SAEPO: SPARSE-AUTOENCODER-GUIDED CREDIT ASSIGNMENT FOR POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has demonstrated strong performance in improving large language model (LLM) reasoning, yet its credit-assignment signal remains coarse. Outcome verifiers provide scalable supervision by judging whether a completed response is correct, but they do not directly indicate which intermediate reasoning tokens contribute to success or failure. As a result, policy optimization can be inefficient, especially when training base models directly from sparse outcome feedback. In this paper, we introduce SAEPO, a **S**parse **A**uto**E**ncoder-guided token-level credit assignment method for outcome-supervised **P**olicy **O**ptimization. Rather than treating correctness only as a rollout-level (sequence-level) signal, SAEPO uses a pretrained sparse autoencoder (SAE) to build token-level semantic representations within reasoning trajectories; it scores each token based on the SAE concepts (e.g., penalizing wrong "final-answer" tokens but rewarding correct "reasoning-process" tokens), learned in an unsupervised manner, to achieve fine-grained credit assignment. SAEPO is a plug-in advantage-shaping module compatible with various policy optimization algorithms. We instantiate SAEPO with both GRPO and GSPO, covering token-ratio and sequence-ratio policy-gradient objectives, and evaluate SAEPO across three LLM backbones on six math and two code benchmarks. Empirical results show that SAEPO improves both the math and code benchmark averages over the matched baselines, with gains of up to 6.63% and 5.57%, respectively. On `Qwen3-1.7B-Base`, training curves also show faster early accuracy growth. Feature and token-level analyses suggest that the SAE-derived signal favors question-relevant concepts and reasoning tokens.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has become an effective paradigm for improving the reasoning ability of Large Language Models (LLMs) on tasks with rule-based correctness signals. In mathematical reasoning (Shao et al., 2024; Team, 2025a; Yu et al., 2025; Wen et al., 2025; Wang et al., 2025d), code generation (Le et al., 2022; Dai et al., 2024; Wang et al., 2025c; Jiang et al., 2026), and theorem proving (Lample et al., 2022; Zhang et al., 2025; Ji et al., 2025), outcome verifiers can judge whether a completed response is correct without requiring human-written reasoning traces. This makes RLVR particularly attractive for training models on challenging reasoning tasks. Recent outcome-supervised policy-gradient methods, including GRPO (Shao et al., 2024) and GSPO (Zheng et al., 2025), have made this recipe practical by updating the model from sampled solutions and verifiable final-answer rewards. Although these methods differ in their policy-ratio and clipping designs, their supervision ultimately arrives at the response level: a completed rollout (sequence) is judged correct or incorrect, and the same resulting scalar learning signal must then be broadcast into distinct token-level policy-gradient updates, as shown in Fig. 1(top).

This broadcast is the source of the credit-assignment bottleneck. The KTAE study by Sun et al. (2025) observes that GRPO- and DAPO-style methods compute rollout-level advantages and assign the same value to every token in the sequence, failing to capture token-specific contributions. OAR similarly describes standard GRPO as propagating group-level rewards uniformly to every token, neglecting the varying contribution of individual reasoning steps (Li et al., 2026). GTPO/GRPO-S frames the same issue as a coarse-grained assignment that applies a uniform reward to all tokens in long-chain reasoning (Tan & Pan, 2025). These observations point to the same limitation: outcome supervision

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.

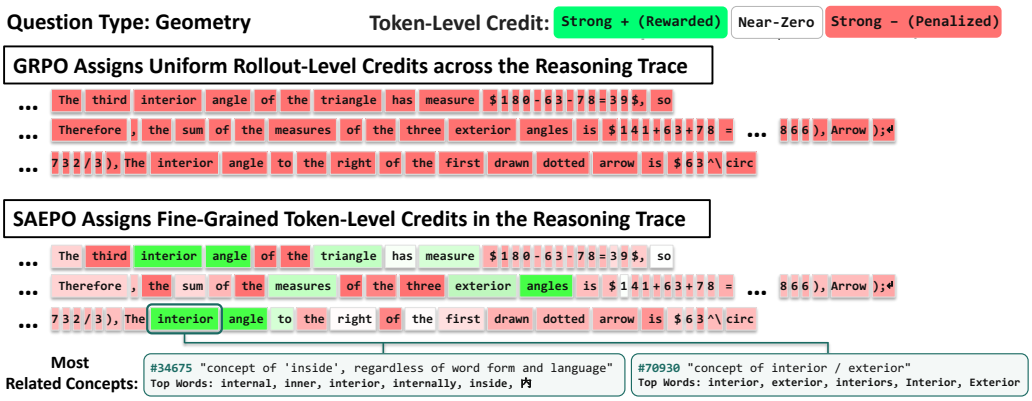

Figure 1: Comparison of credit assignment mechanisms in outcome-supervised policy optimizations. **GRPO** assigns the same rollout-level credit to all tokens in a generated reasoning trace, so decisive reasoning steps and incidental tokens receive identical credit. Our **SAEPO** assigns question-local, token-level semantic credit based on semantic representations from the SAE feature space; therefore it rewards question-relevant reasoning tokens while penalizing uninformative or misleading tokens with more fine-grained credit assignment. Top words of the most related concepts to an example rewarded token "interior" show that our SAEPO can achieve interpretable token-level credit assignment.

separates successful from failed rollouts, but does not explain how credit should be distributed within each rollout.

To obtain finer-grained credit, we need a signal that can distinguish the internal reasoning states of successful and unsuccessful rollouts, rather than relying only on their final scalar outcomes. The model's hidden states provide a natural source of such information, but dense activations are difficult to compare and interpret directly. Sparse Autoencoders (SAEs) (Bricken et al., 2023; Huben et al., 2024; Templeton et al., 2024; Lieberum et al., 2024) offer a scalable way to project dense hidden states into sparse latent features, which often correspond to *more interpretable* and *less noisy* components of model computation. This sparse feature space gives us a semantic interface for RLVR: successful and unsuccessful rollouts of the same question can be contrasted not only by their token identities or final rewards, but by the internal features activated along their reasoning trajectories.

Based on this intuition, it is natural to use the sparse feature space to construct token-level credit signals during policy optimization. We introduce SAEPO, a **S**parse **A**uto**E**ncoder-guided token-level credit assignment method for outcome-supervised **P**olicy **O**ptimization. Rather than treating correctness only as a rollout-level signal, SAEPO compares correct and incorrect rollouts of the same question in semantic space constructed with SAE. Intuitively, in each group of rollouts during GRPO variants, each token's reward depends on how many of its similar tokens are in correct and incorrect rollouts; a token with more similar tokens in correct rollouts than in incorrect rollouts receives a higher reward (more details in Fig. 1(bottom) and a more sophisticated algorithm in Sec. 3.3).

This design makes SAEPO a plug-in for different policy optimization algorithms rather than a new reinforcement learning (RL) objective or reward model. Specifically, the SAE-derived score only reshapes the token-level advantage used during training; the verifier reward, policy ratio, clipping rule, and KL regularizer of the underlying optimizer remain unchanged. As a result, the same SAEPO module can be inserted into different outcome-supervised policy-gradient algorithms, including token-ratio methods such as GRPO (Shao et al., 2024) and sequence-ratio methods such as GSPO (Zheng et al., 2025). SAEPO incurs minimal inference overhead: the SAE is used only during fine-tuning; therefore the final policy requires no SAE computation at inference time.

We instantiate SAEPO with GRPO and GSPO and evaluate it on both mathematical reasoning and coding tasks using `Qwen3-1.7B-Base`, `Qwen3-4B-Base` and `OLMo-3-7B-Base`. Across these settings, SAEPO improves average accuracy on both math and code benchmarks over the corresponding vanilla algorithms and shows improved training efficiency. Further analysis shows that the SAE-derived signal concentrates on semantically meaningful reasoning positions, suggesting that sparse internal representations provide a useful interface for fine-grained credit assignment in outcome-supervised RLVR. Our contributions are summarized as follows:

- We propose SAEPO, a token-level credit-assignment method that uses SAE features to construct question-local contrasts between correct and incorrect rollouts.

- We show that SAEPO can be used as a plug-in advantage-shaping module for different policy optimization algorithms, without modifying the verifier reward, policy ratio, clipping rule, or KL regularizer.
- Interpretability analyses of the SAE-derived credit signal show that SAEPO concentrates on semantically meaningful reasoning positions and highlights question-relevant SAE features, offering a transparent view of how token-level credit is assigned.
- Experiments with different LLM backbones show improved average accuracy and training efficiency over the corresponding base policy optimization algorithms.

