted separately for each run and layer. The t-SNE input is the first 20 PCA dimensions, with perplexity 8 and seed 20260904; rotations and reflections align panel orientations. Main Figure 2 shows GRPO Seed 1, whose PC1 explains 93.0%, 92.6%, and 98.7% of checkpoint variance at H20/H24/H28. Figures 1 and 2 show all six runs. PCA panels share axis limits and ticks, whereas t-SNE axis limits are adjusted separately

for each panel to show its trajectory clearly; PCA axes and t-SNE distances are interpreted within each fitted panel.

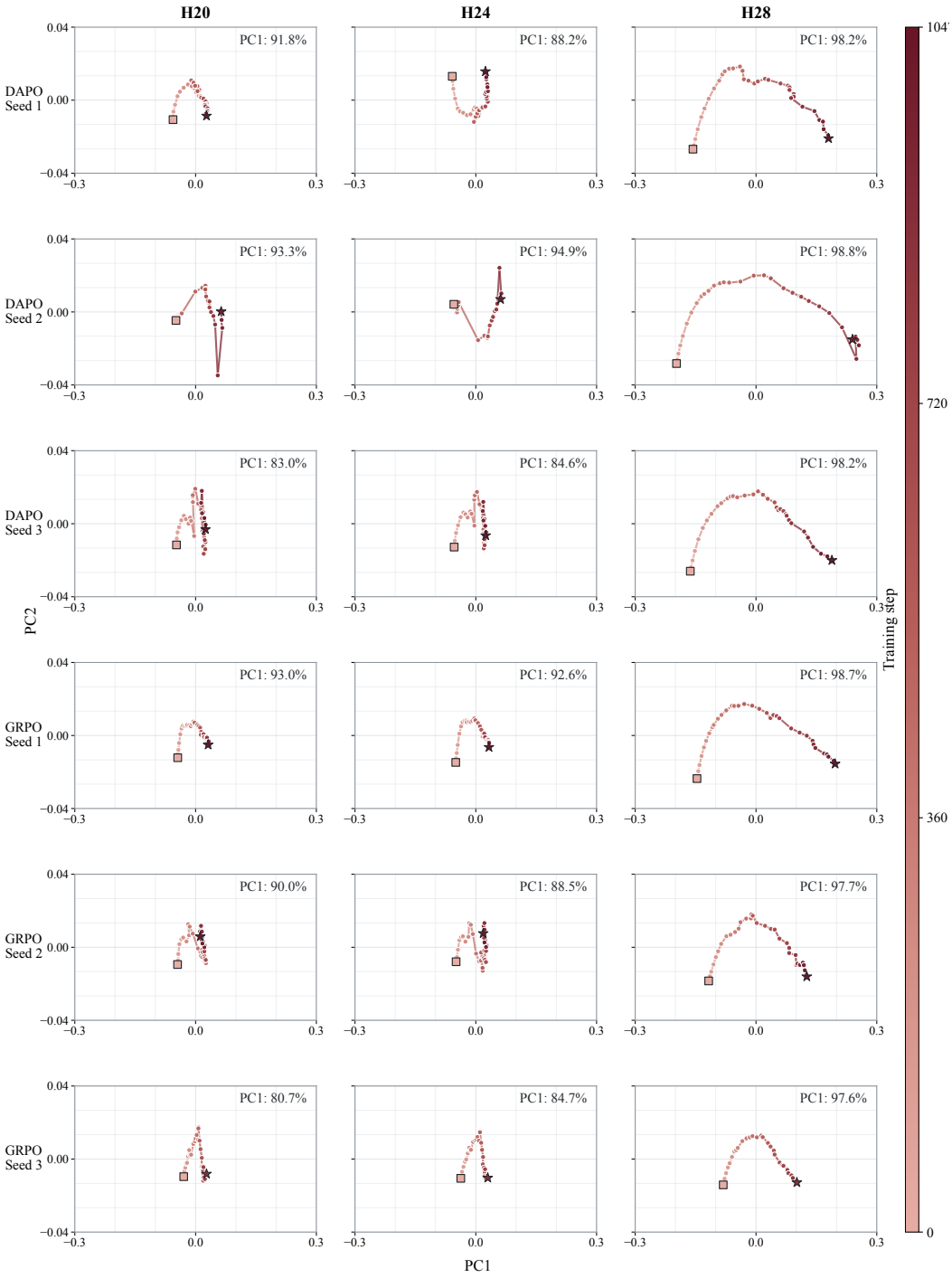


Figure 1: Three-layer PCA trajectories across training seeds. Each row is one training run, each column is one SAE layer, and color indicates training step. Each run and layer is fitted independently; all PCA panels use identical axis ranges and ticks.

The trajectory plots show how the whole representation moves; we next ask which coordinates account for that movement.

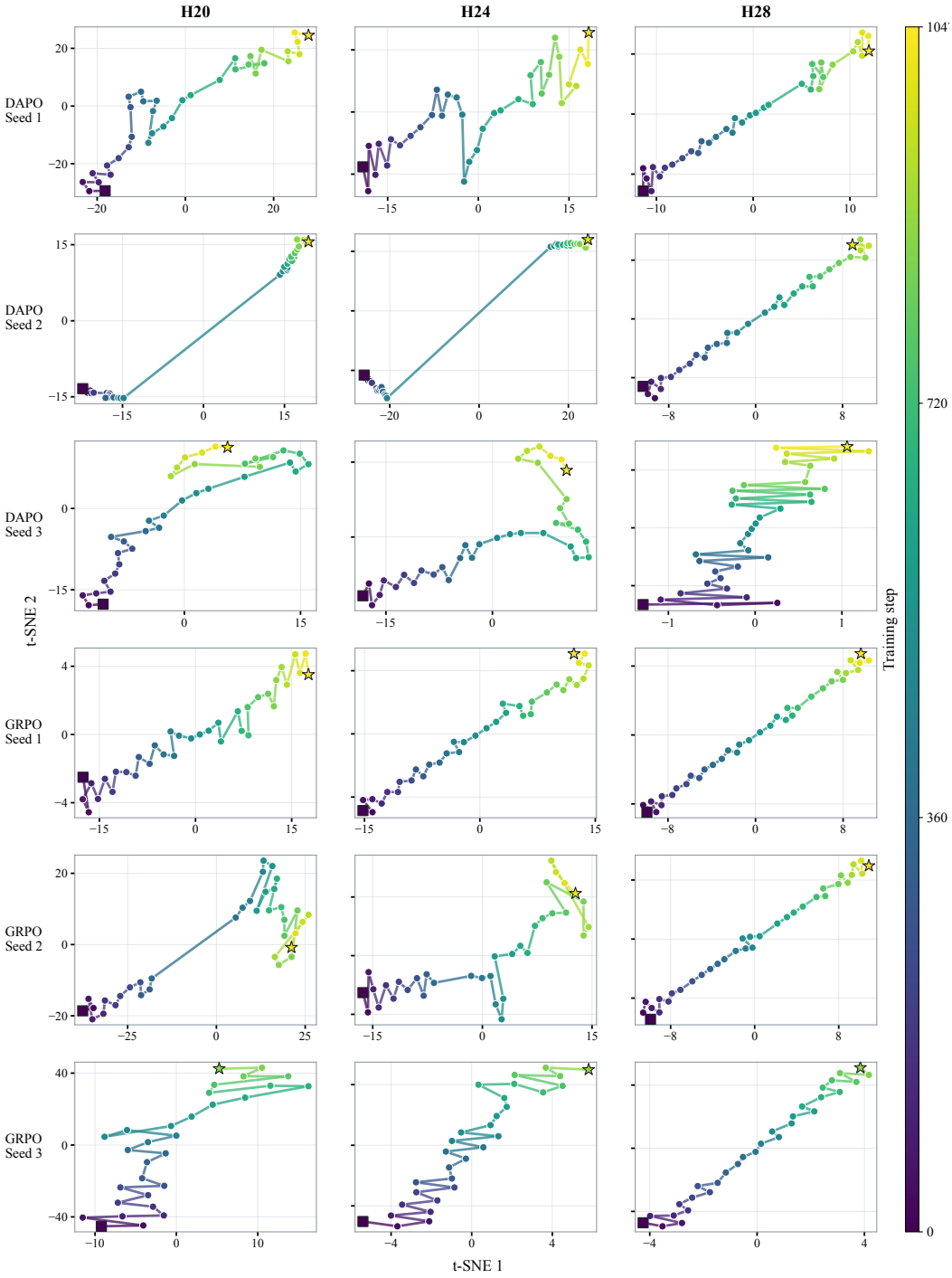


Figure 2: Three-layer t-SNE trajectories across training seeds. The top three rows are DAPO and the bottom three are GRPO; each layer shares the training-step color bar on the right. Axis ranges are adjusted separately for each t-SNE panel. Each run and layer is fitted independently; distances describe the trajectory within each panel.

A.3 Shared Top5 Features across Training Runs

Figure 3 shows the same five leading features, in the same order, across all six runs. Each run selects features on the discovery split and displays them on the confirmation split.

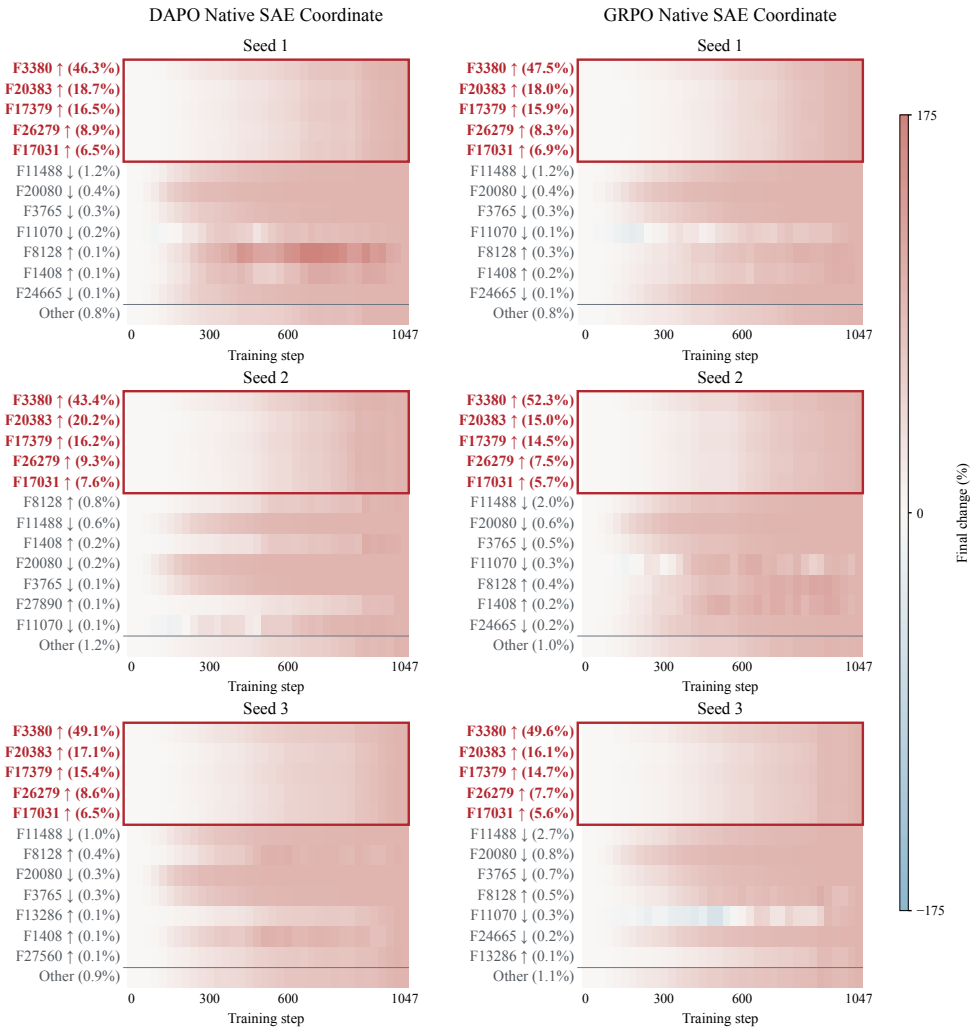


Figure 3: The same five features dominate representation change across all six training runs. Colors show the fraction of endpoint change completed in the native H28 SAE coordinates. Columns show DAPO and GRPO; rows show Seeds 1, 2, and 3. Each run independently selects its Top12 using the same 500 discovery problems, and the heatmap uses the other 500 confirmation problems with answer-token-weighted mean activations. Parentheses report each coordinate's share of squared endpoint change; "Other" aggregates the remaining 28,660 coordinates. All panels use the same color scale.

These features are F3380, F20383, F17379, F26279, and F17031. Across all six runs, the same five features account for 93.43–96.83% of squared net change on the confirmation split. The separation from the sixth-ranked feature is substantial but varies by run: the rank-five/rank-six ratio ranges from 1.35 to 5.58 for adjacent-fluctuation energy and from 1.37 to 3.11 for net change, with the maximum energy ratio of 5.58 observed for DAPO Seed 2 on the confirmation split. The identity of the sixth feature changes across runs (F8128, F11488, or F11070), whereas the same five leading features are recovered in both discovery and confirmation splits; their Top5 coverage differs by at most 0.20 percentage points between the two splits.

Coordinate composition of the endpoint change. Each run independently selects its Top12 by squared pre-RL-to-final change on the discovery split from Appendix C.3, then computes the heatmap on confirmation data. Means weight all answer tokens equally, so longer

answers contribute more. The full Top12 can differ across runs. Confirmation Top12 shares are 99.23/98.80/99.07% for DAPO Seeds 1/2/3 and 99.17/99.04/98.93% for GRPO.

A.3.1 Top5 Ranking and Coverage of Representation Change

All 36 combinations of six runs, two 500-problem splits, and three ranking rules recover the same ordered Top5: F3380, F20383, F17379, F26279, and F17031. Each combination ranks all 28,672 features independently by adjacent fluctuation energy, absolute net change, or activation range $Q_{90}-Q_{10}$. Net change compares the last-five and first-five state means. These five features occupy 0.0174% of the dictionary and account for 93.43–96.83% of squared net change on the confirmation split (Figure 4).

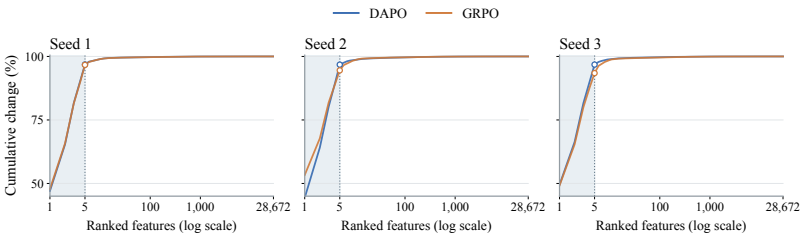


Figure 4: Sparse concentration of representation change across six training runs. Each panel shows one seed, with DAPO and GRPO drawn separately for direct comparison. Each curve reports cumulative squared net change as features are ranked by absolute net change on the 500-problem confirmation split. The first five features are highlighted, and the horizontal axis is logarithmic across all 28,672 features.

Table 2: Cumulative squared net-change share captured by the leading ranked features on the confirmation split. Each row keeps the seed separate; the entries give the cumulative share after the first 1–5 ranked features.

