# BEYOND REWARD SPARSITY: HOW SAMPLE DIFFICULTY SHAPES LEARNING DYNAMICS IN RLVR

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has become a powerful approach for improving the reasoning capabilities of large language models, yet how training sample difficulty shapes the learning process remains poorly understood. Existing analyses often view difficulty through the lens of reward sparsity, but reward availability alone does not explain why some samples improve reasoning while others lead to unstable updates. In this work, we present a learning-dynamics analysis of sample difficulty in RLVR. We distinguish static sample difficulty, measured by base-policy solvability, from dynamic learning-signal availability under the evolving policy, and derive a per-update decomposition of RLVR optimization into three components: the training signal, its transfer through shared parameters, and the resulting prediction change. Through controlled experiments on MATH, we show that different difficulty regimes induce distinct learning behaviors: frontier samples provide stable improvements, while hard samples can produce harmful updates despite retaining non-trivial signals. Using sparse autoencoders and feature interventions, we further reveal that difficulty-dependent training reshapes internal feature dynamics, with samples of similar difficulty producing heterogeneous feature changes depending on their realized rollout groups. These results provide a mechanistic understanding of how sample difficulty governs RLVR learning dynamics beyond reward sparsity.

## 1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has recently become a central approach for improving the reasoning abilities of large language models (LLMs) (Shao et al., 2024; Lambert et al., 2025). By optimizing against automatically checkable outcomes, RLVR enables models to acquire complex reasoning behaviors without requiring explicit supervision over intermediate solutions. This has motivated a rapidly growing line of work on improving RLVR optimization, including more efficient policy objectives (Shao et al., 2024; Yu et al., 2025; Yue et al., 2025b), improved sampling strategies, and extensions to new domains (Xue et al., 2025; Liu et al., 2025a).

Despite this progress, most RLVR methods treat training samples uniformly, overlooking how sample difficulty influences the learning process. Under group-relative objectives such as GRPO (Shao et al., 2024), a sample provides direct learning signal only when its rollout group contains both successful and failed trajectories. Easy samples quickly become saturated, while extremely difficult samples rarely produce successful rollouts. However, empirical observations suggest that difficulty is not merely a measure of reward sparsity: some hard samples improve reasoning, whereas others lead to unstable or harmful updates. The mechanism underlying these different outcomes remains unclear.

Existing approaches address sample difficulty primarily through outcome-level statistics. Curriculum methods schedule examples according to difficulty (Luo et al., 2025), online filtering methods select samples with informative reward distributions (Bae et al., 2026), and entropy- or variance-based approaches attempt to recover learning signals from low-feedback samples (Le et al., 2026). While effective, these methods mainly characterize which samples should be selected. They provide limited understanding of how samples of different difficulty reshape the optimization trajectory and the internal representations of the model. We therefore ask: *How does sample difficulty shape the learning dynamics of RLVR?*



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.

In this work, we study RLVR through the lens of learning dynamics. We distinguish between two notions of difficulty: a fixed *reference difficulty*, measured by the base policy's empirical success rate, and an evolving *learning-signal availability*, which describes whether the current policy can obtain a non-trivial group-relative signal. We further derive a per-update influence decomposition for GRPO, showing that the effect of an RLVR update on an observation is determined by three factors: the training signal produced by the rollout group, its transfer through shared parameters, and the output readout at the observation. This perspective allows us to analyze difficulty-conditioned learning at multiple levels, from behavioral changes to internal feature dynamics.

Our analysis reveals three key findings. First, learning-signal availability alone does not determine learning utility. Easy samples rapidly lose signal after being mastered, while hard samples may retain signal but induce weak or misaligned updates; in contrast, frontier samples provide more consistent improvements on related problems. Second, difficulty shapes the internal evolution of the model. Using a shared sparse autoencoder feature space, we find that different difficulty regimes produce distinct feature trajectories, and samples within the same difficulty regime can reinforce or suppress the same feature in opposite directions. Third, these feature dynamics are behaviorally meaningful and predictable. Steering SAE-derived feature directions changes model outputs, and the observed feature changes from individual RLVR updates can be accurately predicted from the realized optimizer displacement. Together, these results show that sample difficulty influences RLVR not only through reward availability, but also through the direction and transfer of learning signals.

Our contributions are summarized as follows:

- We introduce a learning-dynamics perspective on sample difficulty in RLVR, separating static difficulty from online learning-signal availability and decomposing GRPO updates into signal, transfer, and readout components.
- We provide controlled experiments on MATH showing that different difficulty regimes lead to distinct optimization behaviors: frontier samples improve related reasoning performance, while hard-only training can degrade generalization despite retaining learning signals.
- We analyze the internal mechanisms behind these effects using sparse autoencoders and interventions, showing that difficulty-dependent updates reshape reasoning-related features and that feature changes depend on the realized rollout group.

## Related Work

**Reinforcement Learning with Verifiable Rewards.** RLVR improves LLM reasoning through verifiable reward signals. PPO (Schulman et al., 2017) relies on a critic, introducing additional computational cost and training sensitivity. VAPO (Yue et al., 2025b) improves PPO-style training for long-CoT reasoning, while GRPO (Shao et al., 2024) removes the critic through group-relative reward normalization. Subsequent methods refine this framework: Dr. GRPO (Liu et al., 2025b) corrects optimization biases; DAPO (Yu et al., 2025) improves clipping, sampling, loss aggregation, and reward shaping; GSPO (Zheng et al., 2025) adopts sequence-level optimization; and DCPO (Yang et al., 2025) dynamically adjusts clipping. REINFORCE-style alternatives, including ReMax (Li et al., 2023), REINFORCE++ (Hu et al., 2025), and RLOO (Ahmadian et al., 2024), simplify optimization with lightweight baselines. Beyond optimizer design, PRIME (Cui et al., 2025) introduces implicit process rewards, Scaf-GRPO (Zhang et al., 2026) guides learning on zero-reward hard problems, ExGRPO (Zhan et al., 2026) reuses experience, and NPO (Qin et al., 2026) draws trajectories from near-future checkpoints to balance quality and policy proximity. DanceGRPO (Xue et al., 2025) and Flow-GRPO (Liu et al., 2025a) further extend GRPO-style training to visual generation and flow matching. These works improve training stability, efficiency, and scalability; we examine which samples yield useful RLVR updates and why hard samples are unstable.

**Sample Difficulty and Effectiveness in RLVR.** Recent work examines how prompt difficulty affects RLVR training signals. Yue et al. (2025a) shows that RL gains are often bounded by base-model capabilities, suggesting limited feedback from prompts beyond the model's capability frontier. Following curriculum learning (Bengio et al., 2009), E2H Reasoner (Parashar et al., 2025) schedules tasks from easy to hard, while Online Difficulty Filtering (Bae et al., 2026) prioritizes intermediate-difficulty prompts. VCRL (Jiang et al., 2025) uses rollout reward variance to identify prompts with nonzero group-relative advantages, and DEPO (Tang et al., 2026) combines data selection, online

exploration, and replay to improve data efficiency. RL-ZVP (Le et al., 2026) revisits zero-variance prompts through entropy-guided advantage shaping. To make harder problems learnable, DARS and DARS-B (Yang et al., 2026) expand exploration in depth and breadth, QuestA (Li et al., 2026) augments questions, ReGFT (Wu et al., 2026) introduces reference-guided fine-tuning, and P$^2$O (Lu et al., 2026) jointly optimizes prompts and policies. Our work complements these approaches by investigating hard-sample instability through curriculum-level, one-sample, and SAE feature-level analyses. We show that hard samples have heterogeneous effects: some induce useful policy changes, while others suppress existing reasoning features or reinforce brittle shortcuts, harming performance.

## 2 Preliminaries

**Notation.** We denote an LLM policy parameterized by weights $\theta \in \mathbb{R}^{d_\theta}$ as $\pi_\theta$, and let $\theta^\tau$ denote its parameters after $\tau$ optimizer updates. We sometimes also directly use $\pi_{\text{base}} = \pi_{\theta^0}$ to denote a pre-trained model (e.g., Qwen2.5-math-1.5B, Qwen2.5-math-7B) to contrast its RL counterpart $\pi_\theta$. We use $\mathcal{D}$ to denote the query distribution and $\mathcal{V}$ to denote the vocabulary. Given an input query $\boldsymbol{x}$, the model's output is modeled by $\pi_\theta(\cdot|\boldsymbol{x})$. We use $\boldsymbol{y} \sim \pi_\theta(\cdot|\boldsymbol{x})$ to denote the sampling of answer $\boldsymbol{y}$. During RL training, $\pi_{\theta^i}$ denotes the model after the $i$-th policy update.

**Reinforcement Learning with Verifiable Reward.** We formalize the problem of fine-tuning an LLM for complex reasoning tasks under the RLVR (Lambert et al., 2025) paradigm. At each step $t$, the state $s_t = (\boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ is the concatenation of the query $\boldsymbol{x}$ and previously generated tokens, and the action $a_t$ is to select the next token $\boldsymbol{y}_t \in \mathcal{V}$. The LLM with parameters $\theta$ serves as the policy $\pi_\theta$, producing $a_t \sim \pi_\theta(\cdot \mid s_t)$, so that the probability of a trajectory $\boldsymbol{y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T)$ is

$$\pi_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} \pi_\theta(\boldsymbol{y}_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}), \tag{1}$$

A verifier assigns a sparse, terminal reward $R(\boldsymbol{y})$, which equals 1 if the final answer extracted from $\boldsymbol{y}$ is correct and 0 otherwise. Then, the training objective is therefore optimized with policy gradient methods,

$$\mathcal{J}(\theta) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}, \boldsymbol{y}\sim\pi_\theta(\cdot|\boldsymbol{x})}\big[R(\boldsymbol{y})\big]. \tag{2}$$

The gradient of the objective with respect to the policy parameters $\theta$ can be expressed as

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\, \mathbb{E}_{\tau\sim\pi_\theta(\cdot|\boldsymbol{x})}\Big[R(\tau)\nabla_\theta \log \pi_\theta(\tau|\boldsymbol{x})\Big]. \tag{3}$$

where gradient coefficient $\mathcal{A}$ depends on the verifier and reward feedback. Detailed policy gradient method forms, such as GRPO (Shao et al., 2024), are provided in the Appendix A.1.