2 Related Work

RLVR and Policy Optimization for Reasoning. Reinforcement Learning with Verifiable Rewards (RLVR) has become a central paradigm for improving LLM reasoning on tasks with rule-based correctness signals, including mathematical reasoning, code generation, and theorem proving. Recent systems such as DeepSeek-R1 (Team, 2025a), Kimi k1.5 (Team, 2025b), and Qwen2.5-Math (Yang et al., 2024) demonstrate the effectiveness of outcome-supervised post-training at scale. Algorithmically, GRPO (Shao et al., 2024) replaces PPO's learned critic with a group-normalized baseline computed from multiple rollouts of the same question. Subsequent variants refine different aspects of this recipe: DAPO (Yu et al., 2025) introduces decoupled clipping, dynamic sampling, and token-level loss aggregation; Dr. GRPO (Liu et al., 2025) analyzes length and group-normalization biases; REINFORCE++ (Hu et al., 2025) revisits normalized policy-gradient updates without a learned critic; GSPO (Zheng et al., 2025) moves importance sampling and clipping to the sequence level. These methods differ in their ratio construction, clipping rule, and normalization scheme, but they remain driven by outcome-level correctness signals. In contrast, our work focuses on improving the within-trajectory token-level credit assignment of such outcome-supervised updates.

Fine-Grained Credit Assignment from Outcome Rewards. A growing line of work seeks to extract denser supervision from coarse outcome feedback. Process Reward Models (PRMs) provide step-level feedback using human-labeled or automatically generated supervision (Lightman et al., 2024; Wang et al., 2024; Zhang et al., 2024), but require an auxiliary reward model and additional annotation or search. Other methods infer denser training signals from preference or outcome data: PRIME (Cui et al., 2026) trains an implicit process reward model online, RTO (Zhong et al., 2025) extracts rewards from a DPO-style implicit reward model, and offline preference-optimization methods such as cDPO (Lin et al., 2025), Step-DPO (Lai et al., 2024), TDPO (Zeng et al., 2024), and SVPO (Chen et al., 2024) modify the preference-learning objective to better account for multi-step responses. These methods rely on auxiliary reward modeling, additional supervision, or preference-objective design. SAEPO instead uses a frozen SAE to shape token-level advantages while keeping the verifier reward, policy ratio, and clipping rule unchanged.

A second family constructs training signals directly from sampled rollouts without a separate reward model. VinePPO (Kazemnejad et al., 2025) uses Monte Carlo rollouts from intermediate states, Beyond-80/20 (Wang et al., 2025b) focuses updates on high-entropy forking positions, GTPO (Tan & Pan, 2025) uses entropy and correctness asymmetry, Token Hidden Reward (Deng et al., 2026) reweights advantages by likelihood dynamics, and KTAE (Sun et al., 2025) computes bonuses from correct-versus-incorrect token occurrence statistics. SAEPO differs from these approaches in the signal it uses: it (1) derives fine-grained signals by contrasting rollouts in a sparse semantic space, rather than over token identity or likelihood statistics, and (2) provides concept-level interpretability through its SAE features.

Sparse Autoencoders for Representation-Level Reasoning and Control. Sparse Autoencoders (SAEs) decompose dense LLM activations into sparse latent features and have become a standard tool for mechanistic interpretability (Bricken et al., 2023; Huben et al., 2024; Templeton et al., 2024; Lieberum et al., 2024). Architectural improvements such as Top-K SAEs (Gao et al., 2025), Gated SAEs (Rajamanoharan et al., 2024a), and JumpReLU SAEs (Rajamanoharan et al., 2024b) further improve reconstruction quality and feature sparsity. These developments make SAE features useful not only as post-hoc explanations, but also as structured coordinates for analyzing and manipulating model behavior. This view connects to a broader line of representation-level control, where behavioral directions are obtained by contrasting internal activations from positive and negative examples. Methods such as Contrastive Activation Addition (Rimsky et al., 2024), Inference-Time

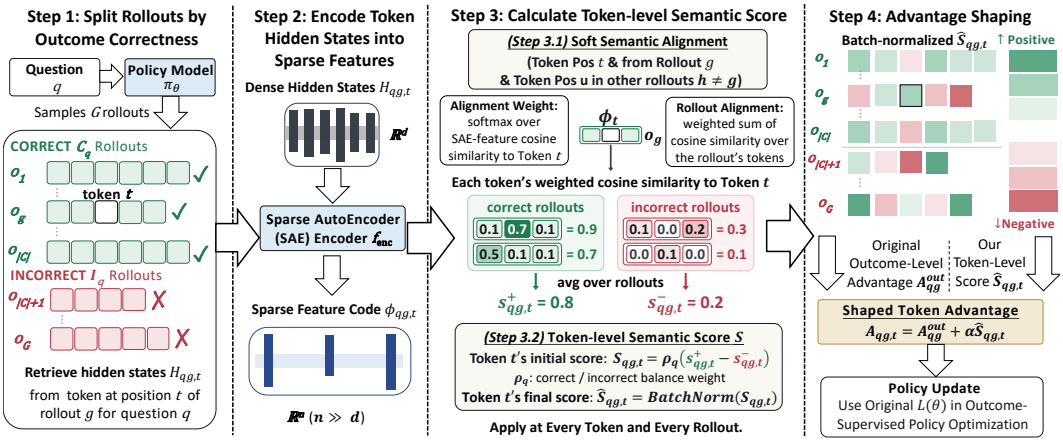

Figure 2: Overview of SAEPO. A frozen SAE encodes token hidden states from grouped rollouts into sparse features. For each token, its counts of similar tokens in correct and incorrect rollouts determine its token-level reward, with higher reward assigned when more similar tokens appear in correct rollouts than in incorrect ones. The token-level reward is normalized and added to the outcome advantage, while the underlying GRPO/GSPO objective remains unchanged.

Intervention (Li et al., 2023), and Representation Engineering (Zou et al., 2023) show that such contrasts can isolate behaviorally meaningful directions in activation space. Recent SAE-based reasoning work applies this idea in sparse feature space: ReasonScore studies SAE features associated with uncertainty, exploration, and reflection (Galichin et al., 2026); SAE-Steering controls reasoning strategies such as backtracking and cross-verification (Fang et al., 2026); and ReSA uses SAE-Tuning to transfer reasoning abilities without reasoning traces (Wang et al., 2025a). SAE features have also been incorporated into alignment and reward-modeling pipelines, including FPO (Yin et al., 2025), SARM (Zhang et al., 2026), SAFER (Shi et al., 2025), and SparseRM (Liu et al., 2026). These methods use SAE features for interpretation, steering, tuning, or reward model training. In contrast, our SAEPO uses SAE to shape token-level advantages during policy optimization.

3 Methodology

3.1 Preliminary: Sparse Autoencoders

Sparse Autoencoders (SAEs) decompose dense model activations into sparse latent features. Given an activation vector $\mathbf{x} \in \mathbb{R}^d$ from a target layer, an SAE encoder maps it to an overcomplete sparse code $\mathbf{f} = f_{\text{enc}}(\mathbf{x}) \in \mathbb{R}^n$, typically with $n \gg d$, and a decoder reconstructs the original activation. Standard SAEs can be written as

$$\mathbf{f} = f_{\text{enc}}(\mathbf{x}) = \sigma(\mathbf{W}_{\text{enc}}(\mathbf{x} - \mathbf{b})), \qquad \hat{\mathbf{x}} = f_{\text{dec}}(\mathbf{f}) = \mathbf{W}_{\text{dec}}\mathbf{f} + \mathbf{b}, \tag{1}$$

where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{n\times d}$, $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{d\times n}$, $\mathbf{b}$ is a centering bias, and σ is an activation or sparsification function. The SAE is trained to reconstruct activations while encouraging sparse codes:

$$\mathcal{L}_{\text{SAE}}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda\,\mathcal{S}(f_{\text{enc}}(\mathbf{x})). \tag{2}$$

Here $\mathcal{S}$ is commonly chosen as an ℓ_1 penalty on the sparse code $\mathbf{f}$. In this work, we use a BatchTopK SAE (Bussmann et al., 2024), which imposes hard sparsity by selecting the largest latent activations across a minibatch. This allows different samples to activate different numbers of features while controlling the average number of active latents. The SAE is trained before policy optimization and kept frozen throughout RL training. We use it only as a sparse representation space for constructing token-level credit signals.