	Seed	Top-1	Top-2	Top-3	Top-4	Top-5
GRPO	1	47.81%	65.80%	81.61%	89.83%	96.67%
	2	53.33%	67.75%	81.84%	89.00%	94.52%
	3	49.16%	65.38%	80.06%	87.85%	93.43%
DAPO	1	47.05%	65.35%	81.77%	90.37%	96.83%
	2	44.38%	64.10%	80.18%	89.37%	96.77%
	3	49.47%	66.34%	81.69%	90.25%	96.79%

The ranking analyses use the DAPO feature-analysis texts from Appendix C.3. The fixed GRPO texts from Experiment 36 support the shared-trend and matched controls (Appendix C.3); the compared models come from the same DAPO training run.

Ranking agreement and separation in DAPO Seed 1. Using the 36 saved states of DAPO Seed 1 on the fixed texts, we separately compute adjacent fluctuation energy and endpoint change on each 500-problem split. Let $a_{k,f}$ be the mean activation of feature f at saved checkpoint k. Adjacent fluctuation energy is

$$E_f = \sum_{k=1}^{35} \left(a_{k,f} - a_{k-1,f}\right)^2. \tag{1}$$

For these ranking and coverage analyses, net change is the mean activation over the last five states (steps 930, 960, 990, 1020, and 1047) minus the mean over the first five states (pre-RL and steps 30, 60, 90, and 120). Ranking either by E_f or by absolute net change yields the same Top5 on both splits; the sixth-ranked features are F8128 and F11488, respectively. The discovery/confirmation ratios between ranks five and six are 3.8158/3.9819 for energy and 2.3540/2.4114 for net change. Ranking by $Q_{90}-Q_{10}$ also yields the same Top5, followed by F11488, with ratios 2.4541/2.5124. The fifth-ranked feature exceeds the sixth by 2.35–3.98 times across these measures.

Change carried by the Top5. For the Top5 set S, coverage is $\sum_{i\in S}\Delta_i^2/\sum_i\Delta_i^2$, where Δ_i is the five-point endpoint-mean difference defined above (Figure 4). For DAPO Seed 1, coverage by the top four/five/six features is 90.24/96.71/97.88% on the discovery split and 90.37/96.83/97.94% on the confirmation split. We further compare the full SAE-decoded change with the change decoded from the shared Top5 across all six runs, using the same frozen decoder and both 500-problem splits. For this comparison, changes are measured from pre-RL to each saved state, and both trajectories are projected onto the full pre-RL-to-step-1047 direction using the same denominator.

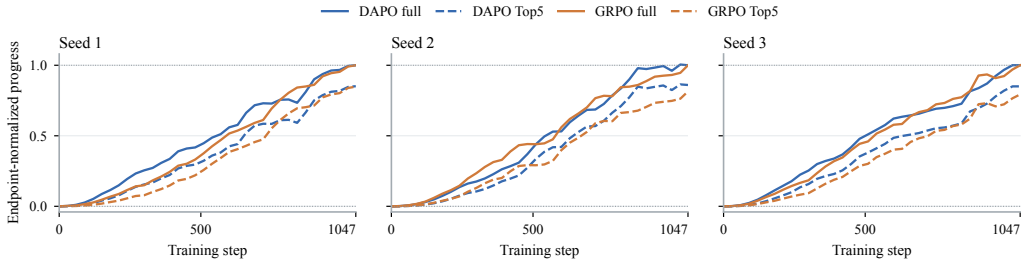


Figure 5: The shared Top5 tracks aggregate representation change across all six training runs. Panels use the same 500-problem confirmation split. Rows correspond to Seeds 1–3, with DAPO on the left and GRPO on the right. Each run pairs the full and Top5 SAE-decoded changes projected onto the full endpoint direction (blue and ochre) with an adjacent comparison of endpoint-direction cosine for 10,000 fixed random five-feature sets (gray) with the Top5 (ochre line and diamond). Arrows indicate the gap from the random-set 99th percentile to the Top5. The same random sets, selected under the original DAPO Seed 1 matching rule, are evaluated separately for each run.

Endpoint-direction cosine ranges from 0.9875 to 0.9937, trajectory correlation from 0.9961 to 0.9986, and squared residual reduction from 94.55% to 96.88%. On the confirmation split, the corresponding ranges are 0.9876–0.9937, 0.9963–0.9986, and 94.63–96.88%; the Top5 reaches 0.7935–0.8601 of the full endpoint projection.

A.3.2 Matched Random Controls and Detrended Tests

Matched random features. We retain the original 10,000 five-feature sets selected from the Rising features in Experiment 36 for DAPO Seed 1 and evaluate these same fixed sets under every training run. In the original selection, initial activation and absolute endpoint change were each divided into five bins, and every Top5 position was matched to one of 235 candidates from the same bins, excluding the original ten largest endpoint changes; the sampling seed was 20260861. Each run supplies its own saved activation means and endpoint direction, while feature-set identities and the SAE decoder remain fixed. None of the 10,000 random sets exceeds the Top5 in endpoint-direction cosine or squared residual reduction on either split of any run. Across the 12 run–split comparisons, the random-control 99th percentile ranges from 0.1337 to 0.1757 for cosine and from 0.55% to 1.64% for squared residual reduction, compared with Top5 values of 0.9875–0.9937 and 94.55–96.88%, respectively (Figure 5).

Synchronized change after removing the shared trend. For DAPO Seed 1, we prespecify a shared temporal component from the standardized trajectories in Experiment 36, explaining 68.18% of the variation, remove this fixed component, and compare the Top5 with the remaining aggregate change over 36 states on the 500-problem discovery and confirmation splits (Table 3).

Table 3: Top5 synchronization after detrending.

Statistic	Discovery, 500 problems	Confirmation, 500 problems
Endpoint-direction cosine	0.8321	0.8575
Squared residual reduction	58.65%	63.70%
Correlation of adjacent-stage changes	0.9468	0.9523

A.3.3 Replication across Training Runs and Individual Feature Trajectories

Figures 6 and 7 use the confirmation data and answer-token-weighted means defined above. The first normalizes each feature by its pre-RL-to-step-1047 change and projects aggregate change onto the full endpoint direction; the second shows unnormalized Top5 activations and F8128. The same five features rise across runs, with different magnitudes and local trajectories.

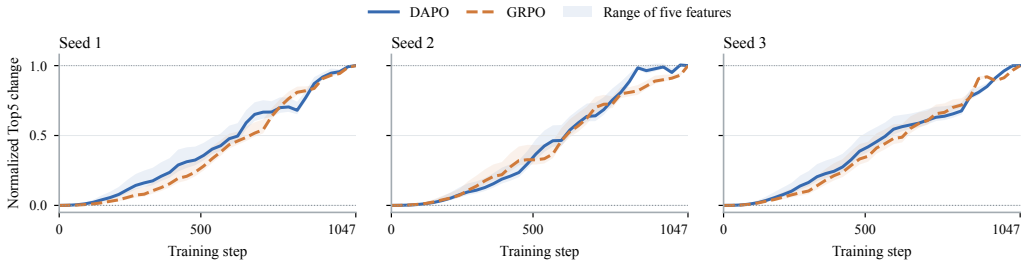


Figure 6: Individual Top5 features and aggregate representation change across all six runs. Colored curves show each feature's mean activation change normalized to its own endpoint change; the dashed black curve shows full decoded progress. All available saved checkpoints through step 1047 are shown.

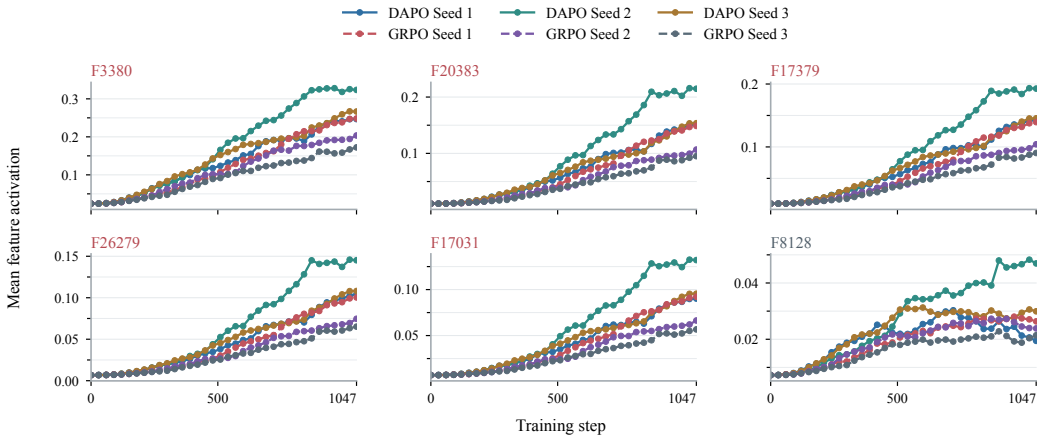


Figure 7: Mean activation trajectories of the shared Top5 and comparison feature F8128 across six training runs. Each panel shows one feature. Solid and dashed curves denote DAPO and GRPO, respectively; feature-specific vertical scales retain the absolute activation values.

Independent-code validation. We evaluate 48 Python 3.12 standard-library functions at step 1047, split by hash into disjoint 24/24 sets. These texts contain no correctness, reward, or test information and do not overlap the MBPP+ interpretation data. Table 4 reports the first principal component of the five standardized activations. All loadings have the same sign; their cross-split cosine is 0.99999. The effect compares scaffold tokens (comments, keywords, and identifiers) with literal tokens (numbers, operators, and strings), showing stronger shared activation on scaffold tokens in both splits.

Table 4: Independent-code validation of the shared Top5 factor.

View	PC1 variance	Same sign for all loadings	Scaffold-minus-literal effect (95% interval)
Discovery, 24 functions	91.48%	Yes	0.282 [0.149, 0.415]
Confirmation, 24 functions	91.17%	Yes	0.234 [0.083, 0.381]

Feature-specific code preferences. F3380 and F26279 retain distinct code preferences beyond the shared activation pattern (Table 5).

Table 5: Code preferences and paired-edit responses. Values are standardized differences with 95% intervals. Natural-code comparisons remove the component predicted by the other four features; paired edits compare mean activations over eight program pairs.

Feature	Natural-code comparison	Discovery	Confirmation
F3380	Keywords − terminal tokens	0.201	0.179
		$[0.061, 0.341]$	$[0.033, 0.332]$
F26279	Keywords/strings − numbers/terminal tokens	0.339	0.391
		$[0.275, 0.398]$	$[0.313, 0.466]$
Feature	Paired edit	Change after edit [95% interval]	
F3380	Remove closing code fence	$+0.075$ $[0.066, 0.084]$	
F26279	Remove closing code fence	$+0.035$ $[0.027, 0.042]$	
Shared score	Add comments	-0.465 $[-0.600, -0.329]$	

The other three features have no consistently supported individual interpretation across both checks. The comment-edit result distinguishes natural-code preference from the effect of adding comments.

The Top5 analysis identifies the dominant coordinates. We then test whether the direction of change itself remains stable across runs and independently held-out problems.

A.4 Persistent Directions across Training and Problem Sets

Across three seeds of each algorithm, 91.4–100% of adjacent updates point toward their run's endpoint direction. Median directional cosine ranges from 0.859 to 0.950, and path directness ranges from 83.4% to 94.2% (Figure 8). Thus, sustained directional progress recurs across both algorithms and all six training runs. For each run, let a_k be the mean activation on the confirmation split, using the aggregation in Appendix A.5.1, and let $D = a_T - a_0$. Cumulative progress is $(a_k - a_0)^\top D / \|D\|_2^2$, adjacent directional alignment is the cosine between $(a_k - a_{k-1})$ and D, and path directness is $\|D\|_2 / \sum_k \|a_k - a_{k-1}\|_2$.

Figure 3 uses discovery-selected Top12 coordinates and independently measured confirmation values. The coordinate selection and endpoint normalization are specified in Appendix A.3.

GRPO consistency across problem splits. Across the 35 intervals, the cosine between adjacent-update directions on the two 500-problem splits ranges from 0.998801 to 0.999981, with median 0.999869. Their endpoint-direction cosine is 0.999974. Projecting both trajectories onto the discovery-fitted endpoint direction gives a cumulative-progress correlation of 0.999995.

DAPO consistency across problem splits. For DAPO Seed 1, both splits are projected onto the endpoint direction fitted on the discovery split (Figure 9). Across 35 intervals, their adjacent-update cosine exceeds 0.976 and cumulative-progress correlation is 0.99999. Both algorithms therefore recover nearly identical training directions from independent problem sets.

The direction checks establish consistency at the level of the full update. We next classify individual features by how their activation changes over the entire training path.

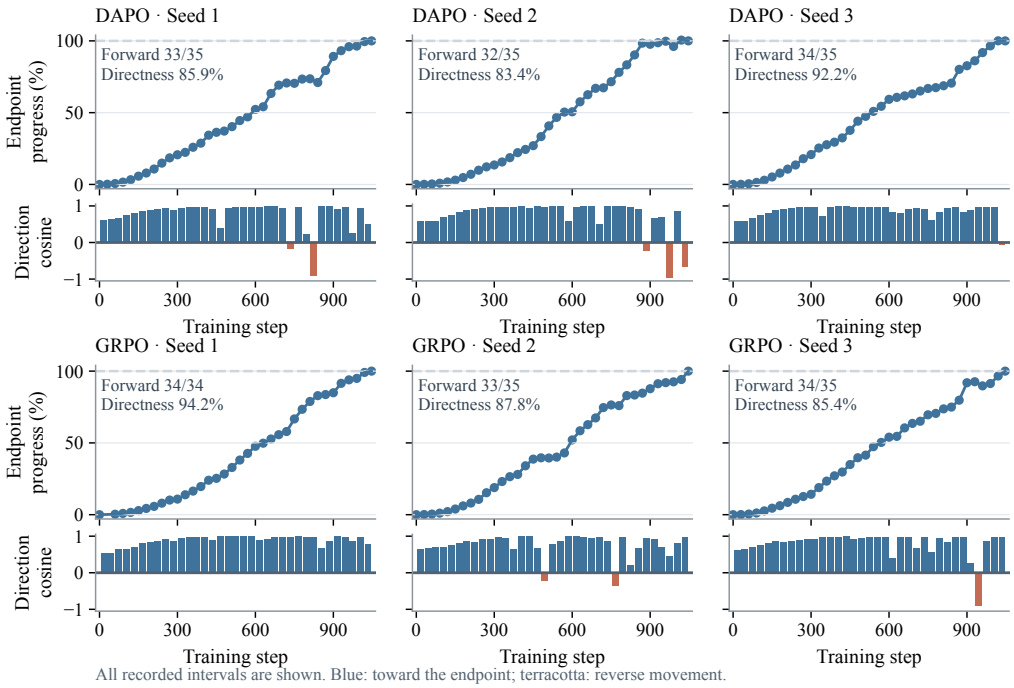


Figure 8: Persistent progress along the H28 endpoint direction across six training runs. Rows show DAPO and GRPO; columns show Seeds 1–3. Within each panel, the upper curve reports cumulative progress and the lower bars report the directional cosine of adjacent recorded updates. Blue bars indicate forward movement and terracotta bars indicate reverse movement. Annotations give the forward-update count and path directness.