**Sparse Autoencoders.** Sparse autoencoders (SAEs) (Ng et al., 2011; Huben et al., 2024) are a class of unsupervised interpretability models that effectively learn sparse feature decompositions of dense LLM activations. Taking in input $\boldsymbol{h}_t \in \mathbb{R}^d$, they map $\boldsymbol{h}_t$ into an overcomplete sparse feature vector $\boldsymbol{z}_t \in \mathbb{R}^m$, typically with $m \gg d$, and reconstructs the original activation from this sparse representation:

$$\boldsymbol{z}_t = f_\theta(\boldsymbol{h}_t) = \sigma(\boldsymbol{W}^{\text{enc}}\boldsymbol{h}_t + \boldsymbol{b}^{\text{enc}}), \qquad \hat{\boldsymbol{h}}_t = \boldsymbol{W}^{\text{dec}}\boldsymbol{z}_t + \boldsymbol{b}^{\text{dec}} \tag{4}$$

where $\boldsymbol{W}^{\text{enc}} \in \mathbb{R}^{m\times d}$ is the encoder matrix, and $\boldsymbol{W}^{\text{dec}} \in \mathbb{R}^{d\times m}$ is the decoder matrix. The encoder and decoder bias are $\boldsymbol{b}^{\text{enc}} \in \mathbb{R}^m$ and $\boldsymbol{b}^{\text{dec}} \in \mathbb{R}^d$, respectively.

## 3 How Does Sample Difficulty Shape RLVR Learning Dynamics?

### 3.1 Reference Difficulty and Learning-Signal Availability

The difficulty of a training sample is closely related to the learning signal it can provide during RLVR. In this section, we study *how the difficulty of a training sample affects learning on other samples at the level of individual updates*. Inspired by the notion of $k$-rollout unsolvable problems (Deng et al., 2025), we first define a fixed reference difficulty using rollouts from the base policy $\pi_{\text{base}}$, separating sample difficulty from the evolving policy.

**Reference Difficulty.** For each training sample $\boldsymbol{x}$, we roll out $n$ responses from $\pi_{\text{base}}$ and use the empirical success rate to define its reference difficulty:

$$\widehat{p}_{0,n}(\boldsymbol{x}) = \frac{\sum_{j=1}^{n} R(\boldsymbol{x}, \boldsymbol{y}_j^0)}{n}. \tag{5}$$

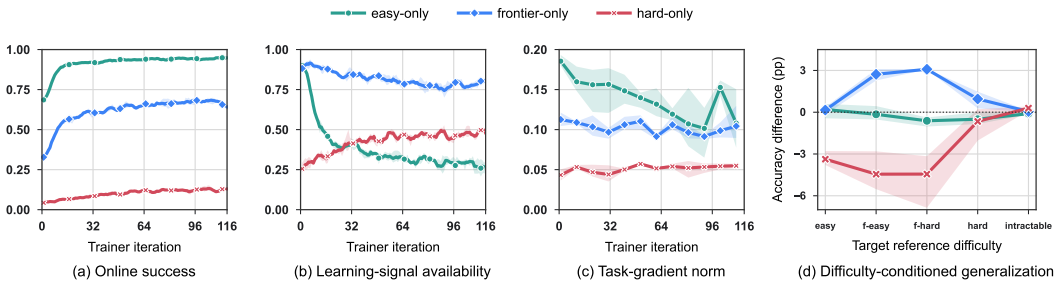
where $\boldsymbol{y}_j^0 \sim \pi_{\text{base}}(\cdot \mid \boldsymbol{x})$ is the $j$-th rollout of $\boldsymbol{x}$. We divide this range in increasing order of empirical success rate into five fixed difficulty levels: `intractable`, `hard`, `frontier-hard`, `frontier-easy`, and `easy`, with lower success rates corresponding to harder samples. These labels remain unchanged throughout RL training. See Appendix A.3 for the exact difficulty thresholds.

**Learning-Signal Availability.** Reference difficulty stays fixed during training, but the learning signal from a sample changes with the current policy $\pi_{\theta^\tau}$. Under outcome-supervised GRPO, a rollout group has a nonzero relative reward signal only when it contains both correct and incorrect responses.

Let $p_\tau(\boldsymbol{x}) = \Pr_{\boldsymbol{y}\sim\pi_{\tilde{\theta}^\tau}(\cdot|\boldsymbol{x})}[R(\boldsymbol{x}, \boldsymbol{y}) = 1]$ be the success probability of $\boldsymbol{x}$ under the behavior policy at update $\tau$. For $G$ independent rollouts, the probability that a group contains both outcomes is

$$\Gamma_\tau(\boldsymbol{x}) = 1 - p_\tau(\boldsymbol{x})^G - \big(1 - p_\tau(\boldsymbol{x})\big)^G. \tag{6}$$

We call $\Gamma_\tau(\boldsymbol{x})$ the *learning-signal availability*. Unlike reference difficulty, it changes as the policy evolves. However, signal availability only tells us whether a learning signal is present, not how the resulting update affects another sample.




(a) Online success (b) Learning-signal availability (c) Task-gradient norm (d) Difficulty-conditioned generalization

Figure 1: Difficulty-conditioned learning dynamics on MATH. Lines show means over three seeds; shading spans the seed-wise minimum and maximum. (a) Online source success. (b) Learning-signal availability, measured by the fraction of responses with nonzero group-relative advantage. (c) Task-gradient norm. (d) Final held-out probe accuracy relative to the size-matched control. Here, f-easy and f-hard denote frontier-easy and frontier-hard.

**Empirical Dynamics on MATH.** We evaluate Qwen2.5-Math-1.5B using easy-only, frontier-only, and hard-only subsets, each paired with a size-matched natural control, over three seeds (18 runs); reference difficulty is calibrated with 1,024 base-policy responses, and 1,455 samples are held out as a fixed probe. Figure 1(a) shows that easy samples are rapidly mastered, frontier samples improve steadily, and hard samples remain difficult. Accordingly, Figure 1(b) shows that easy-sample signal availability collapses from 0.899 to 0.264, whereas frontier signal remains high and hard-sample signal increases from 0.255 to 0.479. Yet hard-only training retains the smallest gradient norm in Figure 1(c), showing that signal availability does not determine update magnitude. Finally, Figure 1(d) shows that frontier-only training improves frontier-easy and frontier-hard probe accuracy by 2.7 and 3.1 points over its control, whereas hard-only training reduces easy, frontier-easy, and frontier-hard accuracy by 3.4, 4.4, and 4.4 points.

### 3.2 Per-Step Learning Dynamics Across Samples

The aforementioned results show that sample difficulty affects when a learning signal is available, but signal availability alone does not determine how an update changes other samples. While $\Gamma_\tau(\boldsymbol{x})$ captures whether a training sample can provide a group-relative signal, it does not describe where this signal transfers. Following Ren & Sutherland (2025), we study this effect at the level of individual updates by separating the training signal, its transfer through shared parameters, and the resulting prediction change.

**Per-Step Influence Decomposition.** In this section, we distinguish the training sample that induces an update from the observation on which its effect is measured. Let $o = (\boldsymbol{x}_o, \boldsymbol{y}_{o,<t_o})$ denote a fixed observation context, with next-token logits $\boldsymbol{u}_\theta(o)$ and probabilities $\boldsymbol{\pi}_\theta(o) = \mathrm{softmax}(\boldsymbol{u}_\theta(o))$. We measure its prediction change after update $\tau$ by

$$\Delta \log \boldsymbol{\pi}_\tau(o) = \log \boldsymbol{\pi}_{\theta^{\tau+1}}(o) - \log \boldsymbol{\pi}_{\theta^\tau}(o). \tag{7}$$

The observation context is held fixed, whereas the parameter change is induced by a training sample and its realized rollout group. Previous work (Ren & Sutherland, 2025) studies such per-step prediction changes in supervised learning. We extend this view to GRPO (Shao et al., 2024), where an update is driven by group-relative feedback from sampled responses.

For a training sample $\boldsymbol{x}$, let $\mathcal{U}_{\boldsymbol{x}}^\tau = \{\boldsymbol{y}_j^\tau\}_{j=1}^G$ denote its realized rollout group at update $\tau$. For brevity, $\boldsymbol{u}_\theta(\mathcal{U}_{\boldsymbol{x}}^\tau)$ denotes the logits over all generated token positions in this group, with the common conditioning query $\boldsymbol{x}$ suppressed. We define the training-side signal in logit space as

$$\mathcal{G}_\tau(\boldsymbol{x}) = \left.\frac{\partial \mathcal{L}_{\boldsymbol{x},\tau}}{\partial \boldsymbol{u}_\theta(\mathcal{U}_{\boldsymbol{x}}^\tau)}\right|_{\theta=\theta^\tau}, \tag{8}$$

where $\mathcal{L}_{\boldsymbol{x},\tau}$ is the policy-gradient loss for the rollout group, excluding explicit regularization terms. Because $\mathcal{G}_\tau(\boldsymbol{x})$ depends on the group-relative advantages, it vanishes when all responses in the group receive the same reward. Inspired by (Ren & Sutherland, 2025), we now state the decomposition of the per-step influence in GRPO below

**Proposition 1** *For a fixed observation context $o$, The one-step learning dynamics of GRPO decompose as*

$$\Delta \log \boldsymbol{\pi}_\tau(o) = -\eta_\tau \underbrace{\mathcal{T}_\tau(o)}_{\textit{readout}} \underbrace{\mathcal{K}_\tau(o, \mathcal{U}_{\boldsymbol{x}}^\tau)}_{\textit{transfer}} \underbrace{\mathcal{G}_\tau(\boldsymbol{x})}_{\textit{signal}} + \mathcal{O}\left(\|\Delta\theta^\tau\|^2\right), \tag{9}$$

*where $\mathcal{K}_\tau(o,\mathcal{U}) = \mathbf{J}_u^\tau(o)\mathbf{M}_\tau\mathbf{J}_u^\tau(\mathcal{U})^\top$ describes how this signal propagates through shared parameters to the observation. $\mathbf{J}_u^\tau(\mathcal{U}) = \left.\frac{\partial \boldsymbol{u}_\theta(\mathcal{U})}{\partial\theta}\right|_{\theta=\theta^\tau}$ is the corresponding logit Jacobian. $\mathbf{M}_\tau$ denotes the local update map. For SGD, $\mathbf{M}_\tau = \mathbf{I}$; more generally, $\mathbf{M}_\tau$ can represent local preconditioning. $\mathcal{G}_\tau$ captures how the realized rollout group acts on its own logits. $\mathcal{T}_\tau = \mathbf{I} - \mathbf{1}\boldsymbol{\pi}_{\theta^\tau}(o)^\top$ converts the resulting logit change into a change in log probability.*

As shown in Proposition 1, the effect of a single GRPO update on a fixed observation can be decomposed into three components: the training signal induced by the realized rollout group, its transfer through shared parameters, and the output readout at the observation. While this decomposition characterizes the per-step influence of a training sample on $\boldsymbol{\pi}_\theta(o)$, training consists of a sequence of such updates. The accumulated effect therefore depends on their iterative interaction, as both the policy and the realized rollout groups evolve over training. Detailed derivations are deferred to Appendix B.1.