3.2 Preliminary: Outcome-Supervised Policy Optimization

We consider outcome-supervised RL training for reasoning tasks. For each question q, the policy samples G rollouts $\{o_{qg}\}_{g=1}^{G}$, where g indexes the rollout sampled for the same question, and receives a scalar verifier reward r_{qg} for each completed rollout. Let $\mathcal{J}_{qg}$ denote the set of positions of the generated response tokens; this is used in the policy objective for rollout o_{qg}. Accordingly, $|\mathcal{J}_{qg}|$ is the generated response length used for the objective.

Outcome Advantage. The rollout-level outcome advantage is computed by normalizing rewards within each question group, following the group-relative normalization used in GRPO-style RLVR training (Shao et al., 2024):

$$A_{qg}^{\text{out}} = \begin{cases} 0, & \sigma_q^r < \epsilon_{\text{std}}, \\ \frac{r_{qg} - \mu_q^r}{\sigma_q^r + \epsilon_{\text{std}}}, & \text{otherwise}, \end{cases} \tag{3}$$

where μ_q^r and σ_q^r are the mean and standard deviation of $\{r_{q1}, \ldots, r_{qG}\}$, and ϵ_{std} is a small numerical constant. In typical outcome-supervised training (including GRPO and GSPO), this scalar advantage is broadcast to every generated token: $A_{qg,t} = A_{qg}^{\text{out}}, t \in \mathcal{J}_{qg}$. This provides group-relative credit across rollouts but does not distinguish the contributions of individual tokens within a rollout.

GRPO. GRPO can be viewed as a critic-free variant of PPO-style clipped policy optimization, where the learned value baseline is replaced by group-relative reward normalization (Schulman et al., 2017; Shao et al., 2024). It uses token-level policy ratios in the clipped surrogate:

$$r_{qg,t}(\theta) = \frac{\pi_\theta(o_{qg,t}|q, o_{qg,<t})}{\pi_{\theta_{\text{old}}}(o_{qg,t}|q, o_{qg,<t})}. \tag{4}$$

With the per-token advantage $A_{qg,t}$, the GRPO objective can be written as

$$\begin{aligned} \mathcal{L}_{\text{GRPO}}(\theta) = &- \mathbb{E}_{q,g,t\in\mathcal{J}_{qg}} \left[\min \left(r_{qg,t}(\theta) A_{qg,t}, \text{clip}(r_{qg,t}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}) A_{qg,t} \right) \right] \\ &+ \beta \, \text{KL}[\pi_\theta \| \pi_{\text{ref}}] \, . \end{aligned} \tag{5}$$

GSPO. GSPO replaces token-level importance ratios with a length-normalized sequence-level likelihood ratio and applies clipping at the sequence level (Zheng et al., 2025). For rollout o_{qg}, the sequence ratio is

$$s_{qg}(\theta) = \exp\left(\frac{1}{|\mathcal{J}_{qg}|} \sum\nolimits_{t\in\mathcal{J}_{qg}} \log \frac{\pi_\theta(o_{qg,t} \mid q, o_{qg,<t})}{\pi_{\theta_{\text{old}}}(o_{qg,t} \mid q, o_{qg,<t})} \right). \tag{6}$$

With a rollout-level advantage A_{qg}, the original GSPO surrogate can be written as

$$\mathcal{L}_{\text{GSPO}}(\theta) = - \mathbb{E}_{q,g} \left[\min \left(s_{qg}(\theta) A_{qg}, \text{clip}\big(s_{qg}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}\big) A_{qg} \right) \right] + \beta \, \text{KL}[\pi_\theta \| \pi_{\text{ref}}] \, . \tag{7}$$

Since SAEPO supplies token-specific advantages $A_{qg,t}$, we use the token-wise GSPO variant introduced for advantage customization (Zheng et al., 2025). This variant preserves the same sequence-level ratio value and clipping condition, but writes the loss over token positions so that $A_{qg,t}$ can be substituted directly. We provide the full token-wise GSPO surrogate in Appendix A.1.

In vanilla GRPO and GSPO, $A_{qg,t} = A_{qg}^{\text{out}}$ for all $t \in \mathcal{J}_{qg}$. In SAEPO, we replace it with a shaped token-level advantage, while leaving the verifier reward, policy-ratio construction, clipping rule, and KL regularizer unchanged.

3.3 Plug-in Outcome-Supervised Policy Optimization with SAEPO

Our SAEPO is a plug-in advantage-shaping module for outcome-supervised policy optimization. Fig. 2 shows an overview of SAEPO. It constructs a question-local token-level semantic signal from SAE features of the current policy's rollouts, then adds the normalized signal to the standard outcome advantage. The verifier reward, policy-ratio construction, clipping rule, and KL regularizer of the underlying optimizer are left unchanged.

Step 1: Split Rollouts by Outcome Correctness. For each question q, the current policy samples G rollouts $\{o_{qg}\}_{g=1}^{G}$, where g indexes the rollout. After grading the final answer of each rollout, we split the group into correct and incorrect rollout sets:

$$y_{qg} = \mathbf{1}\{\text{the final answer of } o_{qg} \text{ is correct}\}, \qquad \mathcal{C}_q = \{g : y_{qg} = 1\}, \quad \mathcal{I}_q = \{g : y_{qg} = 0\}. \tag{8}$$

We use this split to construct the SAE feature contrast. If either $\mathcal{C}_q$ or $\mathcal{I}_q$ is empty, the question has no correct-versus-incorrect contrast and receives no SAE signal for that update.

Step 2: Encode Token Hidden States into Sparse Features. For each generated-token position $t \in \mathcal{J}_{qg}$, let $H_{qg,t} \in \mathbb{R}^d$ denote the target-layer hidden state corresponding to the policy-loss token. We encode this hidden state with the pretrained SAE:

$$\phi_{qg,t} = f_{\text{enc}}(H_{qg,t}) \in \mathbb{R}^n, \tag{9}$$

which generates SAE features for later semantic alignment.

Step 3.1: Semantic Alignment across Rollouts. The core of our method is comparing each token within a rollout against other rollouts of its own question group, rather than unrelated questions. Since different rollouts of the same question may reach the same intermediate result at different positions, we do not require tokens to align by position. For token t in rollout g of question q, we compare its SAE feature with those of every generated token u in another rollout $h \neq g$ of the same question. A softmax with temperature $\tau > 0$ assigns larger weights to tokens with higher cosine similarity. The weighted sum of these similarities gives one score for rollout h:

$$\omega_{qg,t,u}^{(h)} = \frac{\exp\Big(\cos(\phi_{qg,t},\phi_{qh,u})/\tau\Big)}{\sum_{u' \in \mathcal{J}_{qh}} \exp\Big(\cos(\phi_{qg,t},\phi_{qh,u'})/\tau\Big)}, \quad m_{qg,t}^{(h)} = \sum\nolimits_{u \in \mathcal{J}_{qh}} \omega_{qg,t,u}^{(h)} \cos(\phi_{qg,t}, \phi_{qh,u}). \tag{10}$$

Repeating this comparison for each rollout $h \neq g$ yields one score $m_{qg,t}^{(h)}$ for every other rollout h. We then average these scores separately over the correct and incorrect rollouts:

$$s_{qg,t}^{+} = \frac{1}{|\mathcal{C}_q \setminus \{g\}|} \sum\nolimits_{h \in \mathcal{C}_q \setminus \{g\}} m_{qg,t}^{(h)}, \qquad s_{qg,t}^{-} = \frac{1}{|\mathcal{I}_q \setminus \{g\}|} \sum\nolimits_{h \in \mathcal{I}_q \setminus \{g\}} m_{qg,t}^{(h)}. \tag{11}$$

We exclude g to avoid matching token t to itself. The resulting $s_{qg,t}^{+}$ and $s_{qg,t}^{-}$ measure its average similarity to correct and incorrect rollouts of the same question, respectively. If either set in Eqn. (11) is empty, rollout g receives no SAE score.

Step 3.2: Calculate Token-Level Semantic Score. We then compute the difference of two alignment scores from Step 3.1 and scale it by the balance of correct and incorrect rollouts for question q:

$$S_{qg,t} = \rho_q \left[s_{qg,t}^{+} - s_{qg,t}^{-} \right], \qquad t \in \mathcal{J}_{qg}, \tag{12}$$

We call $S_{qg,t}$ the *semantic score*. Here $\rho_q = 2\min(|\mathcal{C}_q|, |\mathcal{I}_q|)/(|\mathcal{C}_q| + |\mathcal{I}_q|)$ downweights imbalanced correct/incorrect splits. A positive $S_{qg,t}$ means more token t's similar tokens are in correct rollouts than in incorrect rollouts, and therefore token t aligns more strongly with correct than incorrect rollouts of the same question; a negative value means the reverse.