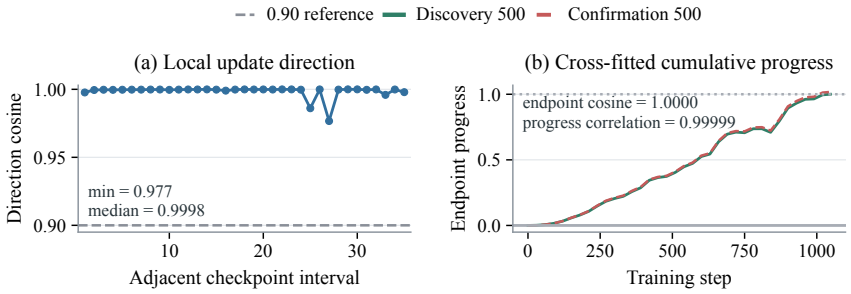


Figure 9: Independent problem sets recover the same H28 training direction and nearly identical cumulative progress in DAPO Seed 1. (a) Cosine between discovery- and confirmation-split change directions for every adjacent-checkpoint interval. (b) Cumulative progress of both splits along the endpoint direction fitted only on the discovery split.

A.5 Feature Evolution

A.5.1 Definitions and Screening of Four Feature Classes

We use the feature discovery and confirmation data from Appendix C.3.

We process the same texts at 36 states: pre-RL, every 30 steps from 30 through 1020, and step 1047; only valid answer tokens are included, excluding prompts and padding. For feature f at state s, we sum positive activations over a problem set and divide by the total number of answer tokens to obtain mean activation $a_{s,f}$, treating inactive positions as zero. The first five states, pre-RL, 30, 60, 90, and 120, define the early set E, and the last five, 930, 960, 990, 1020, and 1047, define the late set L. We compute net change and trajectory

variation as

$$\Delta_f = \frac{1}{5}\sum_{s\in L} a_{s,f} - \frac{1}{5}\sum_{s\in E} a_{s,f}, \quad m_f = \max\{\text{median}_s\, a_{s,f}, 10^{-12}\}, \tag{2}$$

$$u_f = \Delta_f / m_f, \quad v_f = \big(Q_{90}(a_{:,f}) - Q_{10}(a_{:,f})\big)/m_f, \quad \rho_f = \text{Spearman}(a_{:,f}, s). \tag{3}$$

where u_f is net change, v_f is trajectory variation, and ρ_f is the rank correlation with training step. Rising requires $u_f > 0.10$ and $\rho_f \geq 0.70$, while Falling requires $u_f < -0.10$ and $\rho_f \leq -0.70$. Features with small endpoint change, $|u_f| \leq 0.10$, are classified as Fluctuating when $v_f \geq 0.10$ and Stable when $v_f < 0.10$. Features that satisfy none of these conditions remain unclassified.

We measure activation level by $Q_{90}(a_{:,f})$ and retain features in the top 10% by activation, or in the top 10% by change or variation and the top 25% by activation. Thresholds are computed independently on each split, and both membership and class must agree. Each split is bootstrapped 200 times, and the class and inclusion status must be retained in at least 190 resamples, using seeds 20260837 and 20260838. At the peak state, each split must contain at least 437/438 activations across at least 20 problems, with no problem contributing more than 20%. Each feature must also cover at least eight token types, three of four positional bins, and three of four problem-length groups; no single token, position decile, or problem-length group may contribute more than 50%, 50%, or 80%, respectively. The final screen retains 2,365 features, or 8.25% of the dictionary. Table 6 gives the count after each stage and the overlapping auxiliary labels, and Table 7 gives the four class counts.

Table 6: Feature screening.

Screening step	Remaining features
Full dictionary	28,672
Class agreement across splits and activation/change criteria	3,056
Stable class and inclusion across bootstrap samples	2,457
Sufficient activations and problem coverage at the peak state	2,455
Sufficient diversity across tokens, positions, and problem lengths	2,365

Among the final 2,365 features, overlapping auxiliary labels identify 2,046 with high variation, 2,001 with large endpoint change, 1,886 with high activation, and 397 with medium activation; none has low activation.

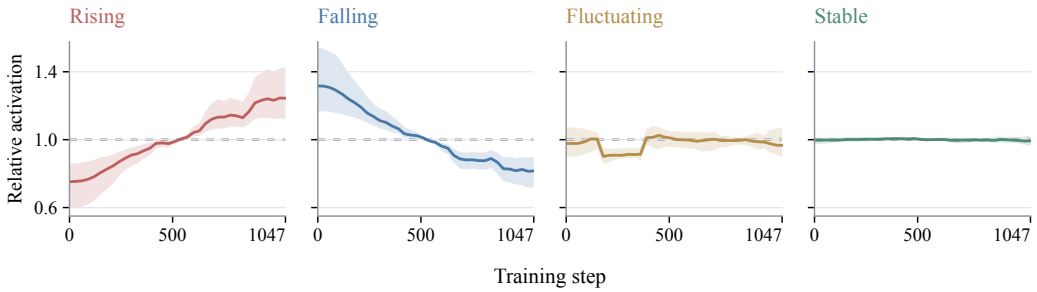


Figure 10: Evolution of the four feature classes over 36 training states. Each panel shows the within-class median (line) and 25th–75th percentiles (shading), using a common vertical scale. Each feature is divided by its own median activation over training.

Rising and Falling have substantial net changes. Fluctuating and Stable both have near-zero net changes but differ in variation, as defined above (Table 7).

Figure 10 summarizes all 2,365 features that pass the final screen. Each feature's 36-point trajectory is divided by its median activation over the full trajectory, after which the median and interquartile range are computed by class at each checkpoint.

Table 7: Counts and typical changes in the four feature classes. Shares use the 2,365 selected features. Net change and variation range are within-class medians on the 500-problem confirmation split, expressed relative to each feature's median activation.

Category	Count	Share	Net change	Variation range
Rising	916	38.73%	+49.38%	47.56%
Falling	1,120	47.36%	−48.11%	45.39%
Fluctuating	131	5.54%	+0.29%	15.93%
Stable	198	8.37%	+0.28%	5.90%
Total	2,365	100%	—	—

A.5.2 Directional Consistency across the Full Dictionary

For each 500-problem split, we select features from the full dictionary of 28,672 with $|u_f| > \tau$, where $\tau \in \{5\%, 10\%, 20\%\}$, and report the fraction satisfying $\text{sign}(u_f)\rho_f \geq 0.70$. This analysis includes features before four-class screening (Table 8).

Table 8: Directional consistency of feature changes.

Net-change threshold	Discovery	Confirmation
5%	20,967/25,461 (82.35%)	20,945/25,523 (82.06%)
10%	19,686/22,428 (87.77%)	19,668/22,469 (87.53%)
20%	15,705/16,872 (93.08%)	15,813/17,078 (92.59%)

Raising the net-change threshold from 5% to 20% increases directional consistency from about 82% to 93% on both splits (Table 8).

At the 10% threshold, among features with negative net change, 10,263/11,570 (88.70%) in the discovery split and 10,059/11,423 (88.06%) in the confirmation split satisfy $\rho_f \leq -0.70$. When weighted by each feature's squared endpoint change, directionally consistent features account for 99.7718% of the total in the discovery split and 99.7767% in the confirmation split.

A.5.3 Approximately Monotonic H28 Features over the Full Trajectory

We apply the approximate-monotonicity screen to the 6,869 H28-K192 candidates that pass the usage and artifact screens in Experiment 36. For each 36-point log-activation trajectory, we orient the endpoint change positively, require reverse motion to be at most 10% of total motion and endpoint-relative change to be at least 10%, and retain the top 5% by absolute endpoint change. For decreasing features, this normalization reverses the sign of the raw activation change for display, so an upward curve denotes progress toward the final decrease rather than an increase in raw activation. This scan yields 124 features, of which 42 increase and 82 decrease. These approximately monotonic trajectories allow limited local reversals. After freezing the feature IDs and directions, we evaluate them on a separate set of 1,000 fixed texts split into 500 discovery and 500 confirmation problems: 110/124 retain the same endpoint direction in both splits and keep reverse motion below 10%.

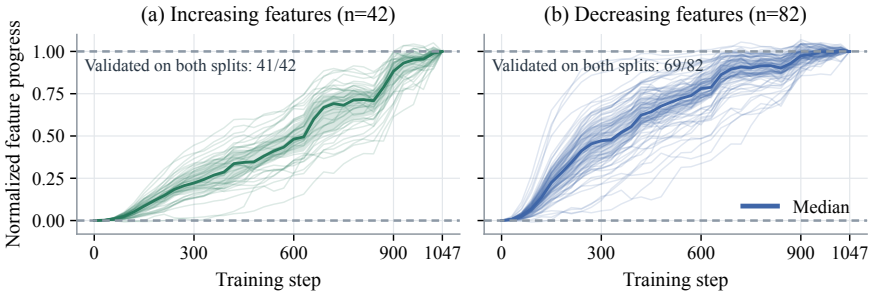


Figure 11: Complete trajectories of approximately monotonic H28-K192 features. Each thin line is one feature, normalized from 0 to 1 along its endpoint direction; the thick line and shaded region show the median and interquartile range. The annotation reports how many features pass the direction and reverse-motion criteria in both independent 500-problem views.

A.5.4 Predicting Later Directions from Change through Step 60

The main-text direction curve reports GRPO Seed 1; this subsection first reports DAPO Seed 1 and then gives the GRPO comparison. Each run selects its own features from its first 60 steps. For DAPO Seed 1, we select features using only pre-RL, step 30, and step 60 on the same discovery and confirmation splits. Define $d_f = a_{60,f} - a_{0,f}$ and $m_f^{(60)} = \max\{\text{median}(a_{0,f}, a_{30,f}, a_{60,f}), 10^{-12}\}$; the activation, absolute-change, and range quantile thresholds from the preceding section are applied to these first three states. The direction must agree between splits, and both direction and inclusion must be retained in at least 190 of 200 bootstrap resamples. The thresholds for activation count, problem count, and token/position/problem-length concentration are inherited from the preceding section, with concentration checked on pre-RL records. This yields 2,335 features, 1,008 increasing and 1,327 decreasing, selected independently of the whole-trajectory classification. In the confirmation split, 98.72%, 90.49%, and 81.63% continue along the early direction at steps 90, 300, and 1047, respectively; Table 9 gives all subsequent states. For a later state $s > 60$, directional continuation is defined by $d_f(a_{s,f} - a_{60,f}) > 0$. Table 9 also reports the fraction that continues in the same direction with $|a_{s,f} - a_{60,f}|/m_f^{(60)} > 5\%$; both fractions use the fixed denominator 2,335.

We also require the early historical change $|d_f|/m_f^{(60)}$ to exceed 1%, 5%, or 10%, checking both direction and magnitude in every bootstrap sample while keeping all other criteria unchanged. Table 10 reports the complete member count at each threshold and the later same-direction fraction in the confirmation split.

From the marginal positive and negative proportions of historical and future directions, random pairing would yield approximately 51% directional agreement. The statistical unit is the feature, and the two problem splits are computed separately.

Across all 33 subsequent checkpoints, the aggregate change from step 60 has positive cosine similarity with the early direction in the confirmation split, using the 2,335 selected features normalized by their early activation scales.

GRPO comparison. Figure 2 applies the same early-selection protocol to GRPO Seed 1. Using only steps 0, 30, and 60 selects 2,164 features: 952 increasing and 1,212 decreasing; their identities and early signs are fixed for all 33 later checkpoints. Trend retention is the percentage of features whose later net change continues the early increase or decrease, satisfying $d_f(a_{s,f} - a_{60,f}) > 0$ on the confirmation split; the curves report this percentage overall and separately by early sign. At step 1047, these fractions are 80.68% overall, 72.06% for early increases, and 87.46% for early decreases. The gray interval marks the first 60 steps; the selected features and their early directions remain fixed throughout subsequent training. The dashed reference is the exact expected agreement under uniformly permuted future-feature identities: $(n_+ m_+(s) + n_- m_-(s))/N^2$, where $n_\pm$ and $m_\pm(s)$ count early and later signs among the $N = 2{,}164$ selected features. Its final value is 51.35%; the pale gray fill extends from zero to this reference curve.

Table 9: Early directions and subsequent changes.

Step	Disc. same	Conf. same	Disc. same, $> 5\%$	Conf. same, $> 5\%$
90	98.67%	98.72%	5.95%	6.25%
120	96.40%	96.23%	30.62%	31.56%
150	95.50%	95.25%	52.68%	53.49%
180	94.26%	93.88%	61.93%	62.66%
210	91.69%	91.56%	67.24%	67.19%
240	92.63%	92.72%	74.30%	74.43%
270	91.78%	91.48%	76.19%	76.15%
300	90.71%	90.49%	76.83%	77.22%
330	90.02%	89.55%	77.56%	77.69%
360	89.85%	89.64%	79.79%	79.53%
390	90.32%	90.15%	80.60%	80.39%
420	91.09%	90.58%	82.70%	82.87%
450	90.45%	90.41%	83.17%	83.25%
480	91.18%	90.92%	83.04%	83.30%
510	89.98%	89.94%	82.40%	82.44%
540	88.52%	88.74%	81.63%	81.50%
570	88.74%	88.57%	81.37%	81.37%
600	88.48%	88.09%	81.54%	81.80%
630	88.27%	88.22%	82.14%	82.36%
660	88.05%	88.05%	82.83%	83.00%
690	87.58%	87.67%	82.57%	82.61%
720	88.22%	87.97%	82.74%	82.74%
750	87.54%	87.28%	82.18%	81.93%
780	87.75%	87.45%	82.36%	82.44%
810	87.37%	87.19%	81.50%	81.71%
840	86.64%	86.38%	80.69%	80.81%
870	85.87%	86.00%	80.43%	80.39%
900	85.74%	85.57%	81.33%	81.20%
930	84.84%	84.63%	79.61%	79.87%
960	84.67%	84.71%	80.21%	80.21%
990	83.25%	83.30%	77.99%	78.33%
1020	82.74%	82.31%	77.13%	77.22%
1047	81.54%	81.63%	76.57%	76.66%

Table 10: Magnitude thresholds for early change.

Threshold	Increasing	Decreasing	Step 90	Step 300	Step 1047
0%	1008	1327	98.72%	90.49%	81.63%
1%	302	419	100.00%	95.15%	89.32%
5%	8	22	100.00%	90.00%	90.00%
10%	1	0	100.00%	0.00%	0.00%

Replication across three GRPO training runs. Applying the same early-selection procedure independently to Seeds 2 and 3, final directional retention across all three seeds is $84.08\% \pm 2.97\%$ (mean $\pm$ sample standard deviation). In each run, aggregate changes remain positively aligned with the early direction at all 33 subsequent checkpoints. Table 11 reports the per-seed feature counts, retention, and random-pairing comparison.

Table 11: Early-direction retention across three GRPO training runs. Each run selects features independently from steps 0, 30, and 60 using the same selection procedure. Retention and the random-pairing reference are measured at step 1047 on the confirmation split.

Seed	Selected features	Direction retained	Random pairing
1	2,164	80.68%	51.35%
2	1,908	86.22%	50.69%
3	1,951	85.34%	50.72%

These trajectory classes describe when features rise or fall; the final part asks what kinds of code contexts those changes are associated with.