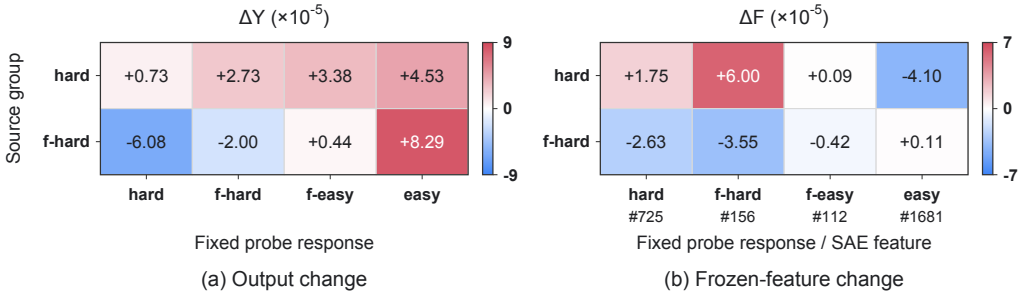



Figure 2: Effects of the same source updates in output and feature space. (a) Mean token log-probability change $\Delta Y_r$. (b) Mean activation change $\Delta F_r$ of one frozen SAE feature per response. Both use the same anchor and $\eta = 10^{-3}$.

**Accumulated Influence and a Demonstration on MATH.** Per-step influence is only local: as training proceeds, the signal, transfer, and readout evolve with the policy, so their effects may reinforce or cancel over time. To make this concrete, we consider a MATH example with four source groups and four fixed responses at a common Qwen2.5-Math-1.5B checkpoint. See Figure 2(a).

Updating on the hard source increases the log-probability of all four responses. In contrast, the frontier-hard source decreases it for the harder responses but increases it for the easier ones. Thus, a nonzero training signal can induce opposite effects on different observations. Results for all sources are in Appendix C.

We can ask the same question in feature space. Keeping the source updates fixed, we replace $Y_r$ with a frozen SAE readout $F_r$, with $\Delta F_r \approx -\eta\langle \nabla_\theta F_r, g_x\rangle$. As Figure 2(b) shows, the same updates can increase or decrease feature activations, and their signs need not agree with the output changes. Forward checks recover the predicted signs in all 16 cases for each readout (Appendix C.3). These examples illustrate how influence depends on both the observation and the readout; they do not imply feature mediation or difficulty-level averages. Over training, such effects accumulate while their underlying factors evolve, so a source's signal magnitude at one checkpoint alone does not determine its cumulative influence.

# 4 How Does Sample Difficulty Reshape Internal Feature Dynamics?

The previous analysis shows that sample difficulty affects RLVR through the quality and transfer of learning signals. However, these effects are observed at the level of outputs and parameter updates. We next investigate how difficulty-dependent updates reshape the model's internal representations during RLVR training.

## 4.1 Feature Dynamics in a Shared SAE Space

To track representation changes across training, we construct a shared sparse feature space using a frozen sparse autoencoder (SAE). The SAE is trained on activations collected from the base model and intermediate RL checkpoints, after which the feature dictionary and normalization are fixed across all training conditions. This provides a consistent coordinate system for comparing feature evolution throughout RL training.

For each observation context $o$, the SAE encodes the hidden activation at layer $\ell$ into a sparse feature vector:

$$\boldsymbol{z}_\theta(o) = E_\varphi\big(\boldsymbol{h}_\theta^\ell(o)\big) \in \mathbb{R}^m, \tag{10}$$

where $m$ is the number of sparse features. Thus, each feature $f$ defines the same scalar readout throughout training.

We evaluate all checkpoints on a fixed bank of prompt–response sequences under teacher forcing and at identical token positions. Let $\mathcal{Q}$ denote the corresponding observation distribution: we sample a question uniformly, a stored response uniformly within the question, and a retained token position uniformly within the response. For each feature $f$, we measure its mean activation and firing rate as

$$F_f(\theta^\tau) = \mathbb{E}_{o\sim\mathcal{Q}}\left[z_{\theta^\tau,f}(o)\right], \tag{11}$$

$$\rho_f(\theta^\tau) = \mathbb{E}_{o\sim\mathcal{Q}}\left[\mathbf{1}\left\{z_{\theta^\tau,f}(o) > 0\right\}\right]. \tag{12}$$

For difficulty regime $d$, we compare its feature trajectory with the corresponding size-matched natural control in Section 3.1. Pairing runs by training step and random seed, we define

$$\Delta F_f^d(\tau) = F_f(\theta^{\tau,d}) - F_f(\theta^{\tau,\mathrm{ctrl},d}), \tag{13}$$

and define $\Delta\rho_f^d(\tau)$ analogously. The final feature shift is $\Delta F_f^d(T)$ at the common training endpoint.

Since different SAE features may have different decoder scales, we additionally report

$$\Delta\widetilde{F}_f^d(\tau) = \|s\boldsymbol{d}_f\|_2 \Delta F_f^d(\tau), \tag{14}$$

where $\boldsymbol{d}_f$ is the frozen decoder direction and $s$ maps normalized SAE activations back to the original hidden-state scale. These quantities measure changes in feature strength and firing frequency relative to matched natural training.

**Sample difficulty induces distinct feature trajectories.** Figure 3 shows that different difficulty regimes produce qualitatively different feature evolution during RLVR training. Frontier training

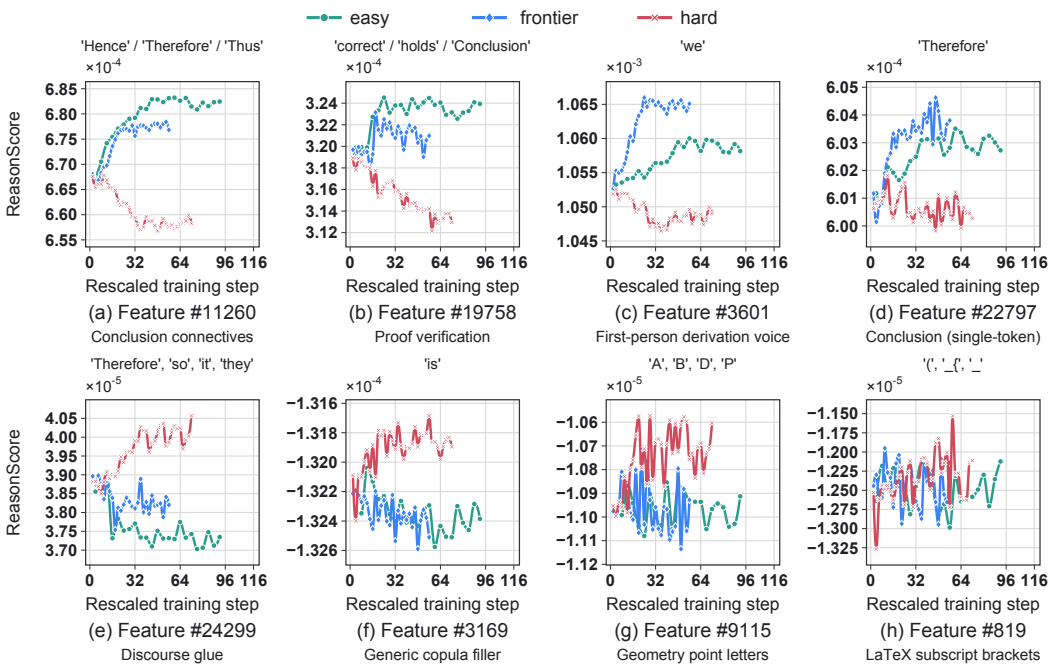



Figure 3: Token-level SAE feature dynamics along representative reasoning trajectories after RL on samples of different difficulty. The top row visualizes features suppressed by hard training, and the bottom row visualizes features reinforced by hard training. Hard samples tend to weaken existing reasoning features while amplifying more local or surface-level patterns.

leads to consistent feature shifts across checkpoints, whereas hard-only training produces more heterogeneous trajectories. The token-level evolution further reveals different adaptation patterns. Frontier training preferentially increases features frequently activated in reasoning-related contexts, such as intermediate verification and derivation patterns. Easy training mainly reinforces features associated with direct solution generation. In contrast, hard training exhibits mixed behavior: some reasoning-related features decrease while features correlated with local formatting or surface patterns increase. These results suggest that sample difficulty influences not only whether RLVR receives a learning signal, but also which internal computation patterns are reinforced during optimization.

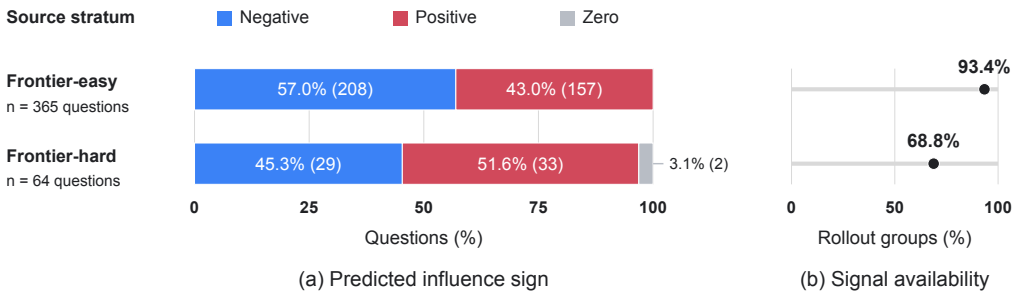
### 4.2 Which Training Samples Drive Feature Changes?

The above analysis describes the feature changes accumulated over training. We next investigate how individual training samples contribute to these changes by applying the local influence analysis in Section 3 to the frozen feature readout $F_f$.