Step 4: Advantage Shaping with Token-Level Semantic Score. The raw semantic scores have batch-dependent scale, so we standardize them before adding them to the outcome advantage. Specifically, for each update batch, we compute the mean μ_S and standard deviation σ_S over all defined raw scores $S_{qg,t}$. For these tokens, we set

$$\widehat{S}_{qg,t} = \frac{S_{qg,t} - \mu_S}{\sigma_S + \epsilon_{\text{score}}},$$

where ϵ_{score} is a small numerical constant for stability. For unscored tokens, we set $\widehat{S}_{qg,t} = 0$. If no scores are defined in a batch, all tokens receive $\widehat{S}_{qg,t} = 0$. The final shaped advantage is

$$A_{qg,t} = A_{qg}^{\text{out}} + \alpha\, \widehat{S}_{qg,t}, \qquad t \in \mathcal{J}_{qg}, \tag{13}$$

where α controls the strength of the SAE signal. When $\alpha = 0$, the update reduces to the vanilla outcome-only update. Tokens in unscored rollouts also use the outcome-only advantage.

Plug-in Policy Optimization. We instantiate SAEPO as GRPO + SAEPO and GSPO + SAEPO by substituting Eqn. (13) into the corresponding objective (Eqn. (5) and Eqn. (7)). In both cases, the only change from the vanilla optimizer is replacing the broadcast *rollout-level* (sequence-level) advantage A_{qg}^{out} with the shaped *token-level* advantage $A_{qg,t}$. The verifier reward, policy-ratio construction, clipping rule, and KL regularizer are unchanged. The SAE is used only during fine-tuning to compute $\widehat{S}_{qg,t}$, and the final policy requires no SAE computation at inference time.

4 Experiments

In this section, we evaluate whether SAEPO's token-level credit improves outcome-supervised RLVR.

4.1 Experimental Setup

Training Data and Reward. For mathematical reasoning, all models are trained on the `DeepScaleR-Preview-Dataset` (Luo et al., 2025b), which contains around 40K math

Table 1: Accuracy (%) on six math and two code benchmarks across three LLM backbones, with unweighted averages. **Bold** marks the best result within each backbone's GRPO or GSPO block.

		Math							Code		
Backbone	**Method**	**MATH-500**	**GSM8K**	**AIME25**	**AMC**	**Minerva**	**Olympiad**	**Avg.**	**LCB-v5**	**LeetCode**	**Avg.**
Qwen3-1.7B-Base	Vanilla GRPO	63.2	74.9	**3.3**	26.5	23.9	21.5	35.56	4.66	**12.28**	8.47
	GRPO + KTAE	61.6	76.1	0.0	22.9	27.6	20.2	34.73	5.38	11.84	8.61
	GRPO + THR	62.8	**76.4**	**3.3**	27.7	26.8	**23.3**	36.72	5.73	7.02	6.38
	GRPO + GTPO	45.2	74.0	**3.3**	13.3	19.9	9.3	27.50	3.58	4.82	4.20
	GRPO + SAEPO (Ours)	**65.6**	76.3	**3.3**	**34.9**	**29.8**	22.1	**38.68**	**6.81**	11.40	**9.11**
	Vanilla GSPO	62.8	76.9	6.7	27.7	23.9	21.8	36.63	**5.73**	9.65	7.69
	GSPO + KTAE	60.4	75.0	0.0	22.9	27.2	19.3	34.13	4.66	10.09	7.37
	GSPO + THR	64.6	77.0	**10.0**	24.1	26.5	20.6	37.12	4.66	8.33	6.50
	GSPO + GTPO	36.8	73.2	0.0	9.6	17.6	7.3	24.09	1.43	7.02	4.23
	GSPO + SAEPO (Ours)	**65.8**	**78.7**	6.7	**32.5**	**30.5**	**23.1**	**39.55**	5.02	**11.40**	**8.21**
Qwen3-4B-Base	Vanilla GRPO	77.6	89.5	**10.0**	44.6	36.0	32.9	48.43	**13.26**	10.96	12.11
	GRPO + KTAE	75.6	88.5	3.3	45.8	**41.2**	32.0	47.74	**13.26**	12.72	12.99
	GRPO + THR	77.0	88.9	6.7	42.2	32.0	33.7	46.74	11.47	11.40	11.44
	GRPO + GTPO	53.2	87.9	0.0	26.5	24.6	12.9	34.19	6.81	5.26	6.04
	GRPO + SAEPO (Ours)	**78.0**	**90.8**	**10.0**	**47.0**	39.7	**34.3**	**49.97**	12.54	**22.81**	**17.68**
	Vanilla GSPO	77.0	90.0	6.7	**43.4**	36.4	32.9	47.73	12.90	13.16	13.03
	GSPO + KTAE	73.2	89.3	**10.0**	39.8	39.0	30.6	46.97	11.47	8.33	9.90
	GSPO + THR	74.4	88.8	**10.0**	42.2	34.2	33.8	47.23	11.11	12.28	11.70
	GSPO + GTPO	47.6	85.7	0.0	18.1	23.5	12.2	31.17	6.81	5.26	6.04
	GSPO + SAEPO (Ours)	**78.4**	**90.1**	**10.0**	**43.4**	**43.8**	**35.2**	**50.15**	**14.34**	**18.86**	**16.60**
OLMo-3-7B-Base	Vanilla GRPO	55.2	86.6	0.0	25.3	24.6	20.0	35.29	8.60	7.89	8.25
	GRPO + KTAE	62.0	86.6	3.3	28.9	**29.4**	22.3	38.75	8.96	7.89	8.43
	GRPO + THR	60.0	87.3	0.0	28.9	23.2	22.4	36.96	**9.68**	7.89	8.79
	GRPO + GTPO	41.0	85.4	0.0	18.1	19.9	9.5	28.98	4.30	3.95	4.12
	GRPO + SAEPO (Ours)	**65.4**	**88.3**	**6.7**	**37.3**	26.8	**27.0**	**41.92**	**9.68**	**10.96**	**10.32**
	Vanilla GSPO	67.2	**86.0**	**10.0**	**36.1**	25.0	26.0	41.71	**11.11**	9.21	10.16
	GSPO + KTAE	63.6	85.5	6.7	32.5	25.7	26.7	40.13	8.60	8.33	8.47
	GSPO + THR	60.8	85.5	**10.0**	27.7	25.0	20.5	38.25	9.68	8.33	9.01
	GSPO + GTPO	50.4	84.8	0.0	21.7	18.8	9.6	30.88	4.66	5.70	5.18
	GSPO + SAEPO (Ours)	**68.6**	85.6	6.7	**36.1**	**30.1**	**31.3**	**43.07**	9.68	**14.47**	**12.08**

problem-answer pairs from competition-style sources such as AIME, AMC, Omni-MATH, and STILL. For each question, the policy samples a group of rollouts and receives an outcome reward from the final answer. The reward combines a binary correctness score from `math_verify` with a small format bonus for producing a boxed answer. For code generation, we train on the TACO, PrimeIntellect SYNTHETIC-1, and LiveCodeBench-v5 training splits of the `DeepCoder-Preview-Dataset` (Luo et al., 2025a).

Backbones and Baselines. We use `Qwen3-1.7B-Base` and `Qwen3-4B-Base` (Team, 2025c) and `OLMo-3-7B-Base` (Olmo et al., 2025) as policy backbones. For each backbone, we compare GRPO + SAEPO against vanilla **GRPO**, and GSPO + SAEPO against vanilla **GSPO**. We also include three state-of-the-art token-level credit-assignment methods as baselines: **KTAE** (Sun et al., 2025), **THR** (Deng et al., 2026), and **GTPO** (Tan & Pan, 2025).

Evaluation. We evaluate accuracy on six mathematical reasoning benchmarks: MATH-500 (Lightman et al., 2023), GSM8K (Cobbe et al., 2021), AIME25 (Zhang & Math-AI, 2025), AMC (Li et al., 2024), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). For code generation, we report LiveCodeBench-v5 (Jain et al., 2025; Luo et al., 2025a) and LeetCode Dataset v0.3.1 (Xia et al., 2025) (228 held-out test problems), executing every unit test and counting a problem as solved only if all tests pass. More implementation details are in Appendix B and Appendix C.