A.6 Feature Effects

A.6.1 Code Contexts of Rising and Falling Features

For the code-context data in Table 16, responses are greedily generated at step 1047 and split by even/odd row number; the SAE reads answer tokens. Each feature is described by four views: preferred code or syntax category, token-type distribution, the problems on which it activates, and the SAE decoder direction. The last three views are each reduced to 32 dimensions; every view is normalized per feature and then concatenated with equal weight. On the discovery split, K-means is evaluated at 8, 12, 16, 24, 32, and 48 clusters with 50 initializations each and a clustering random seed of 20260827, distinct from the RL training seed. After requiring at least ten features per cluster, we select the clearest partition by silhouette score, preferring fewer clusters in a tie. The selected partition has 12 clusters, a silhouette score of 0.06338, and a minimum cluster size of 36 features.

For each cluster, enrichment for a code category is the fraction of activation carried by tokens in that category divided by the category's frequency in the text. The most preferred category must agree between discovery and confirmation, enrichment must be at least 1.25× in both, and confirmation must involve at least five cluster features and ten problems. Table 12 summarizes the 12 clusters by code preference. Discovery and confirmation preferences agree for 11 clusters, and feature-level label agreement is 734/1120 (65.5%).

Table 12: Code preferences of Falling features.

Preferred code context	Feature clusters	Total features
Comments	3	327
Strings	2	175
Numbers	1	116
Identifiers	1	80
Python keywords	2	278
Whitespace, including newlines and indentation	2	97
Mixed context	1	47
Total	12	1120

Table 13 gives three confirmation examples, preserving the original code snippets and marking high-activation tokens in bold. For the number category, F27753 has the additional confirmation record on Mbpp/161: 6, 7, 8, 9, 10.

Table 13: Code examples for Falling features.

Feature / MBPP problem	Category	Code-context excerpt
F13963 / 623	Comment	the n-th power of each number
F6501 / 751	Identifier	if right < n and arr[i] > arr[right]:
F4387 / 725	Keyword	def extract_quotation(s):

Rising features. Rising features use the same code-context data and split as Falling features above. For the fixed set of 916 Rising features, a problem is counted as covered if any answer token has positive activation, with each problem counted once (Table 14).

Table 14: Problem coverage of Rising features.

Statistic	Discovery	Confirmation
Total features	916	916
Median problems covered	57	57
Features covering at least 32 problems	773	758
Features covering at least 48 problems	591	592
Features covering all 64 problems	253	254

Features are divided into eight disjoint groups by activation magnitude, endpoint change, and peak time. Within each group on the discovery set, features are ranked in descending order by problems covered, activation count, and mean positive activation, followed by ascending ID; up to 12 representatives are fixed and then evaluated on the confirmation texts. Each feature receives equal weight when computing a group's content preference. A group receives a code-category label only when the most frequent category agrees across both splits, enrichment is at least 1.25× in each, and at least half of the members are active and jointly cover 20 problems; confirmation examples must also involve at least three features and ten problems. Some groups activate more often near strings, identifiers, operators, or punctuation. Both Rising and Falling classes contain features that prefer strings or identifiers, showing that distinct features associated with the same code category can increase or decrease separately. Change directions and magnitudes distinguish these training effects within a code category and guide the construction of PUSH targets.

B PUSH Implementation

B.1 Target Selection

Decomposition followed by change selection. SAE decomposition separates training-induced activation changes into feature coordinates. PUSH selects the largest absolute pre-to-current changes to construct per-token targets. For a fixed coordinate budget, this rule minimizes squared approximation error in SAE space. TIES-Merging also retains large changes, but in model parameters for weight merging (Yadav et al., 2023); PUSH uses activation changes to define targets.

For a fixed token, let $\delta = z_{\text{now}} - z_{\text{pre}} \in \mathbb{R}^m$ denote the change in the shared SAE. Let P_S retain only the features in S and set all others to zero; if at most k features can be retained, choose the nonzero entries with the largest absolute changes, denoted S_k:

$$S_k \in \arg\min_{|S|\le k} \|\delta - P_S\delta\|_2^2, \qquad \|\delta - P_{S_k}\delta\|_2^2 = \sum_{j>k} |\delta|_{(j)}^2, \tag{4}$$

Here $|\delta|_{(j)}$ is sorted in descending order, and all nonzero entries are retained when fewer than k exist. This follows from $\|\delta - P_S\delta\|_2^2 = \|\delta\|_2^2 - \sum_{i\in S}\delta_i^2$: maximizing the retained squared magnitude minimizes the residual error. When magnitudes are tied, any tied selection attains the minimum.

Target error from selecting fewer features. Let $\gamma \ge 0$ be the extrapolation scale in this analysis, with the full continuation target $y = [z_{\text{now}} + \gamma\delta]_+$ and the truncated target $y_k = [z_{\text{now}} + \gamma P_{S_k}\delta]_+$. Clipping negative values to zero cannot increase the Euclidean distance between the two vectors, so

$$\|y - y_k\|_2 \le \gamma\|\delta - P_{S_k}\delta\|_2. \tag{5}$$

Thus, the selection rule minimizes this upper bound on target error. Table 4 compares task gains under the two coordinate-selection rules. Setting $\gamma = 1$ yields the approximation bound used for the B_t directional target in PUSH. For a linear SAE decoder W_{dec}, the corresponding decoded difference also satisfies $\|W_{\text{dec}}(y - y_k)\|_2 \le \|W_{\text{dec}}\|_2\|y - y_k\|_2$.

The empirical feature-budget comparison is reported in Appendix E.3.

B.2 Frozen-SAE Loss

The branch losses are defined in Equation (4); the following details specify their implementation.

Hidden states are converted to the SAE encoder's data type; sae_dense returns $r(h)$ from Equation (1), and sae_codes additionally applies TopK192 to obtain $z(h)$. ν is read from the SAE checkpoint field activation_norm.

The recorded training runs apply the same positive, θ-independent batch scaling to both branches before gradient clipping. Because every update has a gradient norm above the

clipping threshold and is clipped to 1, this scaling changes neither the normalized gradient direction, up to numerical precision, nor the targets in Equation (3).

B.3 Optimization and Merge

Cached pre-RL activations supply the historical reference without keeping the pre-RL model weights in GPU memory. Let $n_g = |\mathcal{G}|$ be the number of fixed guidance sequences and q the number of sequences per batch. We partition the guidance set into u batches $G_1, \ldots, G_u$, with $n_g = uq$, and take one update per batch in each branch. The main code experiments use $(n_g, u, q) = (64, 8, 8)$ and shuffle with seed 16384570; the mathematical experiments use $(128, 16, 8)$. For code, the rank-32, alpha-64 adapter comprises 392 factor tensors across 196 modules in 28 blocks, with the loss evaluated at H28 and accumulated without averaging over problems. Both branches initially represent the current model, as specified below. Each branch uses the following optimizer and gradient-clipping rule:

$$g_j^{(b)} = \nabla_\theta \mathcal{L}_j(\theta_j^{(b-1)}; G_b), \qquad j \in \{A, B\}, \tag{6}$$

$$\bar{g}_j^{(b)} = g_j^{(b)} \min\left(1, \frac{1}{\|g_j^{(b)}\|_2}\right), \tag{7}$$

$$\xi = (\eta = 5\times10^{-6},\ \beta_1 = 0.9,\ \beta_2 = 0.999,\ \epsilon = 10^{-8},\ \lambda = 0), \tag{8}$$

$$\theta_j^{(b)} = \mathrm{AdamW}\left(\theta_j^{(b-1)}, \bar{g}_j^{(b)}; \xi\right), \qquad b = 1, \ldots, u. \tag{9}$$

Resuming RL after PUSH. PUSH uses temporary LoRA branches to fit its representation targets while holding the current RL model fixed. The effective weights at the continuation starting point are

$$W_{\text{resume}} = W_{\text{now}} + \Delta W_{\text{PUSH}}, \qquad \Delta W_{\text{PUSH}} = s\big(\alpha \Delta W_A + \beta \Delta W_B\big). \tag{10}$$

Weight write-back in the experiments. In the implemented write-back, let W_{base} denote the frozen base-model weights, and let U_0, U_A, and U_B be the effective weight contributions of the original adapter and the two fitted branch adapters, respectively, at a LoRA-injected module. Each branch starts from a copy of U_0, so its contribution to PUSH is its change relative to U_0. With the fixed method coefficients,

$$\Delta W_{\text{PUSH}} = s\big[2(U_A - U_0) + (U_B - U_0)\big], \qquad W'_{\text{base}} = W_{\text{base}} + \Delta W_{\text{PUSH}}. \tag{11}$$

The original adapter's two factor matrices, rank, and LoRA scaling remain unchanged during write-back. Consequently,

$$W'_{\text{base}} + U_0 = W_{\text{base}} + U_0 + \Delta W_{\text{PUSH}} = W_{\text{PUSH}}. \tag{12}$$

RL continuation and direct evaluation use these effective weights.

The saved RL optimizer and training state are restored separately from the temporary PUSH fitting state (Appendix C.1).

C Experimental Setup

C.1 Model and Evaluation

Table 15 gives the code defaults; deviations are stated with each experiment, and Appendix C.4 gives the mathematical settings.

A response is correct only if it passes every test for the problem. The subsequent component and strength analyses use DAPO Seed 2 and GRPO Seed 2 on H200 GPUs; some historical feature analyses use H20 GPUs. Coding rollout-saving results include Seeds 1–3 of both DAPO and GRPO.

Checkpointing and RL continuation. Each continuation starts from the full RL checkpoint, including model parameters and training state. After writing back PUSH (Appendix B.3), we restore the saved optimizer, learning-rate scheduler, random number generator states, and data cursor, then resume the original RL configuration. Temporary PUSH optimizers are discarded.

Table 15: Default RL training, PUSH, and evaluation settings for code RLVR. Experiment-specific deviations are stated in the corresponding sections.

Item	Setting
Base model	Qwen2.5-Coder-7B-Instruct; code RLVR with DAPO and GRPO
RL framework	verl, commit 8fdc4d3f; one node with one GPU and the vLLM rollout backend
RL training seeds	DAPO and GRPO: Seed 1 = 20260803; Seed 2 = 20260725; Seed 3 = 1347429771
Shared representation	H28 by default; all code SAEs use K192 and 28,672 features. H20/H24/H28 use frozen checkpoints 10752/11264/7168, respectively.
PUSH coordinate budget	$D = 96$
RL optimization	Train batch 32; PPO mini-batch 32; PPO micro-batch 2 per GPU; log-probability micro-batch 4; LoRA rank 32 and alpha 64; AdamW learning rate 5×10^{-6} with 11 warmup steps; gradient checkpointing; prompt and response limits 2,048 tokens; 1,047 training steps with checkpoints every 30 steps and automatic resume
Algorithm settings	GRPO uses clip $(0.20, 0.20)$ with sequence-mean token-mean loss; DAPO uses clip $(0.20, 0.28)$ with token-mean loss, group filtering by accuracy, and the 512-token overlong buffer with penalty factor 1.0
PUSH guidance and optimization	Batching, branch optimizer, and merge settings in Appendix B.3
Test split evaluation	test cases per problem: mean 15.04, median 15; greedy $n = 1$; at most 2,048 generated tokens; BF16, SDPA, batch size 32
MBPP/MBPP+ evaluation	378 held-out problems; greedy $n = 1$; at most 2,048 generated tokens; BF16, SDPA, batch size 8
Sampling accounting	Complete pre-filter responses counted: GRPO generates $32 \times 8 = 256$ per step; DAPO uses each run's logged generation counters. Generated tokens are counted separately; PUSH fitting adds no RL rollouts.

C.2 SAE Training

The three code-model SAEs read residual-stream states from their corresponding layers in the frozen pre-RL model (Figure 12); Table 16 specifies the data. Each SAE uses an $8\times$ dictionary expansion, mapping 3,584-dimensional hidden states to 28,672 features, with per-token TopK sparsity at $K = 192$. The training objective combines hidden-state reconstruction error with a dead-feature AuxK term. Training uses batches of 4,096 states, Adam with a learning rate of 2×10^{-4}, 1,000 warmup steps, decoder normalization, gradient clipping at 1.0, and an AuxK weight of 1/32. The SAE checkpoint is selected using prespecified reconstruction metrics on separate selection data; the 450-problem reconstruction evaluation set is used only for evaluation.

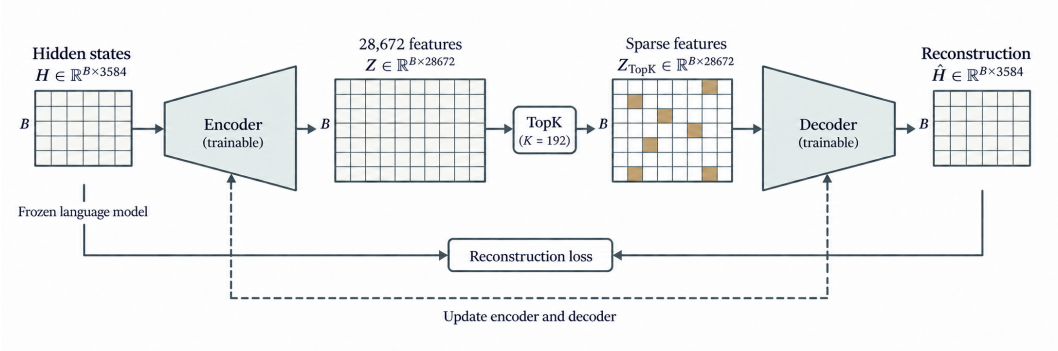


Figure 12: SAE training. Hidden states from the frozen language model are encoded into an expanded feature space, sparsified by TopK, and decoded to reconstruct the input. The encoder and decoder are trained while the language model remains frozen. Matrices are schematic; B denotes the number of hidden states in a batch. The auxiliary training term is described in the text.

C.3 Data and Configuration

All AceCode subsets below are reserved separately by problem ID; SAE fitting, analysis, and PUSH guidance use problems outside the 33,500 RL-training set.

Table 16: Data roles and statistical units.

Role	Data and computation
RL training	33,500 problems from the AceCode corpus; responses are generated and rewarded.
SAE training	A separately reserved AceCode subset; only text and hidden states from the frozen pre-RL model are read, without task-outcome labels.
Validation split	800 problems for subsequent component and strength analyses.
Test split	800 problems for evaluating fixed configurations, method comparisons, continuation, rollout savings, and capability ablations.
SAE reconstruction	450 separately reserved AceCode problems with saved GRPO answers; reconstruction uses prompt and answer tokens (Appendix A.1). The 450 GRPO texts used for Top5 shared-trend and matched controls use answer tokens only (Appendix A.3.1).
PCA and t-SNE trajectories	A separately reserved set of 128 fixed AceCode problems and saved GRPO answers across checkpoints and all six runs; also used for pre/post-PUSH representation comparisons (Appendices A.2 and E.1).
Feature discovery and confirmation	1,000 separately reserved AceCode problems with fixed DAPO answers, split 500/500; used for Top5 ranking and trajectories, four feature classes, and direction analyses (Appendix A.5.1).
Independent-code analysis	48 Python 3.12 standard-library functions, split 24/24, for Top5 shared activation and code preferences; paired-edit comparisons use eight program pairs per reported contrast (Appendix A.3.3).
Code-context analysis	128 MBPP+ problems with saved model answers, split 64/64, for Rising/Falling feature interpretation and problem coverage; distinct from the 128-problem trajectory set (Appendix A.6.1).
PUSH fitting	A separately reserved fixed guidance subset; selection is specified below, and batching in Appendix B.3.
External capability evaluation	378 MBPP problems; responses are regenerated and scored separately by the MBPP and MBPP+ test suites.