**Sample-feature influence.** For a training question $\boldsymbol{x}$ and its realized rollout group $\mathcal{U}_{\boldsymbol{x}}^{\tau}$, the first-order change of feature $f$ under an isolated SGD update is

$$A_{\boldsymbol{x},f}^{\tau} = -\eta_{\tau} \left\langle \nabla_{\theta} F_f(\theta^{\tau}), \nabla_{\theta} \mathcal{L}\left(\theta^{\tau}; \mathcal{U}_{\boldsymbol{x}}^{\tau}\right) \right\rangle . \tag{15}$$

Unlike learning-signal availability, which only measures whether a sample produces a non-zero relative advantage, $A_{\boldsymbol{x},f}^{\tau}$ captures the direction of its effect on a specific internal feature. We analyze a frontier-strengthened feature and evaluate 429 samples from frontier-easy and frontier-hard regimes. For each sample, the influence score is averaged over four independent rollout groups. Figure 4 reveals substantial variation within each difficulty regime. Both frontier-easy and frontier-hard samples contain updates that strengthen and suppress the same feature. Meanwhile, most samples generate non-uniform rewards, indicating that reward variation alone does not determine feature-level effects. Together, these results show that difficulty is not a sufficient description of an RLVR update. The realized rollout group determines the learning signal, while the interaction between this signal and the model's current representation determines the resulting feature change.




(a) Predicted influence sign (b) Signal availability

Figure 4: Sample-level influence on the selected SAE feature. (a) Sign of the mean sample-feature influence within each reference-difficulty. (b) Fraction of rollout groups with non-uniform rewards.

## 5 Intervening on Feature Directions and RLVR Dynamics

The previous section establishes that difficulty-dependent RLVR training produces heterogeneous feature trajectories. We next examine whether these feature directions correspond to measurable behavioral changes and whether short-term feature evolution can be predicted from the actual optimization trajectory.

### 5.1 Steering Difficulty-Associated Feature Directions

Section 4 identifies feature directions whose activation profiles vary with sample difficulty. We now examine how these directions affect the model's predictions. To isolate their output effects, we hold the model parameters and observation sequences fixed and perturb the residual stream along individual SAE decoder directions. This setting allows us to characterize the response to a feature intervention independently of the parameter changes produced by RLVR.

**Intervention and output response.** Let $\mathcal{P}_r$ denote the retained intervention positions in a stored prompt–response pair $r = (\boldsymbol{x}, \boldsymbol{y})$. For feature $f$, we apply

$$\boldsymbol{h}^{\ell}_{\theta,\lambda}(o) = \boldsymbol{h}^{\ell}_{\theta}(o) + \lambda s \boldsymbol{d}_f, \qquad o \in \mathcal{P}_r, \tag{16}$$

where $\boldsymbol{d}_f$ is the frozen SAE decoder direction, $s$ converts it to the original hidden-state scale, and $\lambda$ controls the intervention strength. The model parameters, input tokens, and intervention positions remain fixed across doses. We measure the resulting mean response-token log-probability,

$$Y_{r,f}(\lambda) = \frac{1}{|\boldsymbol{y}|} \sum_{t=1}^{|\boldsymbol{y}|} \log \pi_{\theta,\lambda,f}\big(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}\big), \tag{17}$$

where $\pi_{\theta,\lambda,f}$ denotes the policy evaluated with the hidden-state intervention.

For a small intervention, the local response is described by $Y_{r,f}(\lambda) - Y_{r,f}(0) \approx \lambda J_{r,f}$, where

$$J_{r,f} = \left.\frac{\partial Y_{r,f}(\lambda)}{\partial \lambda}\right|_{\lambda=0} = \sum_{o \in \mathcal{P}_r} \left\langle \nabla_{\boldsymbol{h}^{\ell}(o)} Y_r,\ s\boldsymbol{d}_f \right\rangle, \tag{18}$$

and the hidden-state gradients are evaluated without intervention. This expression makes the role of the observation explicit. The decoder direction specifies the perturbation, while the downstream likelihood gradient determines how that perturbation affects a particular response. A direction's output effect therefore depends on its alignment with the computation used to predict the tokens.

We evaluate eleven features selected from discovery readouts of the shared layer-13 SAE. All interventions use the Qwen2.5-Math-1.5B base checkpoint and 516 validation responses, giving 5,676 feature–response comparisons. We compute $J_{r,f}$ by automatic differentiation and compare it with an independent finite-difference estimate,

$$\widehat{J}_{r,f} = \frac{Y_{r,f}(\epsilon) - Y_{r,f}(-\epsilon)}{2\epsilon}, \qquad \epsilon = 10^{-3}. \tag{19}$$

**Sensitivity Predicts Intervention Effects.** Analytic and finite-difference sensitivities correlate across all eleven features. Mean absolute errors are $(6.8\text{–}8.1) \times 10^{-5}$, compared with a mean absolute

finite-difference sensitivity of approximately $2.4 \times 10^{-4}$. Features 1614, 1719, 2235, and 2285 activate more frequently on harder questions and have negative mean analytic sensitivities, whereas feature 603 is more active on easy questions and has positive mean sensitivity. The means for features 1614 and 603 are $-8.5 \times 10^{-5}$ and $7.7 \times 10^{-5}$, respectively, indicating opposing local effects on the likelihood of stored responses. We next examine how training on individual questions changes the feature readout.

### 5.2 One-Sample RLVR Updates and Feature Dynamics

For the one-step experiment, we select twelve questions from the 429-question screening pool: four positive-score sources and eight controls drawn from negative-score, gradient-norm-matched, and random-matched candidates. Selection is performed within the frontier-easy and frontier-hard strata. Each question receives four fresh groups of $G = 8$ responses, with sampling seeds disjoint from screening. Before each of the 48 trials, we restore the same base parameters $\theta_0$ and initialize a fresh AdamW state. We then apply one optimizer step with learning rate $10^{-6}$ and measure feature 1614 on the same frozen observation bank.

AdamW transforms the sampled gradient before updating the parameters. We therefore evaluate the local prediction along its realized displacement $\Delta\theta_{\boldsymbol{x},\mathcal{U}}$:

$$\begin{aligned} P_{\boldsymbol{x},\mathcal{U},f} &= \langle \nabla_\theta F_f(\theta_0), \Delta\theta_{\boldsymbol{x},\mathcal{U}} \rangle , \\ O_{\boldsymbol{x},\mathcal{U},f} &= F_f(\theta_0 + \Delta\theta_{\boldsymbol{x},\mathcal{U}}) - F_f(\theta_0). \end{aligned} \tag{20}$$

where $P_{\boldsymbol{x},\mathcal{U},f}$ is the first-order prediction conditional on the actual step, and $O_{\boldsymbol{x},\mathcal{U},f}$ is measured by a forward pass through the updated model.

**Example 2.1:** ***Spurious Reward from Answer-Only Output***

**Question:** `In the diagram, $l\|k$. What is the number of degrees in $\angle SRQ$? [asy] draw ((-.4,-.4)--(2,2)--(2,-.4)); draw((-.5,0)--(3,0),Arrows); draw((-.5,1)--(3,1),Arrows); draw((1.9,0)--(1.9,.1)--(2,.1)); label("$S$",(1,1),NNW); label("$R$",(2,2),N); label("$Q$",(2,1),NE); label("$l$",(3,1),E); label("$k$",(3,0),E); label("$130^{\circ}$",(1,1),SSE); [/asy]`

**Answer:** $40^\circ$

**Response:** `The number of degrees in \angle SRQ is \boxed{40}.`

**One Sample Training.** Under one sample training, frontier-easy and hard samples generally preserve or improve downstream accuracy, whereas hard samples yield outcomes ranging from substantial gains to degradation below the base model. High training reward can coexist with large KL growth, and some harmful effects emerge only after repeated updates. Outcome rewards can also reinforce correct answer-only responses without supporting derivations, leaving the reliability of the underlying solution procedure unresolved. In GRPO, all-zero reward groups have zero task advantages, whereas mixed-reward groups reinforce rewarded trajectories. The resulting task signal therefore enables learning without guaranteeing useful transfer, which depends on the behavior reinforced by the realized rollouts. See more about this harmful example in Appendix E.

## 6 Conclusion

We studied how sample difficulty shapes the learning dynamics of RLVR beyond reward sparsity. By separating static reference difficulty from dynamic learning-signal availability, and by decomposing GRPO updates into signal, transfer, and readout components, we showed that obtaining a learning signal is only the first step: the effect of an RLVR update depends on how that signal is transferred through the model. Our analysis reveals that difficulty regimes induce distinct optimization trajectories. Easy samples rapidly saturate, frontier samples provide more consistent improvements on related problems, and hard samples can produce unstable updates despite retaining non-trivial reward signals. Using a shared sparse feature space and controlled interventions, we further show that these differences correspond to measurable changes in the model's internal representations. Importantly, samples with similar difficulty can produce opposite feature-level effects, determined by their realized rollout groups and current model state. Together, these findings suggest a shift in how RLVR data should be understood: difficulty is not merely a property of samples, but a factor that shapes the direction and transfer of learning signals during optimization.

## AI USE STATEMENT

We used generative AI tools solely to improve the language, clarity, and readability of this manuscript. Specifically, these tools were used for grammar correction and language polishing. They were not used to generate the core research ideas, formulate theoretical claims, design the methodology, conduct experiments, analyze experimental results, or make substantive scientific decisions. All AI-assisted text was carefully reviewed and revised by the authors, and the authors take full responsibility for the accuracy, originality, and integrity of the final manuscript.

## REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, 2026.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Jia Deng, Jie Chen, Zhipeng Chen, Daixuan Cheng, Fei Bai, Beichen Zhang, Yinqian Min, Yanzipeng Gao, Wayne Xin Zhao, and Ji-Rong Wen. From trial-and-error to improvement: A systematic analysis of llm exploration mechanisms in rlvr. *arXiv preprint arXiv:2508.07534*, 2025.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. *arXiv preprint arXiv:2501.03262*, 2025.

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

Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. VCRL: Variance-based curriculum reinforcement learning for large language models. *arXiv preprint arXiv:2509.19803*, 2025.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2025.

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Dac Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in LLM reinforcement learning via entropy-guided advantage shaping. In *The Fourteenth International Conference on Learning Representations*, 2026.

Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. QuestA: Expanding reasoning capacity in LLMs via question augmentation. In *The Fourteenth International Conference on Learning Representations*, 2026.