4.2 Results

Main Results. Table 1 compares SAEPO with vanilla GRPO and GSPO, as well as three credit-assignment baselines, i.e., KTAE, THR, and GTPO, across all three LLM backbones and both task types. On `Qwen3-1.7B-Base`, SAEPO successfully enhances both policy optimization algorithms, improving GRPO from 35.56% to 38.68% and GSPO from 36.63% to 39.55%. On `Qwen3-4B-Base`, SAEPO improves the six-benchmark math average from 48.43% to 49.97% with GRPO, and from 47.73% to 50.15% with GSPO. Importantly, SAEPO also consistently outperforms all three credit-assignment baselines in average accuracy across every backbone, policy algorithm, and task type. For example, on `OLMo-3-7B-Base`, SAEPO achieves 41.92% and 43.07% average math accuracy with GRPO and GSPO, respectively, compared with 38.75% and 40.13% for the strongest credit-assignment baseline. The same advantage extends to code generation benchmarks. These

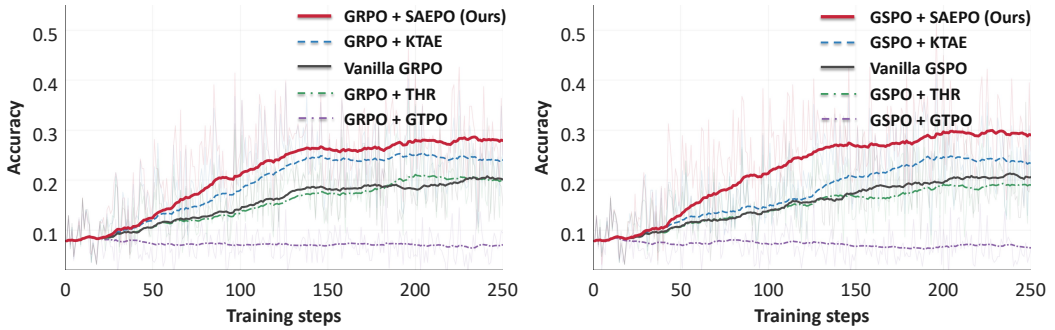

Figure 3: Training dynamics on `Qwen3-1.7B-Base`. We plot training accuracy curves during RL ***training*** for GRPO and GSPO, comparing each vanilla policy optimization algorithm with its SAEPO-augmented variant and with credit-assignment baselines (i.e., KTAE, THR, and GTPO). SAEPO shows faster convergence in training accuracy and generally reaches higher training accuracy.

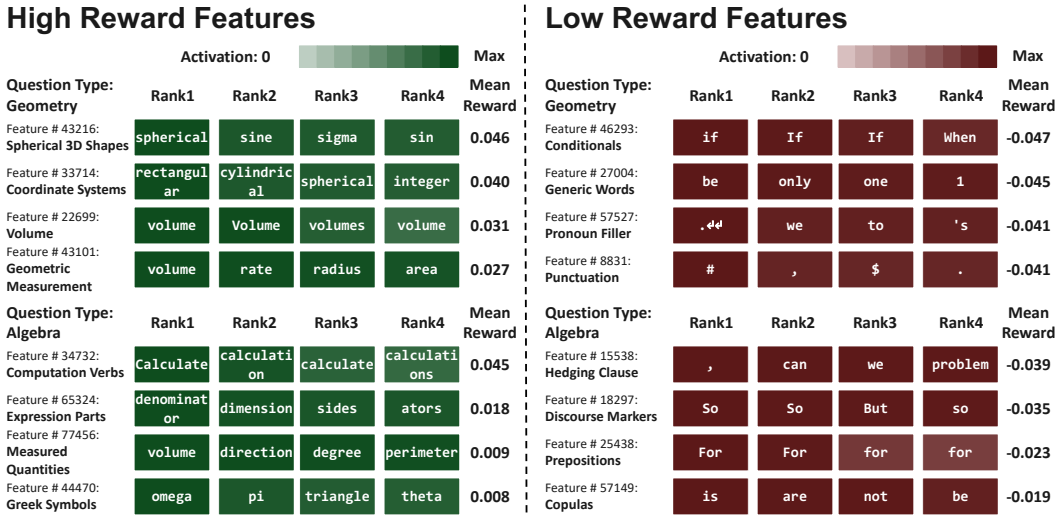

Figure 4: SAE features (concepts) associated with the highest and the lowest SAEPO token scores, computed from rollouts on the "Algebra" and "Geometry" subjects in MATH-500. Feature-level rewards are obtained by averaging semantic scores over tokens where each SAE feature is active. Rows show features with highest/lowest rewards and their top activating tokens; color intensity indicates normalized activation within each feature. **Left:** High-reward features correspond to subject-relevant concepts, such as spherical and coordinate vocabulary in Geometry, and computation and expression terms in Algebra. **Right:** Low-reward features mostly capture connectives, copulas, punctuation, stop words, and discourse markers. These results suggest that the SAEPO-derived signal concentrates on semantically meaningful reasoning features.

results show that SAEPO's token rewards consistently (1) improve the vanilla GRPO/GSPO across all three backbones and both task types, and (2) outperform existing credit-assignment approaches.

Training Efficiency. Fig. 3 compares training accuracy curves on `Qwen3-1.7B-Base`, including vanilla GRPO/GSPO and the credit-assignment baselines KTAE, THR, and GTPO. For both GRPO and GSPO, SAEPO improves early-stage accuracy growth and generally reaches higher accuracy under the same training budget. It also enjoys faster convergence and generally higher training accuracy than the three credit-assignment baselines throughout training. This suggests that the SAEPO-derived *token-level* signal provides a more informative advantage estimate than broadcasting the same *rollout-level* outcome advantage to all generated tokens, while also providing more effective credit assignment than existing token-level alternatives.

Interpretability of the Credit Signal. Fig. 4 visualizes SAEPO's SAE features (concepts) that have the highest and lowest average token-level semantic scores on Algebra and Geometry rollouts from MATH-500. For each top feature, we list its top-4 tokens with the highest activation in the SAE. High-reward features correspond to subject-relevant mathematical concepts, such as computation and expression terms, circle/sphere concepts, and geometric measurements. In contrast, low-reward fea-

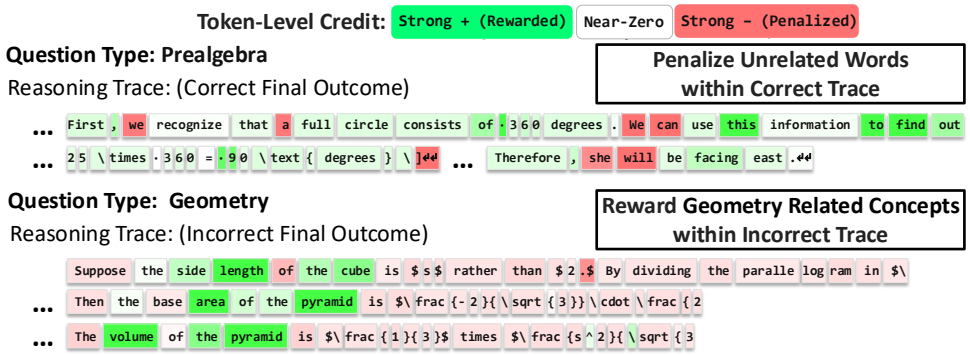

Figure 5: Examples of how SAEPO can assign a fine-grained credit signal to each token. For rollouts from different types of questions, we visualize the SAEPO-derived semantic score assigned to individual tokens in the rollouts. Green tokens represent positive credit that rewards the tokens, gray tokens represent near-zero credit, while red tokens represent negative credit that penalizes the tokens. SAEPO tends to reward *semantically meaningful* tokens like *question-relevant tokens* in incorrect trajectories and *reasoning* tokens in correct trajectories, while penalizing *uninformative* or *misleading tokens* even when they are in correct trajectories, achieving fine-grained token-level credit assignment over the reasoning trace.

Table 2: Ablation studies with `Qwen3-1.7B-Base` on GRPO with SAEPO. We report accuracy (%) on the same six-benchmark evaluation suite as Table 1. **Bold** marks the best result.

Method	MATH-500	GSM8K	AIME25	AMC	Minerva	Olympiad	Avg.
Vanilla GRPO	63.2	74.9	**3.3**	26.5	23.9	21.5	35.56
GRPO + Raw-Embedding Contrast	64.2	74.5	0.0	26.5	30.2	**23.9**	36.55
GRPO + Reverse SAEPO Signal	37.8	75.4	0.0	9.6	22.4	7.6	25.48
GRPO + SAEPO (Ours)	**65.6**	**76.3**	**3.3**	**34.9**	**29.8**	22.1	**38.68**

tures mainly capture connectives, punctuation, generic stop words, copulas, pronouns, and discourse markers. This suggests that the positive credit assigned by SAEPO tends to concentrate on sparse features tied to question-relevant content.