The AceCode subsets in Table 16 are reserved separately by problem ID.

Fixed texts for representation analysis. The separately reserved reconstruction set contains 185 problems from bigcode_python_fns, 145 from evol, and 120 from oss, with 174,889 non-padding prompt and answer tokens. The separate shared-trend and matched-control analyses use 98,591 answer tokens from the fixed GRPO texts of Experiment 36. The separately reserved trajectory set contains 57, 37, and 34 problems from these sources, respectively, with 27,299 answer tokens. The feature discovery and confirmation data contain 395, 332, and 273 problems from these sources. A fixed hash selects one deduplicated nonempty DAPO response per problem. Balanced splits by source and response length contain 156,863 and 157,264 answer tokens, respectively (314,127 in total).

Construction of the test split. We construct the test split from the 53,649 AceCode problems outside the 33,500-problem RL training set, excluding the earlier 500-problem evaluation set and 20 preflight problems. Candidates undergo checks for valid test structure, duplicate or highly overlapping problem statements, and duplicate test suites. AI-agent-assisted review then checks the consistency of problem statements and tests, required context, and the validity of expected outputs. Every first-review pass receives an independent second review, with disagreements adjudicated before selection. After review and duplicate screening, 1,080 candidates remain. We select 800 to match the training set's source proportions and approximate its task types, test structure, topics, and historical reference-model difficulty: 321 problems from bigcode_python_fns, 253 from evol, and 226 from oss. Historical reference-model scores are used only for this matching after quality-review decisions are fixed; scores from the evaluated RL checkpoints are not used to select problems. The resulting test split is held out from RL training, PUSH fitting, and configuration selection.

Construction of the validation split. After fixing the test split, we select the validation split for subsequent component and strength analyses from the 87,149-problem AceCode corpus, excluding the 33,500 RL training problems and every test-split problem. We remove duplicates and highly text-similar matches to training problems and enforce the problem-length limits. Sampling allocates counts according to the training set's source proportions and stratifies by historical reference-model difficulty. The resulting 800 problems comprise 301 from bigcode_python_fns, 258 from evol, and 241 from oss.

Selection of the 64 code guidance problems. From a separately reserved pool of 2,000 AceCode problems with saved GRPO responses, excluding earlier discovery and pilot problems leaves 803 candidates. We sort them by the SHA-256 hash of tokenwise-feature-budget-v1, a NUL separator, and the question ID, then retain 128. The first 64 form the code PUSH guidance set; the other 64 were held out in the original feature-budget experiment. Selection uses no correctness, reward, or test-pass labels. Prompts, responses, and token positions are fixed across model comparisons.

Fixed method evaluation and subsequent policy exploration. Method evaluation uses the fixed A:B ratio from Equation (5), with $s = 1$ for test-set immediate comparisons and all Single PUSH and Multiple PUSH continuations. Comparisons across different strengths use the validation split and are detailed in Appendix E.2. MBPP/MBPP+ are held out from RL rewards, PUSH fitting, and configuration development.

Statistical reporting. Table 1 reports the mean and sample standard deviation of checkpoint gains across three RL training runs per algorithm. RL training seeds, PUSH optimization seeds, and resampling seeds are distinct. Checkpoints, settings, and continuation branches from one trajectory count as one independent training run. Paired bootstrap uses identical problem indices for both conditions and estimates their performance difference for the given model and problem distribution. Each experiment specifies resampling count, stratification, seed, point estimates, and intervals. Curve IQRs cover the central 50% across problems or features, as specified in each caption.

C.4 Mathematical Training and Evaluation

Each mathematical model contributes one GRPO run shared by its baseline and all continuation branches, spanning steps 0–512 for Qwen and 0–1050 for DeepSeekMath. Table 17 lists model-specific settings. Each 8×-width K192 SAE is trained once before RL on that model's GRPO training problems, using only frozen-model text and hidden states without reward or correctness labels. Mathematical PUSH uses $s = 1$, with batching specified in Appendix B.3.

Table 17: Model-specific configurations for mathematical RLVR and PUSH.

Item	Qwen2.5-Math-7B Base	DeepSeekMath-Instruct 7B
RL training seed	1315091043	1315091043
RL training steps	512	1050
Model architecture	28 blocks; hidden size 3,584	30 blocks; hidden size 4,096
SAE candidate layers / dictionary width	H28/H26/H21; 28,672	H30/H28/H23; 32,768
Main-text PUSH target layer	H28	H30
GRPO learning rate	1×10^{-5}	1×10^{-5}
Temporary PUSH A/B branches	Temporary LoRA	Temporary LoRA

Qwen samples 128 problems with eight responses each per step; DeepSeekMath samples 64 with sixteen. Both generate 1,024 rollouts per step. With split seed 20260910, Qwen's 7,473 GSM8K training problems are divided into 6,973 GRPO-training problems and 500 held-out evaluation problems. The latter form the GSM8K test split and are scored greedily after settings are fixed. DeepSeekMath is evaluated on all 1,319 official GSM8K test problems, with accuracy equal to the correct count divided by 1,319. Pre- and post-PUSH scores use the same evaluation problems.

D Main Experimental Results

This appendix expands the main experiments from immediate capability gains to method comparisons, continued training, and computational efficiency. Complete starting-point results connect the short PUSH update to sustained gains and reduced RL sampling.

D.1 Immediate Capability Gains

Coding: three independent training runs. For each starting checkpoint, we subtract the accuracy of that run's original model from its post-PUSH accuracy and then compute the mean and sample standard deviation across Seeds 1–3. Figure 13 shows the complete three-run comparisons under the fixed PUSH setting; Table 18 lists the individual gains.

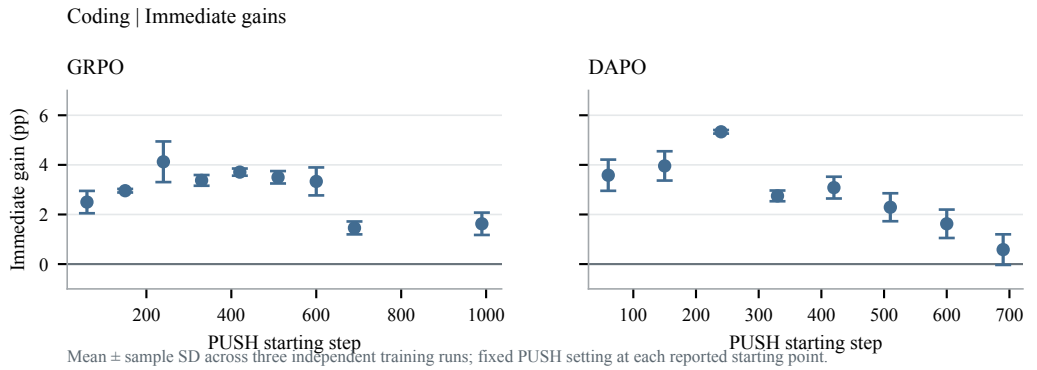


Figure 13: Immediate coding gains at the starting points with three-seed evaluations. Points and error bars show the mean and sample standard deviation of within-run gains; units are percentage points.

Mathematical reasoning. Mathematical Single PUSH uses $s = 1$ at 32 Qwen and 25 DeepSeekMath starting points, with immediate reevaluation on the same problems. Test

Table 18: Individual immediate coding gains (percentage points). Mean and SD summarize three-seed evaluations; a dash denotes an unavailable entry.

	Step	Seed 1	Seed 2	Seed 3	Mean ± SD
GRPO	60	2.38	3.00	2.13	$2.50_{\pm 0.45}$
	150	3.00	3.00	2.88	$2.96_{\pm 0.07}$
	210	6.13	5.75	6.00	$5.96_{\pm 0.19}$
	240	4.00	5.00	3.38	$4.13_{\pm 0.82}$
	270	3.88	2.25	2.63	$2.92_{\pm 0.85}$
	330	3.50	3.50	3.13	$3.38_{\pm 0.22}$
	420	3.63	3.63	3.88	$3.71_{\pm 0.14}$
	510	3.75	3.50	3.25	$3.50_{\pm 0.25}$
	570	3.13	2.88	2.88	$2.96_{\pm 0.14}$
	600	3.38	2.75	3.88	$3.33_{\pm 0.56}$
	690	1.25	1.75	1.38	$1.46_{\pm 0.26}$
	810	-0.88	-1.50	-1.25	$-1.21_{\pm 0.31}$
	990	2.00	1.75	1.12	$1.62_{\pm 0.45}$
DAPO	60	3.00	3.50	4.25	$3.58_{\pm 0.63}$
	150	3.75	4.63	3.50	$3.96_{\pm 0.59}$
	210	3.00	4.63	4.50	$4.04_{\pm 0.90}$
	240	5.25	5.38	5.38	$5.33_{\pm 0.07}$
	270	3.13	4.00	3.63	$3.58_{\pm 0.44}$
	330	2.50	2.88	2.88	$2.75_{\pm 0.22}$
	420	3.50	2.63	3.13	$3.08_{\pm 0.44}$
	510	1.75	2.25	2.88	$2.29_{\pm 0.56}$
	600	1.00	2.13	1.75	$1.62_{\pm 0.57}$
	690	0.88	-0.13	1.00	$0.58_{\pm 0.62}$
	810	-3.00	-3.38	-3.75	$-3.38_{\pm 0.38}$

answers are excluded from training and PUSH supervision. For example, Qwen improves from 77.2% to 82.2% at step 32 and DeepSeekMath from 86.0% to 87.41% at step 530. For each positive gain, the first later baseline checkpoint reaching the post-PUSH accuracy defines the advancement in training steps; multiplying by 1,024 gives saved rollouts. Only observed checkpoints enter this calculation.

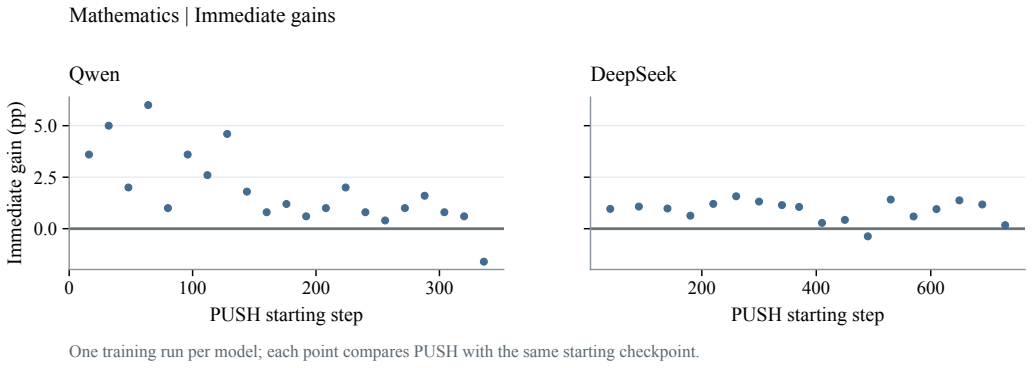


Figure 14: Immediate gains on mathematical reasoning, with one training run per model.

D.1.1 External Benchmarks

Across Starting Points Figures 15 and 16 compare immediate test-split gains at ten starting points through step 690 with mean MBPP/MBPP+ gains across the included coefficient settings. The scans below report the external-benchmark settings separately; only measured points within each figure's displayed range are shown.

DAPO Scan Figure 17 compares six predefined coefficient settings with the baseline on MBPP and MBPP+ at the 22 DAPO checkpoints through step 700; the underlying data cover 34 checkpoints. The six settings are $(4, 0)$, $(0, 4)$, $(1, 1)$, $(2, 2)$, $(4, 4)$, and $(6, 6)$.

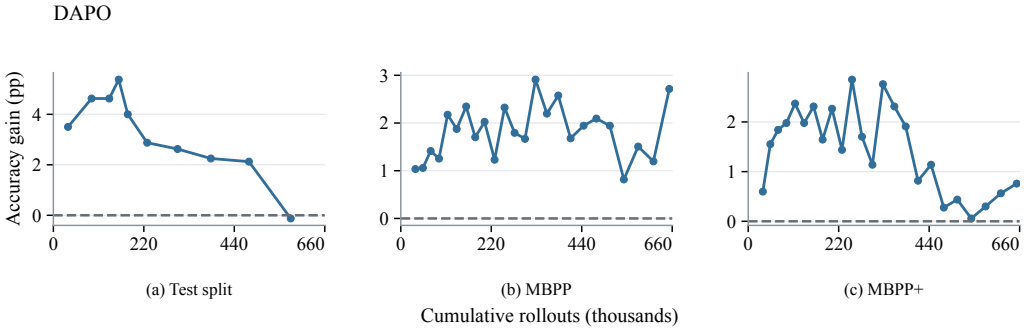


Figure 15: DAPO accuracy gains against cumulative rollouts over 0–660,000 rollouts. (A) Immediate gains at strength $s = 1$ on the test split. (B,C) Mean gains on MBPP and MBPP+ across the included coefficient settings. The external benchmarks include 24 measured checkpoints through step 750.

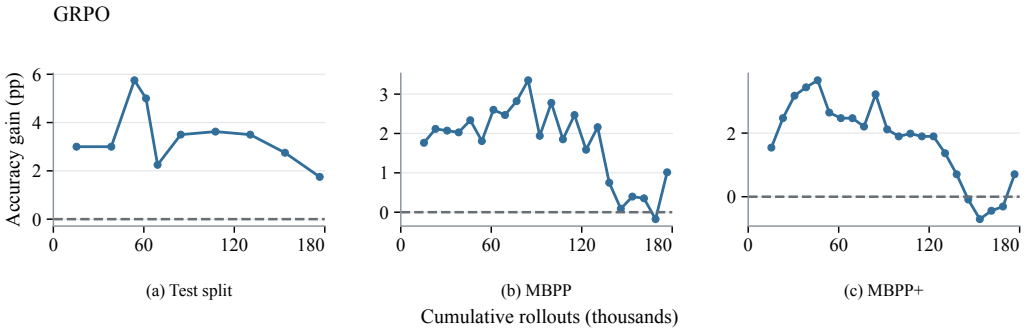


Figure 16: GRPO accuracy gains against cumulative rollouts over 0–180,000 rollouts. (A) Immediate gains at strength $s = 1$ on the test split through step 690. (B,C) Mean gains on MBPP and MBPP+ across the included coefficient settings, at 22 measured checkpoints through step 690.

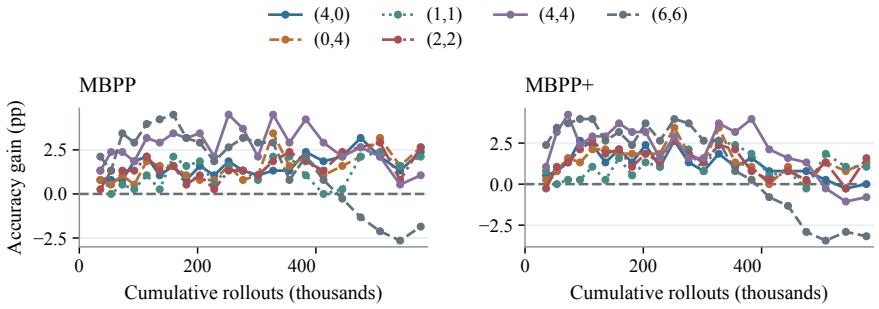


Figure 17: MBPP and MBPP+ accuracy gains in percentage points under fixed DAPO coefficient settings. The rollout axis is restricted to steps 0–700 (0–588,736 rollouts), with 22 measured checkpoints from step 60 to step 690.