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. *arXiv preprint arXiv:2310.10505*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Xinyu Lu, Kaiqi Zhang, Jinglin Yang, Boxi Cao, Yaojie Lu, Hongyu Lin, Min He, Xianpei Han, and Le Sun. $P^2O$: Joint policy and prompt optimization. *arXiv preprint arXiv:2603.21877*, 2026.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.

Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, et al. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *arXiv preprint arXiv:2506.06632*, 2025.

Chuanyu Qin, Chenxu Yang, Qingyi Si, Naibin Gu, Dingyu Yao, Zheng Lin, Peng Fu, Nan Duan, and Jiaqi Wang. Near-future policy optimization. *arXiv preprint arXiv:2604.20733*, 2026.

Yi Ren and Danica J. Sutherland. Learning dynamics of LLM finetuning. In *International Conference on Learning Representations*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Xinyu Tang, Zhenduo Zhang, Yurou Liu, Xin Zhao, zujie wen, Zhiqiang Zhang, and JUN ZHOU. Towards high data efficiency in reinforcement learning with verifiable reward. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yangzhen Wu, Shanda Li, Zixin Wen, Xin Zhou, Ameet Talwalkar, Yiming Yang, Wenhao Huang, and Tianle Cai. Learn hard problems during RL with reference guided fine-tuning. *arXiv preprint arXiv:2603.01223*, 2026.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. Dancegrpo: Unleashing grpo on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

Shihui Yang, Chengfeng Dou, Peidong Guo, Kai Lu, Qiang Ju, Fei Deng, and Rihui Xin. Dcpo: Dynamic clipping policy optimization. *arXiv preprint arXiv:2509.02333*, 2025.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Yongxin Wang, Dongchun Xie, Hanhui Li, Yiwei Wang, Xiaodan Liang, and Jing Tang. Depth-breadth synergy in rlvr: Unlocking llm reasoning gains with adaptive exploration. *arXiv preprint arXiv:2508.13755*, 2026.

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. *arXiv preprint arXiv:2503.14476*, 2025.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025a.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025b.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. ExGRPO: Learning to reason from experience. In *International Conference on Learning Representations*, 2026.

Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-GRPO: Scaffolded group relative policy optimization for enhancing LLM reasoning. In *International Conference on Learning Representations*, 2026.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

# APPENDIX

## A MORE ABOUT POLICY ALGORITHMS AND DEFINITIONS

### A.1 MORE ABOUT POLICY OPTIMIZATION ALGORITHMS

**Group Relative Policy Optimization (GRPO).** GRPO (Shao et al., 2024) is an efficient RL algorithm introduced by DeepSeek for training its `DeepSeek-Math` model, aiming to reduce the training cost of PPO-style (Schulman et al., 2017) reasoning. GRPO removes the value network by estimating advantages from the in-group standardized reward. Concretely, for each query–answer pair $(\boldsymbol{x}, \boldsymbol{y})$, the rollout policy $\pi_{\theta_{\text{old}}}$ samples a group of responses $\{o^i\}_{i=1}^G$ with binary outcome rewards $\{R^i\}_{i=1}^G$ defined by a verifier. Let

$$\hat{A}_t^i = \frac{R^i - \text{mean}(\{R^i\}_{i=1}^G)}{\text{std}(\{R^i\}_{i=1}^G)}, \quad \text{where} \quad R^i = \begin{cases} 1 & \text{if } \texttt{is_equivalent}(a, o^i), \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

GRPO then optimizes a PPO-style clipped objective with an additional KL penalty to a reference model

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o^i\}_{i=1}^G\sim\pi_{\theta\text{old}}(\cdot|q)} \\ \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o^i|}\sum_{t=1}^{|o^i|}\left[\min\left(r_t^i(\theta)\hat{A}_t^i, \text{clip}(r_t^i(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^i\right) - \beta\mathcal{D}_{KL}(\pi_\theta\|\pi_{\text{ref}})\right], \tag{22}$$

where the per-token likelihood ratio is

$$r_t^i(\theta) = \frac{\pi_\theta(o_t^i \mid q, o_{<t}^i)}{\pi_{\theta\text{old}}(o_t^i \mid q, o_{<t}^i)}. \tag{23}$$

$G \in \mathbb{N}$ is the group size, $\epsilon > 0$ is the PPO clipping parameter, $\beta \geq 0$ controls KL regularization, and $\varepsilon \ll 1$ stabilizes the standard deviation.

### A.2 MORE ABOUT TEMPORAL SPARSE AUTOENCODERS.

Sparse Autoencoders (SAEs) decompose dense model activations into sparse latent features. Given an activation $\boldsymbol{h}_t \in \mathbb{R}^d$ at token position $t$, an SAE encodes it into an overcomplete sparse code and reconstructs the original activation:

$$\boldsymbol{z}_t = \zeta(\boldsymbol{h}_t) = \text{TopK}(\sigma(\boldsymbol{W}^{\text{enc}}\boldsymbol{h}_t + \boldsymbol{b}^{\text{enc}})), \qquad \hat{\boldsymbol{h}}_t = \xi(\boldsymbol{z}_t) = \boldsymbol{W}^{\text{dec}}\boldsymbol{z}_t + \boldsymbol{b}^{\text{dec}}, \tag{24}$$

where $\boldsymbol{W}^{\text{enc}} \in \mathbb{R}^{m\times d}$, $\boldsymbol{W}^{\text{dec}} \in \mathbb{R}^{d\times m}$, $\boldsymbol{b}^{\text{enc}} \in \mathbb{R}^m$, $\boldsymbol{b}^{\text{dec}} \in \mathbb{R}^d$, and $m \gg d$. Each coordinate of $\boldsymbol{z}_t$ corresponds to a sparse latent feature, whose value indicates the activation strength of that feature at token position $t$.

Temporal Sparse Autoencoders (T-SAEs) extend standard SAEs by incorporating the sequential structure of language. The key assumption is that high-level semantic features should remain relatively stable across nearby tokens in the same sequence, while low-level syntactic or token-specific features may fluctuate more rapidly. To model this distinction, T-SAE partitions the sparse code into high-level and low-level components:

$$\boldsymbol{z}_t = \left[\boldsymbol{z}_t^{0:h}, \boldsymbol{z}_t^{h:m}\right], \tag{25}$$

where $\boldsymbol{z}_t^{0:h} \in \mathbb{R}^h$ denotes the first $h$ high-level features and $\boldsymbol{z}_t^{h:m} \in \mathbb{R}^{m-h}$ denotes the remaining low-level features. Correspondingly, the decoder matrix is partitioned as

$$\boldsymbol{W}^{\mathrm{dec}} = \left[\boldsymbol{W}_{0:h}^{\mathrm{dec}}, \boldsymbol{W}_{h:m}^{\mathrm{dec}}\right], \tag{26}$$

where $\boldsymbol{W}_{0:h}^{\mathrm{dec}} \in \mathbb{R}^{d\times h}$ and $\boldsymbol{W}_{h:m}^{\mathrm{dec}} \in \mathbb{R}^{d\times(m-h)}$.

The reconstruction objective contains a high-level reconstruction term and a full reconstruction term:

$$\mathcal{L}_{\mathrm{rec}} = \left\|\boldsymbol{h}_t - \left(\boldsymbol{W}_{0:h}^{\mathrm{dec}}\boldsymbol{z}_t^{0:h} + \boldsymbol{b}^{\mathrm{dec}}\right)\right\|_2^2 + \left\|\boldsymbol{h}_t - \left(\boldsymbol{W}^{\mathrm{dec}}\boldsymbol{z}_t + \boldsymbol{b}^{\mathrm{dec}}\right)\right\|_2^2. \tag{27}$$

The first term encourages high-level features to capture stable semantic structure, while the second term preserves full reconstruction quality using both high- and low-level features.

To encourage temporal consistency, T-SAE further adds a contrastive objective on the high-level features. For paired activations $(\boldsymbol{h}_t^{(i)}, \boldsymbol{h}_{t-1}^{(i)})$ from the same sequence, let

$$\boldsymbol{u}_t^{(i)} = \boldsymbol{z}^{0:h}(\boldsymbol{h}_t^{(i)}), \qquad \boldsymbol{u}_{t-1}^{(i)} = \boldsymbol{z}^{0:h}(\boldsymbol{h}_{t-1}^{(i)}), \tag{28}$$

and let $s(\cdot,\cdot)$ denote cosine similarity. The contrastive loss is

$$\mathcal{L}_{\mathrm{contr}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(s(\boldsymbol{u}_t^{(i)}, \boldsymbol{u}_{t-1}^{(i)})\right)}{\sum_{j=1}^{N}\exp\left(s(\boldsymbol{u}_t^{(i)}, \boldsymbol{u}_{t-1}^{(j)})\right)}. \tag{29}$$

This objective pulls together high-level features from nearby tokens in the same sequence while pushing apart high-level features from different examples, preventing collapse to constant representations.

The full T-SAE objective is

$$\mathcal{L}_{\mathrm{TSAE}} = \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{contr}}\mathcal{L}_{\mathrm{contr}}. \tag{30}$$

In our analysis, this temporal structure is important because reasoning unfolds across multiple generated tokens rather than at isolated positions. T-SAE therefore provides a feature basis better suited for tracking reasoning-feature dynamics during RLVR.

### A.3 More About Difficulty Regimes Used in Experiments.

For training-subset comparisons and stratified analysis, we additionally assign a fixed five-way label directly from the same raw success frequency:

$$d_0(\boldsymbol{x}) = \begin{cases} \text{intractable}, & \widehat{p}_{0,n} = 0, \\ \text{hard}, & 0 < \widehat{p}_{0,n} \le 0.2, \\ \text{frontier-hard}, & 0.2 < \widehat{p}_{0,n} \le 0.5, \\ \text{frontier-easy}, & 0.5 < \widehat{p}_{0,n} \le 0.8, \\ \text{easy}, & 0.8 < \widehat{p}_{0,n} \le 1. \end{cases} \tag{31}$$

The query argument is suppressed on the right-hand side. These are fixed frequency intervals, not equal-size quantile bins, and they are assigned from the counts rather than by merging whole `mix@k` bands. At $n = 1024$, the five regimes correspond to success counts 0, 1–204, 205–512, 513–819, and 820–1024. In particular, the exact center belongs to frontier-hard. The frontier-only training subset combines the two frontier regimes; hard-only excludes the zero-success regime, while easy-only includes the all-success endpoint. The labels remain fixed throughout training. They characterize difficulty relative to the base policy, verifier, and decoding protocol: *intractable* means no observed success within the calibration budget, not a proven zero success probability.