Fig. 5 provides a token-level view of SAEPO. Importantly, SAEPO's signal is not simply a copy of the final rollout outcome. In a correct trace for a Prealgebra problem (Fig. 5(top)), SAEPO still assigns negative scores to unrelated stop words (e.g., "we"), while rewarding tokens involved in the reasoning (e.g., "circle" and "degrees"). Conversely, in an incorrect trace for a geometry question (Fig. 5(bottom)), the rollout receives negative outcome supervision, but some reasoning tokens still contain relevant concepts (e.g., "length", "area", "volume", and "pyramid") and therefore successfully receive positive rewards. This illustrates the main advantage of our SAEPO's token-level signal: it can distinguish useful and unhelpful reasoning components within the same trajectory, while a rollout-level advantage would assign the same credit to all tokens (more examples in Appendix G).

Ablation Studies. Table 2 shows the ablation study results for two core components of SAEPO on `Qwen3-1.7B-Base` with GRPO. Replacing SAE features with raw hidden states hurts performance, indicating that the sparse SAE basis is important for constructing a useful token-level signal. Flipping the sign of the SAE-derived score leads to a large degradation, showing that the success-versus-failure direction carries meaningful training information. The full GRPO + SAEPO variant achieves the best average performance, supporting the design of our method. More ablations are in Appendix E.

5 Conclusion

We introduced SAEPO, a sparse-autoencoder-guided framework for fine-grained token-level credit assignment in outcome-supervised RLVR. By contrasting correct and incorrect rollouts of the same question in a sparse SAE feature space, SAEPO derives question-local semantic signals to reshape token-level advantages without modifying the verifier reward or optimization objective. Across GRPO and GSPO, three LLM backbones, and math and code tasks, SAEPO improves average accuracy over vanilla methods and credit-assignment baselines. Interpretability analyses show that the SAE-derived signal captures meaningful reasoning features. Future work includes extending the framework to other verifiable domains, SAE architectures, and optimization algorithms.

AI USE STATEMENT

We used generative AI tools to improve the wording and readability of the manuscript. The authors reviewed the suggested edits and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

The method, including SAE feature extraction, cross-rollout semantic alignment, and advantage shaping, is specified in Sec. 3. The datasets, rewards, baselines, and evaluation protocol are described in Sec. 4.1, with further training details in Appendix B. Main comparisons use matched training budgets and endpoint checkpoints. We will release our training and evaluation code and final checkpoints upon publication.

REFERENCES

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. URL https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *CoRR*, abs/2412.06410, 2024. doi: 10.48550/ARXIV.2412.06410. URL https://doi.org/10.48550/arXiv.2412.06410.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Step-level value preference optimization for mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7889–7903, 2024.

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.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=9SkkifLopZ.

Ning Dai, Zheng Wu, Renjie Zheng, Ziyun Wei, Wenlei Shi, Xing Jin, Guanlin Liu, Chen Dun, Liang Huang, and Lin Yan. Process supervision-guided policy optimization for code generation. *CoRR*, abs/2410.17621, 2024. doi: 10.48550/ARXIV.2410.17621. URL https://doi.org/10.48550/arXiv.2410.17621.

Wenlong Deng, Yi Ren, Yushu Li, Boying Gong, Danica Sutherland, Xiaoxiao Li, and Christos Thrampoulidis. Token hidden reward: Steering exploration-exploitation in group relative deep reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 122109–122133, 2026.

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.

Andrey V Galichin, Alexey Dontsov, Polina Druzhinina, Anton Razzhigaev, Oleg Rogov, Elena Tutubalina, and Ivan Oseledets. I have covered all the bases here: Interpreting reasoning features in large language models via sparse autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30771–30779, 2026.

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.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025.

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.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Xingguang Ji, Yahui Liu, Qi Wang, Jingyuan Zhang, Yang Yue, Rui Shi, Chenxi Sun, Fuzheng Zhang, Guorui Zhou, and Kun Gai. Leanabell-prover-v2: Verifier-integrated reasoning for formal theorem proving via reinforcement learning. *arXiv preprint arXiv:2507.08649*, 2025.

Xue Jiang, Yihong Dong, Mengyang Liu, Deng Hongyi, Tian Wang, Yongding Tao, Zhi Jin, Wenpin Jiao, and Ge Li. Coderl+: Improving code generation via reinforcement with execution semantics alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3608–3622, 2026.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron C. Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in RL training of llms. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *CoRR*, abs/2406.18629, 2024. doi: 10.48550/ARXIV.2406.18629. URL https://doi.org/10.48550/arXiv.2406.18629.

Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem proving. *Advances in neural information processing systems*, 35:26337–26349, 2022.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu Hong Hoi. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:21314–21328, 2022.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9, 2024.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in neural information processing systems*, 36:41451–41530, 2023.

Ziheng Li, Liu Kang, Feng Xiao, Luxi Xing, Qingyi Si, Zhuoran Li, Weikang Gong, Deqing Yang, Yanghua Xiao, and Hongcheng Guo. Outcome-grounded advantage reshaping for fine-grained credit assignment in mathematical reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24681–24693, 2026.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300, 2024.

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

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/lin25j.html.

Dengcan Liu, Jiahao Li, Zheren Fu, Yi Tu, Jiajun Li, Zhendong Mao, and Yongdong Zhang. Sparserm: A lightweight preference modeling with sparse autoencoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32051–32059, 2026.

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. *CoRR*, abs/2503.20783, 2025. doi: 10.48550/ARXIV.2503.20783. URL https://doi.org/10.48550/arXiv.2503.20783.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level, 2025a. Notion Blog.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025b. Notion Blog.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. Improving dictionary learning with gated sparse autoencoders. *CoRR*, abs/2404.16014, 2024a. doi: 10.48550/ARXIV.2404.16014. URL https://doi.org/10.48550/arXiv.2404.16014.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *CoRR*, abs/2407.14435, 2024b. doi: 10.48550/ARXIV.2407.14435. URL https://doi.org/10.48550/arXiv.2407.14435.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.828. URL https://doi.org/10.18653/v1/2024.acl-long.828.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Wei Shi, Ziyuan Xie, Sihang Li, and Xiang Wang. SAFER: probing safety in reward models with sparse autoencoder. *CoRR*, abs/2507.00665, 2025. doi: 10.48550/ARXIV.2507.00665. URL https://doi.org/10.48550/arXiv.2507.00665.

Wei Sun, Wen Yang, Pu Jian, Qianlong Du, Fuwei Cui, Shuo Ren, and Jiajun Zhang. KTAE: A model-free algorithm to key-tokens advantage estimation in mathematical reasoning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/ed05656150804b030366b68df744950a-Abstract-Conference.html.

Hongze Tan and Jianfei Pan. GTPO and GRPO-S: token and sequence-level reward shaping with policy entropy. *CoRR*, abs/2508.04349, 2025. doi: 10.48550/ARXIV.2508.04349. URL https://doi.org/10.48550/arXiv.2508.04349.

DeepSeek Team. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025a. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599, 2025b. doi: 10.48550/ARXIV.2501.12599. URL https://doi.org/10.48550/arXiv.2501.12599.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025c. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.510. URL https://doi.org/10.18653/v1/2024.acl-long.510.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, Deqing Fu, and Willie Neiswanger. Resa: Transparent reasoning models via saes. *CoRR*, abs/2506.09967, 2025a. doi: 10.48550/ARXIV.2506.09967. URL https://doi.org/10.48550/arXiv.2506.09967.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025b. URL http://papers.nips.cc/paper_files/paper/2025/hash/a797c2d2e0c1fdabf4d1ab8cd0b465c6-Abstract-Conference.html.

Yinjie Wang, Ling Yang, Ye Tian, Ke Shen, and Mengdi Wang. Co-evolving LLM coder and unit tester via reinforcement learning. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025c. URL http://papers.nips.cc/paper_files/paper/2025/hash/d38653cdaa8e992549e1e9e1621610d7-Abstract-Conference.html.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon S. Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025d. URL http://papers.nips.cc/paper_files/paper/2025/hash/b1ea3f93167c55f3fc2999b68a170ff3-Abstract-Conference.html.

Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *CoRR*, abs/2506.14245, 2025. doi: 10.48550/ARXIV.2506.14245. URL https://doi.org/10.48550/arXiv.2506.14245.

Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms, 2025. URL https://arxiv.org/abs/2504.14655.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122, 2024. doi: 10.48550/ARXIV.2409.12122. URL https://doi.org/10.48550/arXiv.2409.12122.