GRPO Scan The GRPO scan compares seven predefined settings with baseline at 34 checkpoints under the default PUSH settings and 378-problem MBPP/MBPP+ protocol (Appendix C). Figure 18 shows all 22 measured checkpoints through step 700, including step 450.

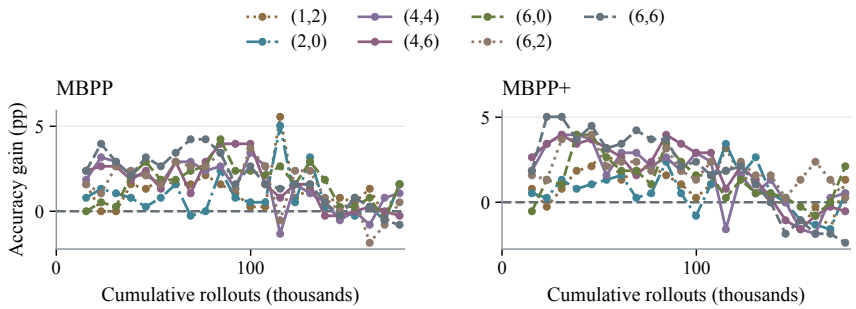


Figure 18: MBPP and MBPP+ accuracy gains in percentage points under fixed GRPO coefficient settings, plotted against cumulative rollouts over steps 0–700 (0–179,200 rollouts). Both panels show all 22 measured checkpoints in this range, including step 450.

D.2 Comparison with Baseline Methods

D.2.1 Shared Conditions and Method-Specific Updates

All methods in Table 1 start from the same checkpoint. Fitted methods share the guidance data and optimization defaults in Appendices C.3 and B.3; Table 19 specifies their targets and update procedures.

Table 19: Targets and update procedures of the compared methods.

Method	Target and update procedure
SFT	Computes next-token cross-entropy on the fixed responses while masking prompt and padding positions; performs 16 consecutive updates and directly uses the resulting weights.
ADL	Fits $2h_{\text{now}} - h_{\text{pre}}$ for 16 consecutive updates and directly uses the resulting weights.
RepE	Fits 96-dimensional PCA to historical H28 differences on the same guidance problems, uses reconstructed differences as extrapolation targets, and performs 16 consecutive updates.
ExPO	Directly forms effective weights $2W_{\text{now}} - W_{\text{pre}}$ with extrapolation coefficient 1; requires neither gradient updates nor guidance-text fitting.
PUSH	Separate A/B fitting and fixed-ratio merge, following Appendix B.3.

ADL and SFT traverse the same problem order for two consecutive passes with eight problems per batch; the SFT loss is mean cross-entropy over answer tokens in each batch.

RepE. RepE fits PCA only on answer tokens from the guidance set: per-token pre-to-current H28 changes are centered, projected onto 96 principal components, shifted back by the mean change, and then added to the current activation as the fixed target. The PCA directions, mean, and targets are frozen before optimization, which uses the ADL loss normalization and extrapolation strength 1. Test problems and later checkpoints do not participate in PCA fitting. Baseline and PUSH results are reused from the existing evaluations. All methods run inference after LoRA merging, with an eight-second judging limit per problem; evaluation otherwise follows Table 15.

At the additional step 210, the gains for PUSH, ADL, SFT, and RepE are 5.75, 1.88, 2.13, and 2.63 percentage points under GRPO, and 4.63, 0.50, 1.50, and 1.38 percentage points under DAPO.

ExPO. The ExPO results at two additional steps, 210 and 270, are +1.75 and +0.88 percentage points under GRPO, and +1.88 and +0.13 under DAPO; the corresponding PUSH gains are +5.75 and +2.25 under GRPO, and +4.63 and +4.00 under DAPO. These additional single-seed comparisons use GRPO Seed 2 and DAPO Seed 2 from Table 15, with the existing baseline and PUSH results at each starting point.

D.2.2 Displacement-Matched ExPO

To control for different update magnitudes between ExPO and PUSH, we construct

$$W_{\text{match}} = W_{\text{now}} + \alpha\,(W_{\text{now}} - W_{\text{pre}})\,, \qquad \alpha = \frac{\|W_{\text{PUSH}} - W_{\text{now}}\|_F}{\|W_{\text{now}} - W_{\text{pre}}\|_F}. \tag{13}$$

Thus, matched ExPO has the same Frobenius norm of effective weight displacement as PUSH at strength $s = 1$ at the same starting point, while retaining the direction of the full RL weight difference. Both methods start from the same checkpoint at each of six training steps: 150, 210, 270, 420, 600, and 690. The test split, judging program, and generation settings are identical across the compared methods.

Table 20: Displacement-matched ExPO versus PUSH at strength $s = 1$. Gains on the test split are in percentage points relative to the same starting-point baseline and are reported as means over Seeds 1–3 with standard deviations as subscripts; α is the matching coefficient, and PUSH entries at shared checkpoints match Table 1.

	Step	α	Matched ExPO	PUSH
GRPO	150	1.033	$1.25_{\pm 0.13}$	$2.96_{\pm 0.07}$
	210	0.827	$1.75_{\pm 0.33}$	$5.96_{\pm 0.19}$
	270	0.691	$1.88_{\pm 0.63}$	$2.92_{\pm 0.85}$
	420	0.511	$4.25_{\pm 0.50}$	$3.71_{\pm 0.14}$
	600	0.426	$-2.63_{\pm 0.22}$	$3.33_{\pm 0.56}$
	690	0.400	$0.88_{\pm 0.45}$	$1.46_{\pm 0.26}$
DAPO	150	0.976	$2.25_{\pm 0.45}$	$3.96_{\pm 0.59}$
	210	0.755	$2.75_{\pm 0.13}$	$4.04_{\pm 0.90}$
	270	0.635	$2.25_{\pm 0.43}$	$3.58_{\pm 0.44}$
	420	0.502	$-0.88_{\pm 0.38}$	$3.08_{\pm 0.44}$
	600	0.381	$-0.50_{\pm 0.50}$	$1.63_{\pm 0.57}$
	690	0.341	$-0.13_{\pm 0.25}$	$0.58_{\pm 0.62}$

PUSH exceeds matched ExPO at 11 of the 12 starting points, and matched ExPO is higher only at GRPO step 420.

D.3 Continued RL with Single and Multiple PUSH

Continuation setup and reading the figures. Every intervention uses $s = 1$, after which RL resumes under the continuation protocol in Appendix B.3. Coding evaluations occur every 30 steps through step 1020. The collection contains 118 branches: 70 Single PUSH and 48 Multiple PUSH, including all three runs of each coding algorithm and one run of each mathematical model. Figures report the accuracy difference from the same run's original RL baseline at the same training step. Coding lines and bands show the mean and sample standard deviation across three runs; mathematical curves show their individual runs. Arrows separate the immediate intervention from subsequent training. Original RL scores are collected once in Figure 19.

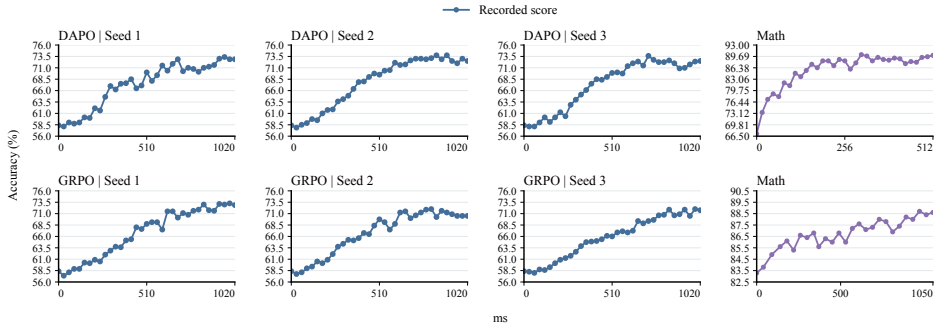
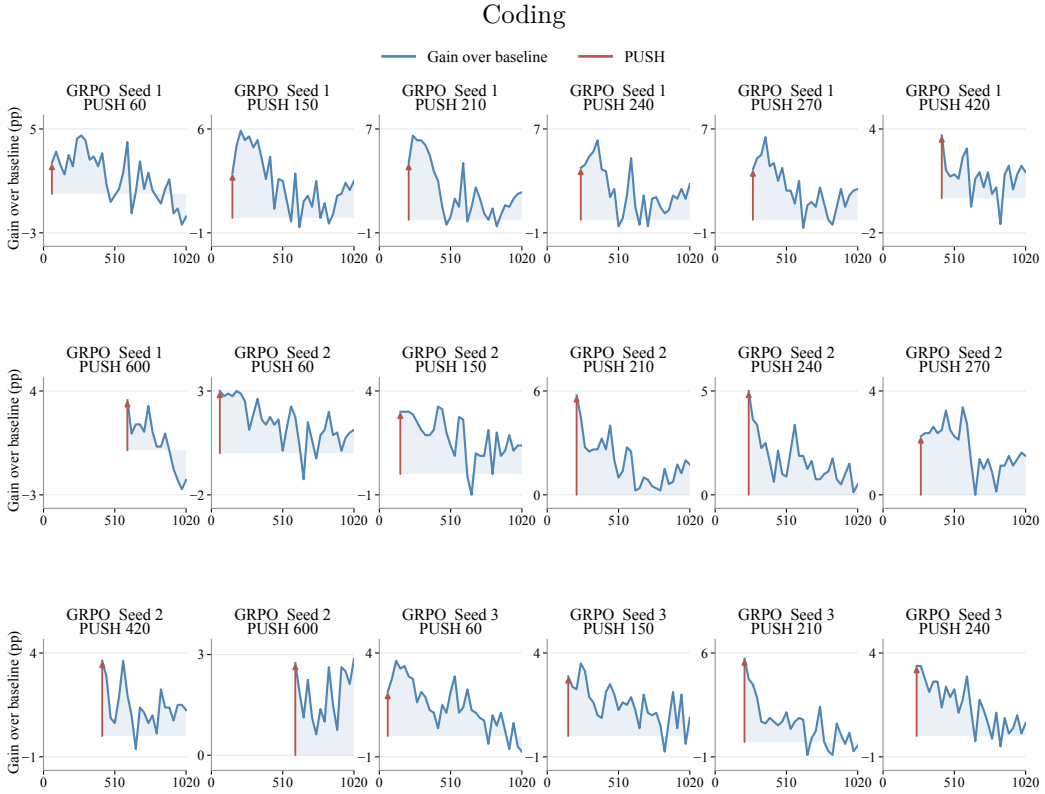
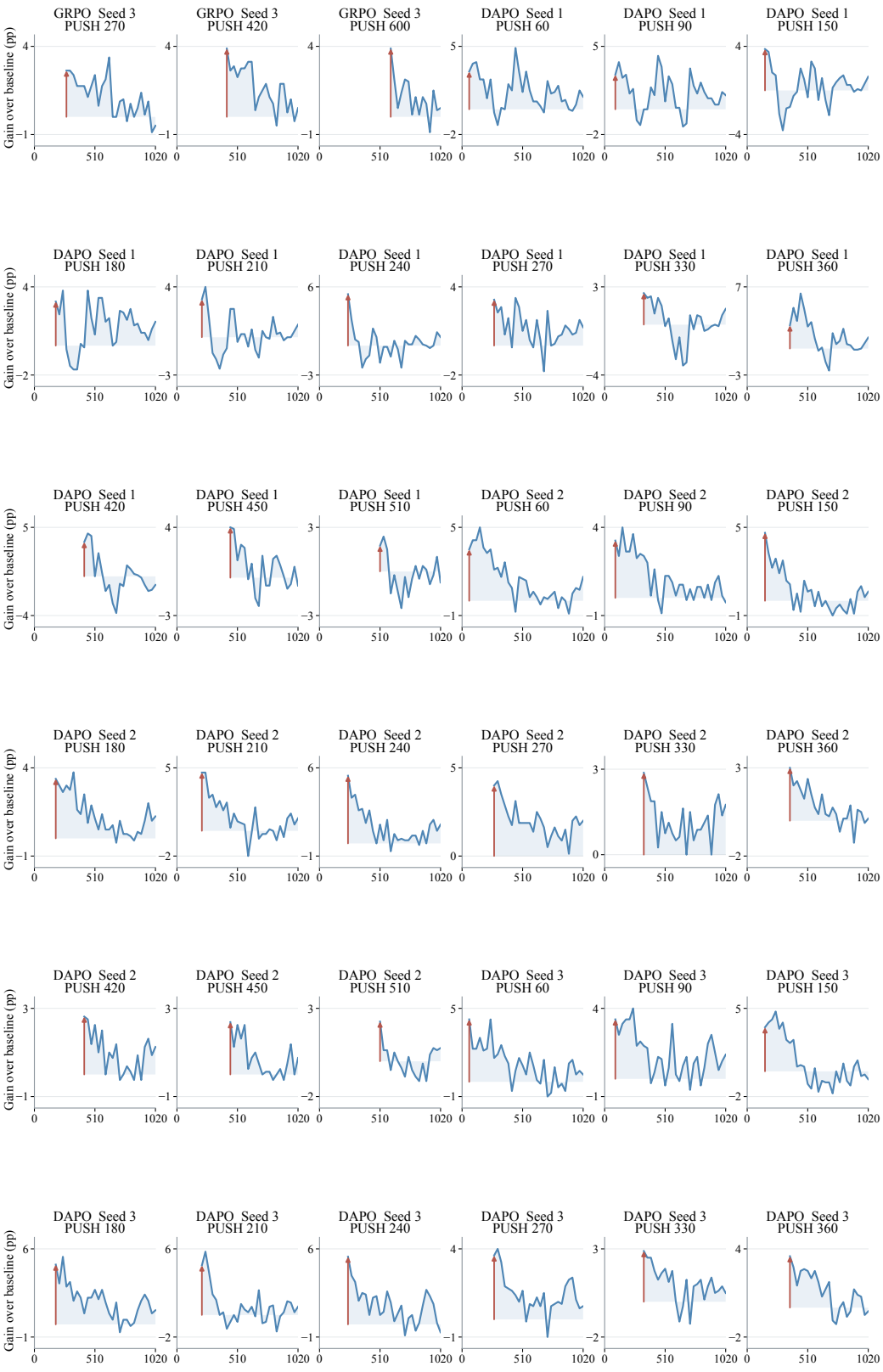


Figure 19: Baseline test scores for all three DAPO and GRPO training seeds and the single training run of each mathematical model. Each coding curve contains all 35 checkpoints from step 0 through 1020 at intervals of 30. The mathematical curves use their recorded evaluation checkpoints through step512 for Qwen and step1050 for DeepSeek. Coding step0 uses the shared pre-RL base-model score.

D.3.1 Single PUSH: Persistence of the Initial Gain

GRPO contributes seven starting points per seed and DAPO contributes twelve. Each panel follows one starting point through the full recorded continuation, making the duration of the initial advantage directly visible. Qwen and DeepSeekMath contribute seven and six Single PUSH branches, respectively.