## B More details about Proof

### B.1 Proof of Proposition 1

We condition on the training sample $\boldsymbol{x}$, its realized rollout group $\mathcal{U}_{\boldsymbol{x}}^{\tau}$, and the optimizer state at update $\tau$. Thus, the sampled responses are treated as fixed when differentiating the per-step objective. Recall that

$$\mathcal{G}_\tau(\boldsymbol{x}) = \left.\frac{\partial \mathcal{L}_{\boldsymbol{x},\tau}}{\partial \boldsymbol{u}_\theta(\mathcal{U}_{\boldsymbol{x}}^{\tau})}\right|_{\theta=\theta^\tau} \tag{32}$$

denotes the training signal in logit space, and define the corresponding logit Jacobian as

$$\mathbf{J}_u^\tau(\mathcal{U}) = \left.\frac{\partial \boldsymbol{u}_\theta(\mathcal{U})}{\partial \theta}\right|_{\theta=\theta^\tau}. \tag{33}$$

By the chain rule, the parameter gradient induced by the realized rollout group is

$$\begin{aligned} g_\tau(\boldsymbol{x}) &= \nabla_\theta \mathcal{L}_{\boldsymbol{x},\tau}|_{\theta=\theta^\tau} \\ &= \mathbf{J}_u^\tau(\mathcal{U}_{\boldsymbol{x}}^\tau)^\top \mathcal{G}_\tau(\boldsymbol{x}). \end{aligned} \tag{34}$$

Let

$$\Delta\theta^\tau := \theta^{\tau+1} - \theta^\tau. \tag{35}$$

Under the local update map used in Proposition 1, the parameter update can be written as

$$\Delta\theta^\tau = -\eta_\tau \mathbf{M}_\tau g_\tau(\boldsymbol{x}) = -\eta_\tau \mathbf{M}_\tau \mathbf{J}_u^\tau(\mathcal{U}_{\boldsymbol{x}}^\tau)^\top \mathcal{G}_\tau(\boldsymbol{x}). \tag{36}$$

For SGD, $\mathbf{M}_\tau = \mathbf{I}$; more generally, $\mathbf{M}_\tau$ represents the local preconditioning applied to the gradient.

We next examine how this parameter update changes the prediction at a fixed observation context $o$. A first-order Taylor expansion around $\theta^\tau$ gives

$$\Delta \log \boldsymbol{\pi}_\tau(o) = \left.\frac{\partial \log \boldsymbol{\pi}_\theta(o)}{\partial \theta}\right|_{\theta=\theta^\tau} \Delta\theta^\tau + \mathcal{O}\left(\|\Delta\theta^\tau\|^2\right). \tag{37}$$

Since $\boldsymbol{\pi}_\theta(o) = \mathrm{softmax}(\boldsymbol{u}_\theta(o))$, the Jacobian of the log-probabilities with respect to the logits is

$$\left.\frac{\partial \log \boldsymbol{\pi}_\theta(o)}{\partial \boldsymbol{u}_\theta(o)}\right|_{\theta=\theta^\tau} = \mathbf{I} - \mathbf{1}\boldsymbol{\pi}_{\theta^\tau}(o)^\top = \mathcal{T}_\tau(o). \tag{38}$$

Applying the chain rule once more,

$$\left.\frac{\partial \log \boldsymbol{\pi}_\theta(o)}{\partial \theta}\right|_{\theta=\theta^\tau} = \mathcal{T}_\tau(o)\mathbf{J}_u^\tau(o). \tag{39}$$

Substituting Eq. 36 and Eq. 39 into Eq. 37 yields

$$\begin{aligned} \Delta \log \boldsymbol{\pi}_\tau(o) = &-\eta_\tau \mathcal{T}_\tau(o)\mathbf{J}_u^\tau(o)\mathbf{M}_\tau \mathbf{J}_u^\tau(\mathcal{U}_{\boldsymbol{x}}^\tau)^\top \mathcal{G}_\tau(\boldsymbol{x}) \\ &+ \mathcal{O}\left(\|\Delta\theta^\tau\|^2\right). \end{aligned} \tag{40}$$

Defining

$$\mathcal{K}_\tau(o,\mathcal{U}) = \mathbf{J}_u^\tau(o)\mathbf{M}_\tau \mathbf{J}_u^\tau(\mathcal{U})^\top, \tag{41}$$

we obtain

$$\Delta \log \boldsymbol{\pi}_\tau(o) = -\eta_\tau \mathcal{T}_\tau(o)\mathcal{K}_\tau(o,\mathcal{U}_{\boldsymbol{x}}^\tau)\mathcal{G}_\tau(\boldsymbol{x}) + \mathcal{O}\left(\|\Delta\theta^\tau\|^2\right), \tag{42}$$

which proves Proposition 1. □

Finally, summing the per-step changes over training gives the corresponding cumulative relation

$$\begin{aligned} \log \boldsymbol{\pi}_{\theta^T}(o) - \log \boldsymbol{\pi}_{\theta^0}(o) &= \sum_{\tau=0}^{T-1} \Delta \log \boldsymbol{\pi}_\tau(o) \\ &= -\sum_{\tau=0}^{T-1} \eta_\tau \mathcal{T}_\tau(o)\mathcal{K}_\tau(o,\mathcal{U}_{\boldsymbol{x}_\tau}^\tau)\mathcal{G}_\tau(\boldsymbol{x}_\tau) \\ &\quad + \sum_{\tau=0}^{T-1} \mathcal{O}\left(\|\Delta\theta^\tau\|^2\right). \end{aligned} \tag{43}$$

This expression makes explicit that the cumulative learning dynamics are not determined by the availability of training signals alone. Both the signal $\mathcal{G}_\tau$ and its transfer $\mathcal{K}_\tau$ evolve with the policy over training, so the effect of a sample at one stage of training need not persist, or transfer in the same way, at later stages.

# C Details of the MATH Transfer Demonstration

## C.1 Common Anchor, Source Selection, and Output Readouts

The demonstration uses Qwen2.5-Math-1.5B at one base-policy checkpoint. Four source groups and four fixed responses are selected before computing influence, with one of each from hard, frontier-hard, frontier-easy, and easy. The intractable level is not included. For each source level, three short-prompt candidates receive one complete $G = 8$ rollout group; the group with the shortest total complete context is selected, with stable IDs breaking ties. Probe selection prefers verified-correct responses and then shorter complete contexts. Source and observation samples have disjoint sample IDs, exact-prompt family IDs, and prompt-token sequences; no broader near-duplicate audit is claimed. The four observations come from a base-policy rollout bank and are not selected using their influence scores.

Each source gradient is computed from the clipped GRPO PG loss averaged over all valid response tokens in its complete group, using frozen advantages and old log-probabilities. The selected groups have ratio one and inactive clipping at the anchor. The isolated update uses all model parameters, $\mathbf{M} = \mathbf{I}$, and no KL, entropy, momentum, or weight decay. It is not a replay of historical AdamW training. Table 1 reports the realized source signals. All four groups have mixed rewards, so this particular demonstration contains no zero-signal source.

Table 1: The four frozen source groups. $S_x$ is a logit-space gradient norm under the complete-group token-mean loss. Tokens count valid response positions over the eight responses.

| Source difficulty | $S_x$ | Successful responses | Tokens |
|---|---|---|---|
| Hard | 0.00666 | 2/8 | 2,435 |
| Frontier-hard | 0.00901 | 6/8 | 3,882 |
| Frontier-easy | 0.00860 | 5/8 | 2,664 |
| Easy | 0.00914 | 5/8 | 3,142 |

For a fixed response $r$, let $\mathcal{P}_r$ contain its frozen response-content target positions. The output readout is

$$Y_r(\theta) = \frac{1}{|\mathcal{P}_r|} \sum_{t \in \mathcal{P}_r} \log \pi_\theta(y_{r,t} \mid \boldsymbol{x}_r, \boldsymbol{y}_{r,<t}). \tag{44}$$

Suppressing the common checkpoint index and evaluating all gradients at the anchor, we write $g_x = \nabla_\theta \mathcal{L}_{\boldsymbol{x}}$ and summarize the decomposition by

$$S_x = \|\mathcal{G}(\boldsymbol{x})\|_2, \qquad \kappa_{x,r} = -\frac{\langle \nabla_\theta Y_r, g_x \rangle}{S_x}, \qquad I^{\text{lin}}_{x,r} = \eta S_x \kappa_{x,r}. \tag{45}$$

The signed effective transfer $\kappa_{x,r}$ includes the output readout and the direction of the realized source signal; it is not a kernel norm or cosine similarity. Figure 5 gives all 16 linear predictions. For example, frontier-hard and easy have similar $S_x$ but opposite predicted effects on the frontier-hard response. Finite-step measurements cover only the hard and frontier-hard sources; the other two rows remain predictions.

## C.2 Frozen SAE Readout Construction

The feature experiment reuses the same anchor, frozen selection, and cached source gradients as the output experiment. A demonstration SAE is trained on base-model activations from the same bank of 128 responses. The four probe families are reserved for validation, leaving 60 questions for fitting and four for validation. Activations are taken at `model.layers.13`; the Matryoshka dictionary has 12,288 features and configured training sparsity $k = 32$, with seed 42. The dictionary is frozen before readout selection and differentiation. This small base-only dictionary is used for the local bridge; it does not establish cross-checkpoint feature quality.