Qingyu Yin, Chak Tou Leong, Hongbo Zhang, Minjun Zhu, Hanqi Yan, Qiang Zhang, Yulan He, Wenjie Li, Jun Wang, Yue Zhang, and Linyi Yang. Constrain alignment with sparse autoencoders. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/yin25a.html.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng,

Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58348–58365. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/zeng24c.html.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: LLM self-training via process reward guided tree search. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/76ec4dc30e9faaf0e4b6093eaa377218-Abstract-Conference.html.

Jingyuan Zhang, Qi Wang, Xingguang Ji, Yahui Liu, Yang Yue, Fuzheng Zhang, Di Zhang, Guorui Zhou, and Kun Gai. Leanabell-prover: Posttraining scaling in formal reasoning. *CoRR*, abs/2504.06122, 2025. doi: 10.48550/ARXIV.2504.06122. URL https://doi.org/10.48550/arXiv.2504.06122.

Shuyi Zhang, Wei Shi, Sihang Li, Jiayi Liao, Tao Liang, Hengxing Cai, and Xiang Wang. Interpretable reward model via sparse autoencoder. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 34808–34816. AAAI Press, 2026. doi: 10.1609/AAAI.V40I41.40783. URL https://doi.org/10.1609/aaai.v40i41.40783.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *CoRR*, abs/2507.18071, 2025. doi: 10.48550/ARXIV.2507.18071. URL https://doi.org/10.48550/arXiv.2507.18071.

Han Zhong, Zikang Shan, Guhao Feng, Wei Xiong, Xinle Cheng, Li Zhao, Di He, Jiang Bian, and Liwei Wang. DPO meets PPO: reinforced token optimization for RLHF. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zhong25b.html.

Andy Zou, Long Phan, Sarah Li Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency. *CoRR*, abs/2310.01405, 2023. doi: 10.48550/ARXIV.2310.01405. URL https://doi.org/10.48550/arXiv.2310.01405.

APPENDIX

A ADDITIONAL METHOD DETAILS

A.1 TOKEN-WISE GSPO SURROGATE

For completeness, we describe the token-wise GSPO variant used in our implementation. Recall that the original GSPO sequence-level ratio for rollout o_{qg} is

$$s_{qg}(\theta) = \exp\left(\frac{1}{|\mathcal{J}_{qg}|}\sum_{t\in\mathcal{J}_{qg}}\log\frac{\pi_\theta(o_{qg,t}\mid q, o_{qg,<t})}{\pi_{\theta_{\text{old}}}(o_{qg,t}\mid q, o_{qg,<t})}\right).$$

To support token-specific advantages, we use the token-wise GSPO construction introduced for advantage customization. Define the effective token-wise weight as

$$\widetilde{s}_{qg,t}(\theta) = \text{sg}[s_{qg}(\theta)]\ \frac{\pi_\theta(o_{qg,t}\mid q, o_{qg,<t})}{\text{sg}[\pi_\theta(o_{qg,t}\mid q, o_{qg,<t})]}, \tag{14}$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operator. Numerically, $\widetilde{s}_{qg,t}(\theta) = s_{qg}(\theta)$, so the clipping condition is still determined by the sequence-level ratio. However, the gradient of each token loss term flows through the corresponding token log-probability, allowing a token-specific advantage $A_{qg,t}$.

The token-wise GSPO surrogate is

$$\begin{aligned}\mathcal{L}_{\text{GSPO}}(\theta) = &-\mathbb{E}_{q,g,t\in\mathcal{J}_{qg}}\left[\min\left(\widetilde{s}_{qg,t}(\theta)A_{qg,t}, \text{clip}\left(\widetilde{s}_{qg,t}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}}^{\text{high}}\right)A_{qg,t}\right)\right]\\ &+\beta\,\text{KL}[\pi_\theta\|\pi_{\text{ref}}]\,.\end{aligned} \tag{15}$$

When all tokens in a rollout share the same advantage, this formulation preserves the original GSPO sequence-level ratio value and clipping behavior. In SAEPO, we substitute the shaped token-level advantage from Eq. (13) into Eq. (15).

Clip values used in this paper. Because the GSPO ratio is sequence-level and length-normalized, its admissible clip range is much tighter than GRPO's. Following the GSPO paper, all GSPO runs in this paper use asymmetric clipping $\epsilon_{\text{clip}} = 3\times10^{-4}$ and $\epsilon_{\text{clip}}^{\text{high}} = 4\times10^{-4}$; GRPO uses the symmetric $\epsilon_{\text{clip}} = 0.2$ standard. The KL coefficient $\beta = 0.02$ and reference policy π_{ref} are shared across all algorithms and arms.

B TRAINING DETAILS

B.1 TRAINING DATA

For math, RL runs use the `DeepScaleR-Preview-Dataset` (Luo et al., 2025b), loaded from `agentica-org/DeepScaleR-Preview-Dataset`. Its training split contains 40,315 math problems from competition-style sources, including AIME, AMC, Omni-MATH, and STILL. We do not apply subject filtering, a difficulty curriculum, or additional subsampling. Matched methods use the same training pool and data order.

For code generation, we use the TACO, PrimeIntellect SYNTHETIC-1, and LiveCodeBench-v5 training splits of the `DeepCoder-Preview-Dataset` (Luo et al., 2025a). These splits provide executable unit tests, and matched methods use the same problems.

Disjoint from evaluation. The DeepScaleR training pool is disjoint from the six-benchmark math evaluation suite. Its internal AIME content is restricted to historical (pre-2024) competitions, so it is also disjoint from AIME25.

B.2 Prompt and Generation Settings

For math, training rollouts and downstream evaluation use the same MINERVA_MATH prompt template. The demonstrations are three worked-out math problems plus the target problem, in the format

```
Problem:\n{example_1}\n\nSolution:\n{worked_solution_1}\nFinal
Answer:  ...\n\n(×3)Problem:\n{problem}\n\nSolution:\n
```

where each demonstration solution terminates in a `\boxed{...}` expression so the model copies that surface form. The string `Problem:` is registered as a vLLM stop token so a generation cannot leak into a hallucinated next problem; the model EOS token `<|endoftext|>` is the other stop signal.

All runs sample $G = 8$ rollouts per question at temperature $T = 1.0$; evaluation uses greedy decoding. The maximum generation length is 8192 tokens for math and 4096 for code. For the Qwen math runs, we use vLLM 0.16 with top-$p = 1.0$, maximum prompt length 1280, and no length penalty. Weights are synchronized from the HuggingFace policy to vLLM every 10 training steps via CuMemAllocator wake/sleep, with `gpu_memory_utilization` set to 0.50 for 1.7B and 0.85 for 4B.

B.3 Reward Construction

For math, the scalar outcome reward is $R_{\text{out}} = R_{\text{acc}} + R_{\text{fmt}}$, where $R_{\text{acc}} \in \{0, 1\}$ records final-answer correctness and $R_{\text{fmt}} \in \{0, 0.1\}$ rewards a boxed answer. We extract the final boxed answer and grade it with `math_verify`. For code, $R_{\text{out}} \in \{0, 1\}$ is one only when every unit test for the problem passes.

The GRPO/GSPO outcome advantage A_{qg}^{out} is computed from R_{out}. For SAEPO, the correct/incorrect split uses R_{acc} for math and test-suite success for code; a math rollout with only the format bonus remains incorrect. After group normalization of the outcome reward and batch standardization of the semantic score, only the advantage changes, as in Eqn. (13).

B.4 Optimization Hyperparameters

Within each matched vanilla/SAEPO pair, the verifier reward, policy-ratio construction, clipping rule, KL regularizer, sampling configuration, and optimizer hyperparameters are identical. SAEPO replaces the broadcast rollout-level advantage A_{qg}^{out} with the token-level advantage $A_{qg,t}$ from Eqn. (13).

All runs use full-parameter fine-tuning, without LoRA or adapters. We use `torch.optim.AdamW` with $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0, and numerical constant $\epsilon = 10^{-8}$. The learning rate is 10^{-5} on a constant schedule (no warmup, no decay). We use KL coefficient $\beta = 0.02$ against a frozen reference policy π_{ref} (the SFT initialization), bf16 mixed precision, gradient checkpointing, micro-batch size 4, and 4 gradient-accumulation steps. Randomness is controlled with seed 42 for Python, NumPy, PyTorch, vLLM, and rollout sampling.

The Qwen3-4B and OLMo-3-7B runs use 8 questions per batch, corresponding to 64 rollouts per training step. The Qwen3-1.7B runs use 32 questions per batch, corresponding to 256 rollouts per training step. The 4B and 7B runs are trained for 1000 steps and the 1.7B runs for 250 steps. Both settings therefore consume the same budget of $64{,}000$ rollouts, so results across model scales are compared at matched sampling cost. All main results are reported at the last checkpoint rather than selecting the best checkpoint independently for each method.