GRPO Seed 3 PUSH 270
GRPO Seed 3 PUSH 420
GRPO Seed 3 PUSH 600
DAPO Seed 1 PUSH 60
DAPO Seed 1 PUSH 90
DAPO Seed 1 PUSH 150
DAPO Seed 1 PUSH 180
DAPO Seed 1 PUSH 210
DAPO Seed 1 PUSH 240
DAPO Seed 1 PUSH 270
DAPO Seed 1 PUSH 330
DAPO Seed 1 PUSH 360
DAPO Seed 1 PUSH 420
DAPO Seed 1 PUSH 450
DAPO Seed 1 PUSH 510
DAPO Seed 2 PUSH 60
DAPO Seed 2 PUSH 90
DAPO Seed 2 PUSH 150
DAPO Seed 2 PUSH 180
DAPO Seed 2 PUSH 210
DAPO Seed 2 PUSH 240
DAPO Seed 2 PUSH 270
DAPO Seed 2 PUSH 330
DAPO Seed 2 PUSH 360
DAPO Seed 2 PUSH 420
DAPO Seed 2 PUSH 450
DAPO Seed 2 PUSH 510
DAPO Seed 3 PUSH 60
DAPO Seed 3 PUSH 90
DAPO Seed 3 PUSH 150
DAPO Seed 3 PUSH 180
DAPO Seed 3 PUSH 210
DAPO Seed 3 PUSH 240
DAPO Seed 3 PUSH 270
DAPO Seed 3 PUSH 330
DAPO Seed 3 PUSH 360
Gain over baseline (pp)
0
510
1020

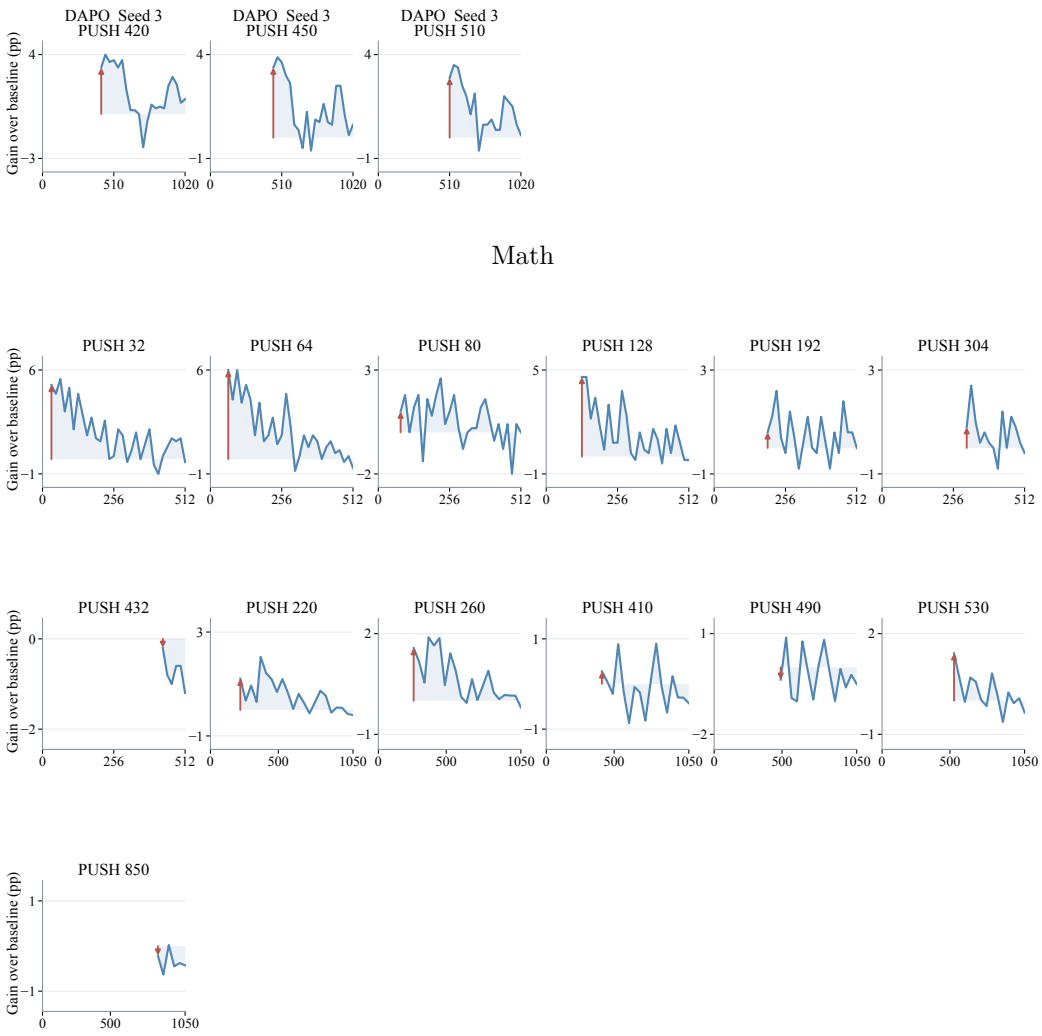


Figure 20: Continuation of RL after Single PUSH. Each row contains six panels. Coding panels show the gain above the same-step baseline; Math panels follow below.

D.3.2 Multiple PUSH: Additional Gains from Repeated Intervention

Multiple PUSH uses six schedules for each coding seed, seven for Qwen, and five for DeepSeekMath. Intervention schedules are fixed independently of development- or test-set scores. Each coding panel pairs a Multiple PUSH branch with the Single PUSH branch starting at the same checkpoint. Their common first intervention makes the subsequent benefit of additional PUSH applications directly visible. The panel headings give the full schedules.


Coding
Gain over baseline
PUSH
GRPO Seed 1 PUSH 60/180/570
GRPO Seed 1 PUSH 150/360/720
GRPO Seed 1 PUSH 210/570
GRPO Seed 1 PUSH 240/420
GRPO Seed 1 PUSH 270/600/870
GRPO Seed 1 PUSH 600/720/840
GRPO Seed 2 PUSH 60/180/570
GRPO Seed 2 PUSH 150/360/720
GRPO Seed 2 PUSH 210/570
GRPO Seed 2 PUSH 240/420
GRPO Seed 2 PUSH 270/600/870
GRPO Seed 2 PUSH 600/720/840
GRPO Seed 3 PUSH 60/180/570
GRPO Seed 3 PUSH 150/360/720
GRPO Seed 3 PUSH 210/570
GRPO Seed 3 PUSH 240/420
GRPO Seed 3 PUSH 270/600/870
GRPO Seed 3 PUSH 600/720/840
DAPO Seed 1 PUSH 60/180/390
DAPO Seed 1 PUSH 90/420
DAPO Seed 1 PUSH 180/240/420/660
DAPO Seed 1 PUSH 270/420/690
DAPO Seed 1 PUSH 360/450/810
DAPO Seed 1 PUSH 450/630/870
DAPO Seed 2 PUSH 60/180/390
DAPO Seed 2 PUSH 90/420
DAPO Seed 2 PUSH 180/240/420/660
DAPO Seed 2 PUSH 270/420/690
DAPO Seed 2 PUSH 360/450/810
DAPO Seed 2 PUSH 450/630/870
DAPO Seed 3 PUSH 60/180/390
DAPO Seed 3 PUSH 90/420
DAPO Seed 3 PUSH 180/240/420/660
DAPO Seed 3 PUSH 270/420/690
DAPO Seed 3 PUSH 360/450/810
DAPO Seed 3 PUSH 450/630/870
Gain over baseline (pp)
0
510
1020
Math

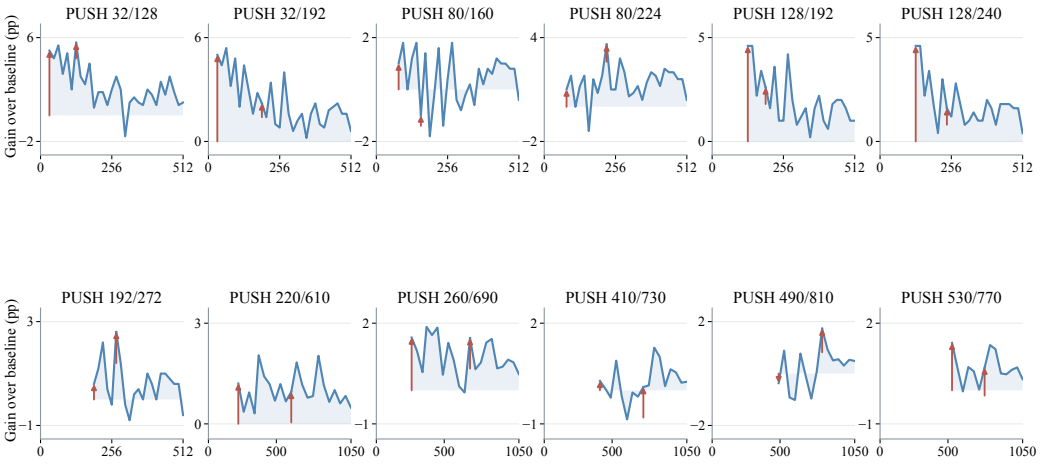


Figure 21: Continuation of RL after Multiple PUSH. Each row contains six panels. Coding panels show the gain above the same-step baseline; Math panels follow below.

Individual-run matched-budget examples. For GRPO Seed 2, six pairs of Single and Multiple PUSH branches provide individual-run comparisons at matched cumulative rollout counts. All six pairs are compared at step 1020, corresponding to 261,120 rollouts. Multiple PUSH improves over Single PUSH in three pairs, ties in one, and declines in two. For the PUSH sequences 60/180/570, 150/360/720, 210/570, 240/420, 270/600/870, and 600/720/840, the respective differences are +0.88, +1.13, 0.00, +1.50, −2.00, and −1.13 percentage points.

DAPO Branch DAPO includes a branch with the first PUSH at step 90 and the second at step 420: the second PUSH raises accuracy from 69.625% to 71.250%; subsequent evaluations at steps 450 and 480 attain 71.000% and 70.750%, respectively.

D.4 Training Efficiency

D.4.1 Rollout Savings

For a target accuracy q, let $R_{\mathrm{RL}}(q)$ and $R_{\mathrm{PUSH}}(q)$ be the cumulative rollout counts at the first observed baseline and PUSH-branch checkpoints reaching q. Let R_0 be the cumulative count immediately before the first PUSH in that branch, for both Single and Multiple PUSH. We report two reductions:

$$\mathrm{Saving}_{\mathrm{post}}(q) = \frac{R_{\mathrm{RL}}(q) - R_{\mathrm{PUSH}}(q)}{R_{\mathrm{RL}}(q) - R_0}, \tag{14}$$

$$\mathrm{Saving}_{\mathrm{total}}(q) = \frac{R_{\mathrm{RL}}(q) - R_{\mathrm{PUSH}}(q)}{R_{\mathrm{RL}}(q)}. \tag{15}$$

Post-PUSH reduction measures the fraction of subsequent RL sampling avoided after the intervention, while total reduction measures savings relative to the complete baseline training budget at that accuracy. Absolute savings are $R_{\mathrm{RL}}(q) - R_{\mathrm{PUSH}}(q)$ under both definitions.

Table 2 uses Seeds 1–3 separately for each algorithm. Within each seed, Single and Multiple PUSH share six starting points: 60/150/210/240/270/600 for GRPO and 60/90/180/270/360/450 for DAPO. For each branch and its own baseline, we take cumulative maximum accuracy and thresholds spaced 0.025 percentage points from the branch's immediate post-PUSH accuracy to the smaller of the two observed peak accuracies through step 1020. At each threshold, we use the first observed checkpoint reaching it, without interpolation, and compute both reductions before averaging. All evaluated thresholds have a

positive remaining baseline budget $R_{\mathrm{RL}}(q) - R_0$. We average thresholds within each branch, then the six starting points within each seed, assigning equal weights at each stage. The reported mean and sample standard deviation are computed from the three unrounded seed-level averages; the standard deviation uses divisor $3 - 1 = 2$. Table 21 reports each training seed separately.

Table 21: Rollout savings for each RL training seed on the test split. Each row averages target accuracies within a branch and then the six shared starting points.

	Seed	PUSH	Rollouts saved (k)	Post-PUSH reduction (%)	Total reduction (%)
GRPO	1	Single	39.05	51.76	27.82
	1	Multiple	52.59	61.53	35.53
	2	Single	28.20	42.74	20.45
	2	Multiple	34.29	48.78	24.84
	3	Single	26.93	33.21	18.59
	3	Multiple	41.88	45.26	26.90
DAPO	1	Single	49.61	22.70	12.30
	1	Multiple	135.10	51.05	29.79
	2	Single	89.16	37.36	22.14
	2	Multiple	133.88	49.54	30.17
	3	Single	96.41	41.15	22.49
	3	Multiple	153.37	58.23	34.86
Qwen		Single	53.13	27.10	22.92
		Multiple	43.28	34.21	24.35
DeepSeekMath		Single	8.65	35.19	1.54
		Multiple	153.78	23.05	24.87

D.4.2 Matched Budget

The evaluated GRPO checkpoints immediately before the two thresholds have accuracies 65.625% and 71.250%, corresponding to 99,840 and 161,280 rollouts. The first checkpoints reaching the thresholds have accuracies 66.625% and 71.500%, corresponding to 107,520 and 168,960 rollouts.

Rows in Table 22 are independent comparisons at the stated strengths.

Table 22: Generation budget at matched performance on the test split.

Algorithm	Strength s	Threshold	PUSH	Baseline	Saved	Saving
GRPO	$s = 1$	66.625	53,760	107,520	53,760	50.0%
	$s = 1$	71.500	153,600	168,960	15,360	9.1%
DAPO	$s = 1$	67.250	158,976	276,704	117,728	42.5%
	$s = 1$	67.625	180,992	276,704	95,712	34.6%

Thresholds are accuracy percentages.

D.4.3 PUSH Cost

Table 23 reports timing under the common guidance and optimization settings in Appendix B.3.

Each run loads the model and SAE, reads the fixed guidance sequences and cached pre-RL activations, extracts current-model representations, constructs targets, and fits A/B in parallel. We write the combined PUSH update directly into the resident model, restore the saved optimizer and training state for subsequent RL updates, and save the A/B artifacts.

We measure each end-to-end total directly with a single wall clock spanning process startup, computation, synchronization, saving, and cleanup. A/B model loading overlaps cache reading and target preparation; parallel fitting and saving is timed through completion of both branches.

Table 23: End-to-end PUSH timing on one NVIDIA H200 (seconds, three runs per model), from process startup to readiness for RL continuation. Stages report means; totals also report sample standard deviation and range.

Stage	Code	Qwen Math	DeepSeekMath
Model, tokenizer, and SAE loading	4.36	4.07	4.23
Fixed guidance and cache reading	0.12	0.65	0.14
Current representations and targets	3.14	7.69	4.34
A/B model loading (overlapping)	4.45	4.01	4.20
Parallel A/B fitting and saving	16.96	43.32	25.11
Weight write-back	0.47	0.34	0.35
Preparation for subsequent RL updates	0.01	0.01	0.01
Remaining artifact saving	1.87	1.29	1.02
End-to-end total	52.67	77.07	58.88
Sample standard deviation	0.29	0.40	0.56
Range	[52.35, 52.92]	[76.62, 77.38]	[58.55, 59.53]

E Supplementary Discussion and Analyses

E.1 Joint Representation and Capability Progress

For one RL training run per algorithm, we apply PUSH at $s = 1$ to twelve starting points and measure post-PUSH test accuracy. Both algorithms use Seed 2. Representation comparisons use the trajectory texts in Appendix C.3, averaging Raw H28 states and SAE activations separately over answer tokens within each problem. For each representation view, we compare the post-PUSH mean with 32 baseline states sampled every 30 steps from 60 through 990. Each problem's Euclidean distances are normalized by its baseline distance from step 60 to step 990, then averaged equally across problems. The minimum identifies the nearest baseline checkpoint. The main text reports the direct measurements; the table below gives the nearest-checkpoint summary. Accuracy uses the first recorded baseline checkpoint reaching the post-PUSH test score in the same reference window; "Over max" means the score exceeds the baseline maximum.