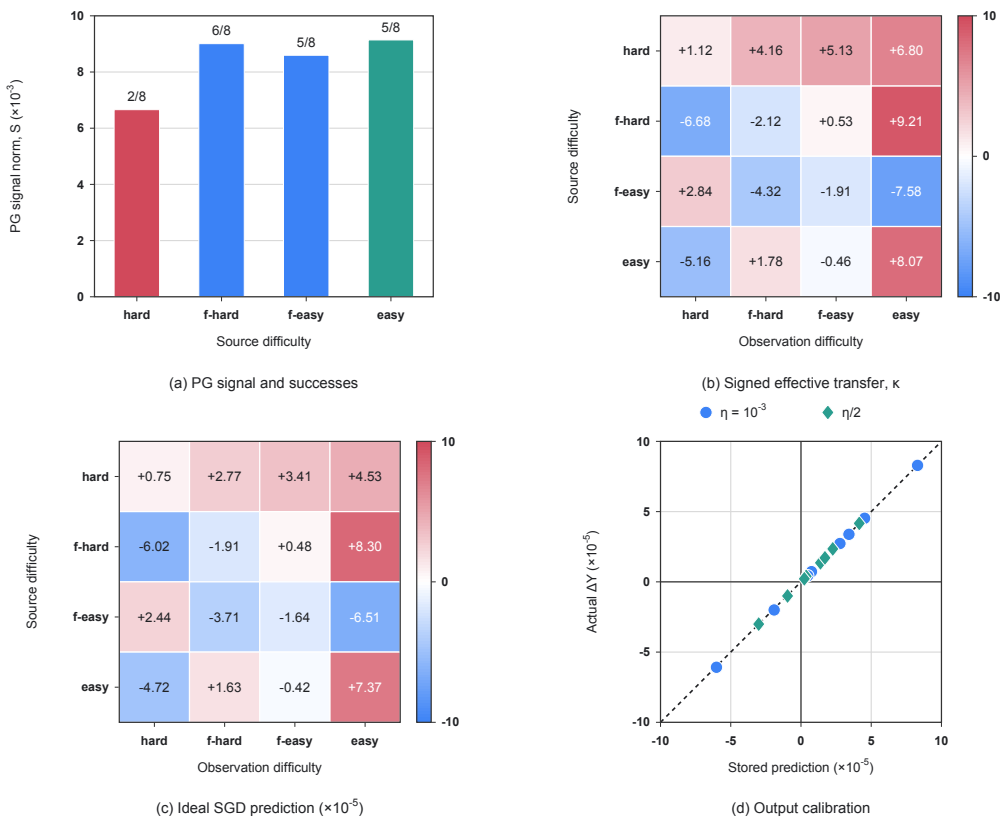



Figure 5: Full output-space diagnostics for the local MATH demonstration. (a) PG signal norm and successful responses out of eight. (b) Signed effective transfer $\kappa_{x,r}$. (c) Ideal SGD prediction $I^{\mathrm{lin}}_{x,r}$ at $\eta = 10^{-3}$. (d) Independent forward calibration of the hard and frontier-hard sources at $\eta$ and $\eta/2$, giving $2 \times 2 \times 4 = 16$ comparisons. Panel (d) predicts with the stored FP32 displacement, $\langle \nabla_\theta Y_r, \Delta\theta_{\mathrm{FP32}} \rangle$; both axes are scaled by $10^{-5}$. Positive values mean increased fixed-response log-probability, not increased generation accuracy.

Using the same target positions $\mathcal{P}_r$, let $z_f(h_\theta(r,t))$ denote the activation of feature $f$ under the frozen dictionary and its inference rule. We select and then freeze

$$f_r = \arg\max_f \frac{1}{|\mathcal{P}_r|} \sum_{t \in \mathcal{P}_r} z_f(h_{\theta^0}(r,t)),$$
$$F_r(\theta) = \frac{1}{|\mathcal{P}_r|} \sum_{t \in \mathcal{P}_r} z_{f_r}(h_\theta(r,t)). \tag{46}$$

The selected features for hard, frontier-hard, frontier-easy, and easy are 725, 156, 112, and 1681, respectively. Each column therefore has its own fixed feature target; the targets are not selected for agreement with output changes and do not define a single common semantic feature across the four responses.

All readouts use fixed teacher-forced text. For target token position $t$, the output log-probability uses the preceding position's logits, while the feature readout uses the hidden state at position $t$, after that token is supplied. Their comparison does not by itself measure a causal contribution of that feature to the same token's probability.

The inference encoder applies a frozen threshold gate to rectified preactivations. Its derivative is local to the active gates at the anchor; the readout is not globally smooth. Threshold crossings were not separately audited in this run, so the empirical finite-step check is necessary and does not establish an unconditional second-order remainder bound for the SAE readout.

On the validation response segments, reconstruction NMSE ranges from 0.242 to 0.325, explained variance from 0.516 to 0.629, and mean activation count from 63.0 to 85.5. These are descriptive measurements on four held-out questions, with two responses per question, rather than a broad assessment of SAE quality. The local readout check below does not establish semantic labels for the selected features or causal mediation of output changes.

### C.3 Finite-Step Calibration and Numerical Resolution

For each of the two calibrated sources and each step size $\eta \in \{10^{-3}, 5\times10^{-4}\}$, we restore the anchor, apply its cached PG gradient, and measure the change of each fixed readout by an independent forward pass. Both $H_r = Y_r$ and $H_r = F_r$ are checked against the ideal SGD prediction $-\eta\langle\nabla_\theta H_r, g_x\rangle$ and the stored-displacement prediction $\langle\nabla_\theta H_r, \Delta\theta_{\mathrm{FP32}}\rangle$. Figures 5(d) and 6 use the latter. Each readout has eight source–observation pairs evaluated at two step sizes, not 16 independent pairs.

Table 2: Calibration against the stored FP32 displacement. Relative error is $|I^{\mathrm{stored}} - \Delta H|/|\Delta H|$; absolute errors are in each readout's own units. Signs agree in all 16 comparisons for each readout.

| Readout | Median relative error | Maximum relative error | Maximum absolute error |
|---|---|---|---|
| Output $Y_r$ | 2.0% | 10.1% | $9.12 \times 10^{-7}$ |
| SAE feature $F_r$ | 1.2% | 70.3% | $1.08 \times 10^{-6}$ |

Predicted and measured changes exceed the configured $10^{-7}$ resolution threshold in all comparisons, and all stored parameter displacements are nonzero. These checks do not impose an approximation-error tolerance. Repeated no-update readouts are exactly repeatable; output calibration baselines match the original anchor baselines, and the SAE readout has zero measured drift after restoration. These checks do not bound numerical error after parameters change. Table 2 therefore reports residuals explicitly rather than treating sign agreement as uniformly accurate magnitude prediction.

For the feature readout, 12 of 16 relative errors are at most 6%. The remaining four are approximately 9.5%, 15.6%, 17.1%, and 70.3%, all for measured changes no larger than $1.54 \times 10^{-6}$ in magnitude. The largest occurs for the frontier-hard source on the easy response at $\eta/2$: the prediction is $4.57 \times 10^{-7}$ and the measured change is $1.54 \times 10^{-6}$. This case retains the predicted sign but gives a weak magnitude approximation. The checks do not isolate the sources of residual error or demonstrate quadratic error scaling with step size.

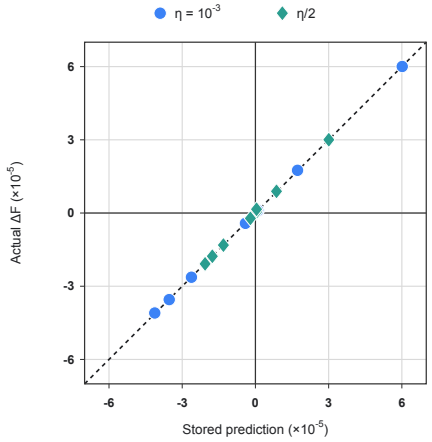



Figure 6: Calibration of the frozen SAE feature readouts using the same two source gradients as the output checks. The horizontal axis uses the stored FP32 displacement; the vertical axis is the independently measured feature change. Each marker is a source–response pair at one of the two step sizes. The near-diagonal appearance at the full plotting scale does not resolve the large relative errors for the smallest effects, reported in Table 2.

All results in this appendix concern selected readouts at one anchor. They do not measure accumulated influence along an RL training trajectory, estimate difficulty-level average effects, or establish that the selected SAE features mediate changes in output or accuracy.

# D More Details about Experiment

## D.1 Difficulty Calibration and Training Setup

We calibrate difficulty on 7,500 MATH training questions using 1,024 base-policy responses per question. We reserve 1,455 questions as a fixed held-out probe and use the remaining 6,045 for training-subset construction. We compare easy-only, frontier-only, and hard-only training, each paired with a natural control matched in the number of distinct questions. Each condition uses three seeds. Frontier-only combines frontier-easy and frontier-hard questions, while hard-only excludes questions with zero calibration successes.

Table 3: Training-subset sizes and nominal exposure, defined as prompt presentations divided by subset size.

| Subset | Questions | Exposure |
|---|---|---|
| Easy | 3,751 | 1.98 |
| Frontier | 1,634 | 4.54 |
| Hard | 660 | 11.25 |

All runs initialize from Qwen2.5-Math-1.5B and use GRPO with eight responses per prompt. Training comprises 116 trainer iterations of 64 prompts each, totaling 7,424 prompt presentations. The learning rate is $3 \times 10^{-6}$, the KL-loss coefficient is $10^{-3}$, and entropy regularization is disabled. Prompt and response lengths are capped at 1,024 and 3,072 tokens, respectively. Trainer iterations are distinct from actor optimizer steps; generated-token and optimization-token budgets may differ across conditions because response lengths vary.

# E Additional Examples of Harmful Hard Samples

We provide additional representative examples of harmful hard samples. Example 2.2 shows misunderstanding of the target goal, where the model treats a make-up requirement as a simple cumulative-goal problem. Example 2.3 illustrates a question–answer mismatch, where the provided label corresponds to the daily cost rather than the total cost requested by the question. Example 2.4 demonstrates missing aggregation operations in the final answer. Example 2.5 shows a Python calculation error that contradicts the correct arithmetic. Example 2.6 captures inconsistent output value types, where the model alternates between integer and decimal monetary formats. Example 2.7 presents a logical error in combining multiple constraints.

**Example 2.2: Misunderstanding the Goal**

```
Question: Shawna's workout goal is 30 situps. On Monday, Shawna was only able to do 12 situps,
      so she decided that she would make up for the rest on Tuesday. However, she was only
     able to do 19 situps on Tuesday. How many situps would Shawna have to do on Wednesday to
     meet her minimum goal and make up for the ones she didn't do? (Answer: 59)
Response: Shawna's goal is to do 30 situps in total. On Monday, she did 12 situps, and on
     Tuesday, she did 19 situps. To find out how many situps she needs to do on Wednesday to
     meet her goal, we can calculate the total number of situps she has already done and then
     subtract that from her goal.

1. Calculate the total number of situps Shawna has done so far:
\[ \text{Total situps done so far} = 12 + 19 = 31 \]

2. Since Shawna's goal is 30 situps, she has already exceeded her goal by 1 situp. Therefore,
     she doesn't need to do any more situps on Wednesday to meet her goal. However, if we
     interpret the problem as finding out how many situps she would need to do on Wednesday to
      make up for the ones she didn't do (which is not possible since she has already exceeded
      her goal), we can still calculate it.
```

````
3. The number of situps Shawna needs to do on Wednesday to make up for the ones she didn't do
     is:
\[ \text{Situps needed on Wednesday} = 30 - (12 + 19) = 30 - 31 = -1 \]

Since it's not possible to do a negative number of situps, the answer is 0. Let's confirm this
      with Python code.
```python
# Shawna's goal
goal = 30

# Situps done on Monday and Tuesday
situps_monday = 12
situps_tuesday = 19

# Total situps done so far
total_done = situps_monday + situps_tuesday

# Situps needed on Wednesday to meet the goal
situps_needed_wednesday = goal - total_done

# Since it's not possible to do a negative number of situps, we set it to 0
if situps_needed_wednesday < 0:
    situps_needed_wednesday = 0

# Output the final answer
print(f"\boxed{{{situps_needed_wednesday}}}")
```
```output
\boxed{0}
```
The final answer is \(\boxed{0}\).
````

**Example 2.3:** ***Question Mismatches Answer***

````
Question: Henry took 9 pills a day for 14 days. Of these 9 pills, 4 pills cost $1.50 each, and
      the other pills each cost $5.50 more. How much did he spend in total on the pills? (
     Answer: 41)
Response: First, let's break down the problem:

1. Henry takes 9 pills a day for 14 days, so the total number of pills he takes is \(9 \times
     14 = 126\).
2. Out of these 9 pills, 4 pills cost $1.50 each, and the other 5 pills cost $1.50 + $5.50 =
     $7.00 each.
3. The total cost for the 4 pills is \(4 \times 1.50 = 6.00\).
4. The total cost for the 5 pills is \(5 \times 7.00 = 35.00\).
5. The total cost for the 9 pills is \(6.00 + 35.00 = 41.00\).
6. Since Henry takes these 9 pills every day for 14 days, the total cost for all the pills is
     \(41.00 \times 14 = 574.00\).

Now, let's write the Python code to verify this calculation.
```python
# Number of pills per day
pills_per_day = 9

# Number of days
days = 14

# Cost of 4 pills
cost_4_pills = 4 * 1.50

# Cost of 5 pills
cost_5_pills = 5 * (1.50 + 5.50)

# Total cost for 9 pills
total_cost_9_pills = cost_4_pills + cost_5_pills

# Total cost for all pills over 14 days
total_cost = total_cost_9_pills * days

# Output the final answer
print(f"\boxed{{{int(total_cost)}}}")
```
```output
\boxed{574}
```
The total amount Henry spent on the pills is \(\boxed{574}\).
````

**Example 2.4: Missing Multiplication or Addition**

**Question:** In a section of the forest, there are 100 weasels and 50 rabbits. Three foxes invade this region and hunt the rodents. Each fox catches an average of 4 weasels and 2 rabbits per week. How many rabbits and weasels will be left after 3 weeks? (**Answer:** 96)
**Response:** 1. First, we need to calculate the number of weasels and rabbits caught by the foxes in one week.
2. Each fox catches 4 weasels and 2 rabbits per week. With 3 foxes, the total number of weasels caught per week is \(3 \times 4 = 12\) weasels, and the total number of rabbits caught per week is \(3 \times 2 = 6\) rabbits.
3. Over 3 weeks, the total number of weasels caught is \(12 \times 3 = 36\) weasels, and the total number of rabbits caught is \(6 \times 3 = 18\) rabbits.
4. The initial number of weasels is 100, so after 3 weeks, the number of weasels left is \(100 - 36 = 64\).
5. The initial number of rabbits is 50, so after 3 weeks, the number of rabbits left is \(50 - 18 = 32\).
6. The final answer should be boxed.

Let's implement this in Python to ensure the calculations are correct.
```python
# Initial number of weasels and rabbits
initial_weasels = 100
initial_rabbits = 50

# Number of foxes and their catch rates
foxes = 3
weasels_per_fox_per_week = 4
rabbits_per_fox_per_week = 2

# Total number of weasels and rabbits caught per week
weasels_caught_per_week = foxes * weasels_per_fox_per_week
rabbits_caught_per_week = foxes * rabbits_per_fox_per_week

# Total number of weasels and rabbits caught in 3 weeks
weasels_caught_in_3_weeks = weasels_caught_per_week * 3
rabbits_caught_in_3_weeks = rabbits_caught_per_week * 3

# Number of weasels and rabbits left after 3 weeks
weasels_left = initial_weasels - weasels_caught_in_3_weeks
rabbits_left = initial_rabbits - rabbits_caught_in_3_weeks

# Final answer in boxed format
final_answer = f"\boxed{{{int(weasels_left)} {int(rabbits_left)}}}"
print(final_answer)
```
```output
\boxed{64 32}
```
The final answer is:

\[
\boxed{64 32}
\]

**Example 2.5: Python Calculation Error**

**Question:** A cake has 8 slices and each slice contains 347 calories. A pan of brownies has 6 brownies and each slice contains 375 calories. How many more calories does the cake have ? (**Answer:** 526)
**Response:** First, we need to calculate the total number of calories in the cake and the total number of calories in the pan of brownies. Then, we will find the difference between these two values.

1. Calculate the total number of calories in the cake:
\[ \text{Total calories in cake} = 8 \text{ slices} \times 347 \text{ calories/slice} \]

2. Calculate the total number of calories in the pan of brownies:
\[ \text{Total calories in brownies} = 6 \text{ brownies} \times 375 \text{ calories/brownie} \]

3. Find the difference between the total number of calories in the cake and the total number of calories in the pan of brownies:
\[ \text{Difference} = \text{Total calories in cake} - \text{Total calories in brownies} \]

Let's calculate this using Python code.
```python
# Calculate the total number of calories in the cake
total_calories_cake = 8 * 347
```

```
# Calculate the total number of calories in the pan of brownies
total_calories_brownies = 6 * 375

# Calculate the difference
difference = total_calories_cake - total_calories_brownies

# Output the final answer in boxed format
print(f"\boxed{{{int(difference)}}}")
```
```output
\boxed{14}
```
The cake has \(\boxed{14}\) more calories than the pan of brownies.

**Example 2.6: Different Value Types**

**Question**: Tim buys 3 dozen eggs. Eggs cost $.50 each. How much did he pay for eggs? (**Answer**: 18)
**Response**: First, we need to determine the total number of eggs Tim bought. Since Tim buys 3 dozen eggs, and one dozen is equal to 12, the total number of eggs is \(3 \times 12 = 36\).

Next, we need to calculate the total cost of these eggs. Since each egg costs $0.50, the total cost is \(36 \times 0.50 = 18.00\).

So, Tim paid $18.00 for the eggs. Let's put this in the final answer format.

\boxed{18.00}
```python
# Calculate the total number of eggs
total_eggs = 3 * 12

# Calculate the total cost
total_cost = total_eggs * 0.50

# Output the final answer
print(f"\boxed{{{int(total_cost)}}}")
```
```output
\boxed{18}
```
The total cost of the eggs is $18.00. Therefore, the final answer is:

\[
\boxed{18.00}
\]

**Example 2.7: Logical Error**

**Question**: Amaya scored 20 marks fewer in Maths than she scored in Arts. She also got 10 marks more in Social Studies than she got in Music. If she scored 70 in Music and scored 1/10 less in Maths, what's the total number of marks she scored in all the subjects? (**Answer**: 296)
**Response**: First, let's define the variables for the scores in each subject:
- Let \( A \) be the score in Arts.
- Let \( M \) be the score in Music.
- Let \( S \) be the score in Social Studies.
- Let \( T \) be the score in Maths.

From the problem statement, we know:
1. Amaya scored 20 marks fewer in Maths than she scored in Arts. So, \( T = A - 20 \).
2. She scored 10 marks more in Social Studies than she got in Music. So, \( S = M + 10 \).
3. She scored 70 in Music. So, \( M = 70 \).
4. She scored \( \frac{1}{10} \) less in Maths than in Arts. So, \( T = A - \frac{1}{10}A = \frac{9}{10}A \).

Using the value of \( M \), we can find \( S \):
\[ S = 70 + 10 = 80 \]

Now, using the value of \( T \) and the relationship \( T = \frac{9}{10}A \), we can find \( A \):
\[ A - 20 = \frac{9}{10}A \]
\[ A - \frac{9}{10}A = 20 \]
\[ \frac{1}{10}A = 20 \]
\[ A = 200 \]

````
Now we can find \( T \):
\[ T = 200 - 20 = 180 \]

So, the scores in each subject are:
- Arts: \( A = 200 \)
- Music: \( M = 70 \)
- Social Studies: \( S = 80 \)
- Maths: \( T = 180 \)

The total number of marks she scored in all the subjects is:
\[ 200 + 70 + 80 + 180 = 530 \]

Let's confirm this with Python code.
```python
# Given values and relationships
M = 70  # Score in Music
S = M + 10  # Score in Social Studies
A = (M + 20) * (10 / 9)  # Score in Arts, derived from T = (9/10)A and T = A - 20
T = A - 20  # Score in Maths

# Total marks
total_marks = A + M + S + T

# Output the final answer
print(f"\boxed{{{int(total_marks)}}}")
```
```output
\boxed{530}
```
The total number of marks Amaya scored in all the subjects is \(\boxed{530}\).
````

## F ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or other sensitive personal data. The datasets and experimental materials used in this study were public, and we comply with the applicable terms of use and data usage requirements. To the best of our knowledge, this work does not introduce significant risks related to privacy, security, discrimination, or harmful applications.

## G REPRODUCIBILITY STATEMENT

We have made efforts to facilitate the reproducibility of our results by providing complete methodological and experimental details throughout the paper and supplementary materials. The main paper describes the proposed method, experimental protocol, and evaluation setting, while the appendices provide the details necessary for reproduction, including data construction and difficulty calibration procedures, training and evaluation settings, implementation details, and additional experimental analyses. For the mechanistic analyses, we also provide the definitions, intervention procedures, and analytical derivations required to reproduce the reported results. All experiments are conducted under fixed and explicitly documented protocols, and the supplementary materials provide the additional information needed to independently reproduce our main findings.