Table 24: Nearest baseline checkpoints after PUSH. Raw hidden and SAE report the nearest representation state; Accuracy reports the first checkpoint reaching the post-PUSH test score. ">max" means the post-PUSH score exceeds the baseline maximum within steps 60–990. Both algorithms use Seed 2.

	PUSH step	60	150	210	240	270	330	420	510	600	690	810	990
GRPO	Raw hidden	120	270	330	390	480	600	690	720	930	990	990	990
	SAE	120	270	360	420	480	600	690	720	930	990	990	990
	Accuracy	240	270	420	480	420	510	630	>max	660	780	630	>max
DAPO	Raw hidden	120	240	360	420	450	510	540	690	810	870	870	870
	SAE	150	240	360	420	450	510	540	690	810	870	870	960
	Accuracy	270	300	360	390	390	390	600	600	>max	690	480	390

E.2 Training Stage and PUSH Strength

Figure 6 in Section 6.4 shows the GRPO validation scan: twelve checkpoints and ten coefficient settings. Coefficients (α, β) weight suppression (A) and continuation (B) in Equation (5). Both algorithms use $(1,0), (2,0), (4,0), (0,2), (1,1), (2,1), (2,2), (4,2), (6,2), (6,4)$. Both branch increments are fitted once at each checkpoint; changing the merge coefficients requires no further fitting. Each cell gives the gain over the same-step baseline. Cross-algorithm stage comparisons use each algorithm's recorded cumulative rollout count, since equal step indices can require different generation counts.

Training stage and update strength. Doubling s doubles parameter displacement: at 53,760 rollouts, the gain rises from 3.13 to 7.38 points; at 153,600, it falls from 4.13 to 2.63. Five of ten settings improve accuracy at 207,360 GRPO rollouts, and nine at 253,440. No DAPO

setting improves accuracy at 732,608 or 983,456 rollouts. These validation measurements characterize sensitivity to training stage and update strength. Test-set PUSH evaluations use $s = 1$.

Exploratory stage-wise policy. We explore a manually specified policy on the validation split: stronger PUSH early ($s = 2$), weaker PUSH in the middle ($s = 1$), and no PUSH late in training. The suppression-to-continuation coefficient ratio remains $2 : 1$, giving coefficients $(4, 2)$ and $(2, 1)$ for the two strengths. Table 25 defines the stage assignments, and Figure 22 shows the corresponding validation gains. Each starting point is evaluated independently; these evaluations do not form a sequential continuation run. Test-set evaluations use $s = 1$ or no PUSH.

Table 25: Manually specified stage-wise PUSH policy explored on the validation split. Stage assignments refer to the listed starting checkpoints.

	Stage	Strength s	Starting steps
DAPO	Early	$s = 2$	60, 150, 210, 240
	Middle	$s = 1$	270, 330, 420, 510, 600, 690
	Late	No PUSH	810, 990
GRPO	Early	$s = 2$	60, 150, 210
	Middle	$s = 1$	240, 270, 330, 420, 510
	Late	No PUSH	600, 690, 810, 990

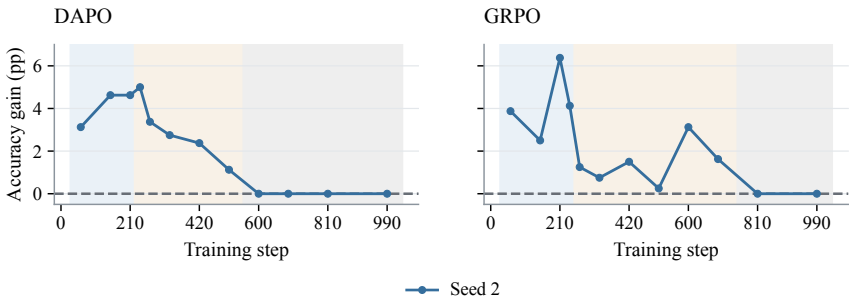


Figure 22: Validation gains under the manually specified stage-wise Single PUSH policy. Both algorithms use Seed 2. Blue, beige, and gray backgrounds indicate early-stage PUSH at $s = 2$, middle-stage PUSH at $s = 1$, and late-stage no PUSH, respectively.

For each algorithm, the plotted values are taken from its validation scan using the stage assignments in Table 25; every point starts independently from its corresponding baseline checkpoint. The points without PUSH coincide with the baseline.

E.3 Supplementary Ablations

Section 6.3 reports the core component ablations. This appendix specifies their settings and reports sensitivity to target layer, guidance data, and feature budget using the same seven starting steps and three-seed summary as the main table.

Main-text comparison details. The component comparisons are reported in Table 4. Each seed starts independently from its corresponding GRPO checkpoint; Table 4 reports the mean and sample standard deviation across the three seeds. All conditions use the guidance data and optimization settings in Appendices C.3 and B.3.

Historical targets. The w/o historical alignment condition permutes complete pre-RL SAE vectors across guidance answer tokens while keeping current activations fixed; differences, D96, and A/B targets are then recomputed. The main table reports the complete three-seed comparison. The w/o token-level directions condition replaces position-specific historical directions with their mean over guidance answer tokens; evaluation problems do not participate in constructing this mean.

Feature selection. The w/o Top-D selection condition selects features randomly while preserving the per-token feature count. The w/o feature budget condition replaces Top-96 with every nonzero changed coordinate, at most 384 from the union of the two K192 supports, using guidance answer tokens only. A/B targets and the full-history energy denominator remain fixed when changing coordinate selection, without matching final update magnitudes. Full-support coverage is checked, and selected-coordinate counts, A/B counts, and effective update norms are recorded.

Branch combination. The w/o continuation and w/o suppression conditions use the same fitted suppression and continuation branches as Full PUSH, retaining the original coefficient of the remaining branch when the other is removed. Full PUSH, w/o continuation, and w/o suppression use coefficients $(2, 1)$, $(2, 0)$, and $(0, 1)$, respectively. They require no additional branch fitting. The validation scan in Figure 6 is separate from the test-set comparisons in Table 4.

Target Layer. We compare H20 and H24 targets with the H28 Full PUSH reference at the seven starting points used in the main ablation table. All conditions use $(\alpha, \beta) = (2, 1)$, $s = 1$, the same guidance data, and the layer-specific SAEs in Table 15. Each entry reports the mean ± sample standard deviation across GRPO Seeds 1–3.

A step-240 check confirms that H20 and H24 modify 280 and 336 factor tensors in the first 20 and 24 layers, respectively, while later-layer weights remain unchanged. A positive Δ indicates a higher gain than Full PUSH.

Table 26: Supplementary ablations on GRPO (test split, $s = 1$). The table uses the seven starting steps from Table 4. Each entry is the mean ± sample standard deviation across Seeds 1–3. Within each block, Full PUSH is the corresponding reference: H28 for target-layer tests, the original guidance set for guidance tests, and Top-96 for the feature-budget test. Δ is the mean difference from the Full PUSH row, summarized across the seven steps within each seed.

Variant	150	210	240	420	510	570	600	Δ vs. full
Target layer								
Full PUSH	$2.96_{\pm 0.07}$	$5.96_{\pm 0.19}$	$4.13_{\pm 0.82}$	$3.71_{\pm 0.14}$	$3.50_{\pm 0.25}$	$2.96_{\pm 0.14}$	$3.33_{\pm 0.56}$	—
H20 target	$2.58_{\pm 0.80}$	$5.42_{\pm 1.05}$	$3.38_{\pm 1.42}$	$1.04_{\pm 0.19}$	$0.75_{\pm 0.13}$	$1.00_{\pm 0.13}$	$1.83_{\pm 0.26}$	$-1.51_{\pm 0.50}$
H24 target	$2.96_{\pm 0.69}$	$6.08_{\pm 1.12}$	$3.63_{\pm 0.70}$	$1.63_{\pm 0.25}$	$1.17_{\pm 0.80}$	$2.04_{\pm 0.31}$	$2.21_{\pm 0.73}$	$-0.98_{\pm 0.15}$
Guidance set								
Full PUSH	$2.96_{\pm 0.07}$	$5.96_{\pm 0.19}$	$4.13_{\pm 0.82}$	$3.71_{\pm 0.14}$	$3.50_{\pm 0.25}$	$2.96_{\pm 0.14}$	$3.33_{\pm 0.56}$	—
Disjoint set $\mathcal{G}_1$	$3.08_{\pm 0.07}$	$5.75_{\pm 0.13}$	$4.75_{\pm 0.13}$	$3.75_{\pm 0.13}$	$3.38_{\pm 0.13}$	$3.50_{\pm 0.13}$	$2.50_{\pm 0.13}$	$0.02_{\pm 0.05}$
Disjoint set $\mathcal{G}_2$	$2.79_{\pm 0.07}$	$5.38_{\pm 0.13}$	$5.13_{\pm 0.13}$	$3.50_{\pm 0.13}$	$3.75_{\pm 0.13}$	$3.13_{\pm 0.13}$	$2.63_{\pm 0.13}$	$-0.04_{\pm 0.03}$
Feature budget								
Full PUSH	$2.96_{\pm 0.07}$	$5.96_{\pm 0.19}$	$4.13_{\pm 0.82}$	$3.71_{\pm 0.14}$	$3.50_{\pm 0.25}$	$2.96_{\pm 0.14}$	$3.33_{\pm 0.56}$	—
Top-192	$2.71_{\pm 0.07}$	$6.00_{\pm 0.13}$	$3.88_{\pm 0.70}$	$3.88_{\pm 0.13}$	$3.21_{\pm 0.14}$	$2.96_{\pm 0.26}$	$3.13_{\pm 0.63}$	$-0.11_{\pm 0.07}$

All three seed groups use the same seven starting steps as Table 4; the appendix adds the layer comparison without introducing a different step range.

Choice of Guidance Problems. The two disjoint 64-problem guidance sets exclude original guidance, development, and evaluation problems; selection uses no correctness labels. The guidance-set block of Table 26 compares them with the archived original-guidance PUSH and baseline results.

All seven starting points use $s = 1$, with the default optimization settings. Both subsets retain positive gains at every starting point.

Number of Guidance Problems. At DAPO step 360, we vary guidance size across 48/64/80/128 problems while retaining the default update count and other PUSH settings. The respective batch sizes are 6/8/10/16 problems. The 48-problem condition uses the first six problems of each original batch, while the 80- and 128-problem conditions add two

or eight problems to each original eight-problem batch; correctness labels are not used for selection. Evaluation on the test split and MBPP378 uses batch size 8, with remaining settings from Table 15. All paired 95% intervals on the test split relative to the 64-problem condition include zero.

After merging the effective updates ΔW_n across 196 modules, we compute the directional cosine c_n, norm ratio r_n, and projection ratio p_n relative to the 64-problem condition (Table 27):

$$c_n = \frac{\langle \Delta W_n, \Delta W_{64} \rangle_F}{\|\Delta W_n\|_F \|\Delta W_{64}\|_F}, \qquad r_n = \frac{\|\Delta W_n\|_F}{\|\Delta W_{64}\|_F}, \qquad p_n = \frac{\langle \Delta W_n, \Delta W_{64} \rangle_F}{\|\Delta W_{64}\|_F^2} = c_n r_n. \quad (16)$$

Table 27: Guidance-set size at DAPO step 360: task accuracy and parameter changes. Accuracy is reported in percent, with denominators of 800 for ACE-Test and 378 for MBPP and MBPP+. Parameter changes are relative to the 64-problem condition.

	Accuracy (%)			Parameter changes		
Guidance problems	Test split	MBPP	MBPP+	Direction cosine c_n	Norm ratio r_n	Projection ratio p_n
48	68.25%	85.45%	74.07%	0.98	0.99	0.97
64	69.13%	85.71%	74.87%	1.00	1.00	1.00
80	68.63%	85.45%	74.34%	0.97	1.04	1.01
128	68.63%	85.45%	74.07%	0.96	1.06	1.01

Feature Budget. The main experiments use $\gamma = 1$ and $D = K/2$, instantiated as $D = 96$ for the $K = 192$ SAEs. The coordinate-budget comparison uses the step-570 starting point and default guidance set, selects at most 96, 192, or 384 features with the largest change at every answer token, and then applies the same A/B partition and target construction. The SAE remains K192, so the pre-to-current difference has at most 384 nonzero entries; the largest budget covers the complete change.

Before PUSH fitting, we measure the target energy of each branch over all guidance tokens. For A, the target is zero, so suppression energy is the sum of $r^2_{\mathrm{now},t,i}$ over selected support exits, using the pre-TopK activations fitted by the A loss. For B, squared target displacement is the sum of $([2z_{\mathrm{now},t,i} - z_{\mathrm{pre},t,i}]_+ - z_{\mathrm{now},t,i})^2$ over the selected B coordinates. Each branch uses its own 384-feature result as 100% (Table 28).

The feature-budget block of Table 26 includes the intermediate budget $D = 192$ at the seven starting points in Table 4, using $s = 1$ and otherwise unchanged settings. Full PUSH ($D = 96$) and w/o feature budget (at most 384) are reported in that main-text table. These test-set comparisons are separate from the step-570 target-coverage measurements above.

Table 28: Feature budget and target coverage. Coverage is normalized separately within suppression and continuation to the full changed-feature budget.

	Suppression (A)		Continuation (B)	
Features per token	Squared magnitude	Coverage	Squared magnitude	Coverage
96	6.68	81.71%	500.23	94.99%
192	8.10	99.07%	523.89	99.49%
384	8.17	100.00%	526.59	100.00%

F Limitations and Future Work

Our capability results cover both code and mathematical RLVR, providing evidence that PUSH can improve RLVR outcomes in both settings. More broadly, compute and time constraints have so far limited validation beyond the current RLVR settings, including RLHF and other reinforcement-learning objectives. We have not yet tested how well the same framework transfers to these settings.

MBPP and MBPP+ provide an early check on transfer to external tasks and do not enter PUSH configuration development (Appendix C.3); broader task and model coverage is still needed.

PUSH constructs targets from the cumulative change since pre-RL, which includes learning completed in earlier stages. Near a training plateau, the effect of this cumulative direction can therefore remain entangled with the size of the update. Separating these two effects would support stage-adaptive target construction.

Future work will investigate whether PUSH transfers beyond the evaluated RLVR settings, including to RLHF and other reinforcement-learning objectives, as well as supervised fine-tuning (SFT) and other training procedures. It will also evaluate additional base models, datasets, and task domains, and test stage-adaptive PUSH across more points in training. For interpretability, future work can characterize candidate semantic associations and functional roles of the selected features, test whether similar feature changes correspond to consistent behaviors across tasks and training stages, and study their relationship with capability gains.