Single-GPU compute. Each math run occupies a single NVIDIA RTX PRO 6000 Blackwell ($\sim$97 GB HBM); vLLM, the HF policy, π_{ref}, and the SAE all share that GPU through CuMemAllocator wake/sleep. End-to-end wall-clock is approximately 80 hours for a 7B run, 36 hours for a 4B run and 14 hours for a 1.7B run. Training, evaluation, and SAE encoding are all single-GPU and require no inter-node communication.

B.5 Sparse Autoencoder Details

For each backbone, we train a BatchTopK SAE (Bussmann et al., 2024) on target-layer hidden activations before RL. During RL, the frozen encoder provides $\phi_{qg,t} = f_{\text{enc}}(H_{qg,t})$ for cross-rollout semantic alignment in Eqn. (10); the SAE is not used at inference time.

For `Qwen3-4B-Base`, the SAE uses layer 27, hidden dimension 2560, a $32\times$ expansion, and 81,920 latent features. For `Qwen3-1.7B-Base`, it uses layer 21, hidden dimension 2048, a $32\times$ expansion, and 65,536 latent features. The target average sparsity is $k = 16$ for 4B and $k = 8$ for 1.7B. The `OLMo-3-7B-Base` SAE uses layer 21, a $32\times$ expansion, and target average sparsity $k = 16$. The Qwen SAEs are trained on FineWeb-Edu hidden activations dumped from the corresponding base models using the EleutherAI `sparsify` library, with Adam and learning rate 10^{-4}.

The SAE parameters remain fixed throughout policy optimization, and no SAE forward pass is required at inference time. We set $\alpha = 2$ and $\tau = 0.05$ for all three backbones and both tasks.

C Evaluation Details

We evaluate accuracy with greedy decoding on six math benchmarks: MATH-500 (Lightman et al., 2023), GSM8K (Cobbe et al., 2021), AIME25 (Zhang & Math-AI, 2025), AMC (Li et al., 2024), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). The number of problems per benchmark is $(500, 1319, 30, 83, 272, 674)$, for a total of 2878 problems. We report the unweighted average across these six benchmarks. AIME25 has only 30 problems, so one problem changes its accuracy by 3.3 percentage points. For math, answers are extracted as the last boxed expression and graded by `math_verify`.

For code generation, we evaluate on LiveCodeBench-v5 (Jain et al., 2025; Luo et al., 2025a) and LeetCode Dataset v0.3.1 (Xia et al., 2025) (228 held-out problems). A problem counts as solved only if all its unit tests pass. The reported code average weights the two benchmarks equally.

Unless otherwise stated, comparisons use endpoint checkpoints under matched training budgets. This avoids selecting each method's best checkpoint independently.

D Interpretability Analysis Details

D.1 Feature-Level Heatmap Construction

Stage 1: Rollout pool. We draw a stratified subsample of 200 problems from the MATH-500 test split. The seven subject labels are Algebra, Geometry, Number Theory, Counting & Probability, Prealgebra, Intermediate Algebra, and Precalculus. Sampling uses `numpy.random.default_rng` with seed 42, with approximately $200/7$ problems per subject. For each question, we generate $G = 8$ rollouts with `Qwen3-4B-Base` at $T = 1.0$, *max_new_tokens* $= 512$, and the math prompt described in Appendix B.2. We grade each rollout with `math_verify` and keep groups with at least two correct and two incorrect rollouts, so every retained rollout has both comparison sets after self-exclusion.

Stage 2: SAE encoding. For every retained rollout, we capture each generated token's layer-27 hidden state and encode it with the frozen Qwen3-4B SAE. We store the (`top_idx`, `top_acts`) pairs, with $k = 16$ active features per token on average and 81,920 latent features in total.

Stage 3: Cross-rollout semantic alignment. For each token, we compute soft alignment against all generated positions in every other rollout of the same question, then average those alignment scores over correct and incorrect rollouts separately. The resulting raw semantic score $S_{qg,t}$ follows Eqn. (10)–Eqn. (12). The token's own rollout is excluded, and rollouts need not align by generation position.

Stage 4: Feature-level aggregation. For each subject and SAE feature, we average $S_{qg,t}$ over tokens where that feature is active. We rank features by this mean score and display the five highest and five lowest for each subject in the figure. Each row lists the four token strings with highest mean

Table 3: Ablation study of SAE hook layer selection for `Qwen3-1.7B-Base` with GRPO + SAEPO. We report accuracy (%) on six math benchmarks and their unweighted average. Boldface marks the best value in each column.

Layer	MATH-500	GSM8K	AIME25	AMC	Minerva	Olympiad	Avg.
Layer 3	**66.0**	74.5	**3.3**	25.3	28.6	23.1	36.80
Layer 9	64.4	75.6	**3.3**	24.1	29.6	**23.7**	36.78
Layer 15	64.8	70.2	**3.3**	25.3	**29.8**	22.1	35.92
Layer 21	65.6	**76.3**	**3.3**	**34.9**	**29.8**	22.1	**38.68**
Layer 27	64.4	75.9	0.0	33.7	27.6	22.4	37.33

Table 4: Ablation of SAE sparsity k for `Qwen3-1.7B-Base` with GRPO + SAEPO. We report accuracy (%) on six math benchmarks and their unweighted average.

Method	MATH-500	GSM8K	AIME25	AMC	Minerva	Olympiad	Avg.
Vanilla GRPO	63.2	74.9	**3.3**	26.5	23.9	21.5	35.56
SAEPO (k=4)	65.6	74.7	**3.3**	27.7	26.1	21.8	36.54
SAEPO (k=8)	65.6	76.3	**3.3**	34.9	29.8	22.1	**38.68**
SAEPO (k=16)	64.6	**79.0**	0.0	30.1	30.1	21.8	37.61
SAEPO (k=32)	63.6	72.3	**3.3**	**37.3**	32.4	22.1	38.50
SAEPO (k=64)	**66.2**	72.0	0.0	28.9	**33.5**	**23.4**	37.34

activation for that feature; token strings must occur at least twice, and displayed features must fire at least 30 times. Cell color is normalized within each feature row, showing relative activation among its tokens.

D.2 Token-Level Case Study

The case study displays token-level semantic scores along excerpts of individual reasoning traces. Green indicates positive scores, red negative scores, and gray near-zero scores. The examples show that a correct Prealgebra trace can contain negatively scored words, while an incorrect Geometry trace can contain positively scored geometry terms.

E Ablation Details

The component ablation appears in Table 2. Here we give its scoring details and report the SAE sparsity and hook-layer ablations on `Qwen3-1.7B-Base` in Tables 3 and 4.

Raw-Embedding Contrast. This ablation replaces the SAE code $\phi_{qg,t}$ with the raw target-layer hidden state $H_{qg,t}$, while keeping the cross-rollout soft-alignment and correct-minus-incorrect scoring rules. Its lower average in Table 2 suggests that the sparse feature representation matters for this credit signal.

Reverse SAEPO Signal. This ablation reverses the raw semantic score, $S^{\text{rev}}_{qg,t} = -S_{qg,t}$, before batch standardization and advantage shaping. Its lower accuracy indicates that the correct-minus-incorrect direction matters.

Table 4 varies target average sparsity k, with $k = 8$ giving the highest average among the tested values. Table 3 varies the SAE hook layer, with layer 21 giving the highest reported average.

F SAE Quality

Table 5 reports reconstruction quality on FineWeb-Edu sample-10BT. We use all tokens from its first 400 documents, truncating each to 1024 tokens and retaining documents with at least 64 tokens. Activations are collected in bf16 from the residual stream after decoder blocks 21, 27, and 21 for the 1.7B, 4B, and OLMo 7B backbones, respectively; metrics are computed in float32. Evaluation

batches match the SAEs' training sizes of 4096, 2048, and 1024 tokens, since BatchTopK allocates activations across a batch.

MSE is the mean squared reconstruction error, while FVU normalizes this error by activation variance; lower values indicate better reconstruction. A latent is counted as dead if it is never selected with positive activation among the evaluated tokens. We also evaluate each model's RL rollouts using completion tokens only, removing the prompt according to `prompt_len` and truncating at `Problem:`; the table shows the FineWeb-Edu measurements.

Table 5: SAE reconstruction quality on FineWeb-Edu sample-10BT.

Backbone	FVU	MSE	dead (%)
`Qwen3-1.7B-Base`	0.012	0.031	4%
`Qwen3-4B-Base`	0.014	0.052	5%
`OLMo-3-7B`	0.056	0.106	17%

G CASE STUDY EXAMPLES

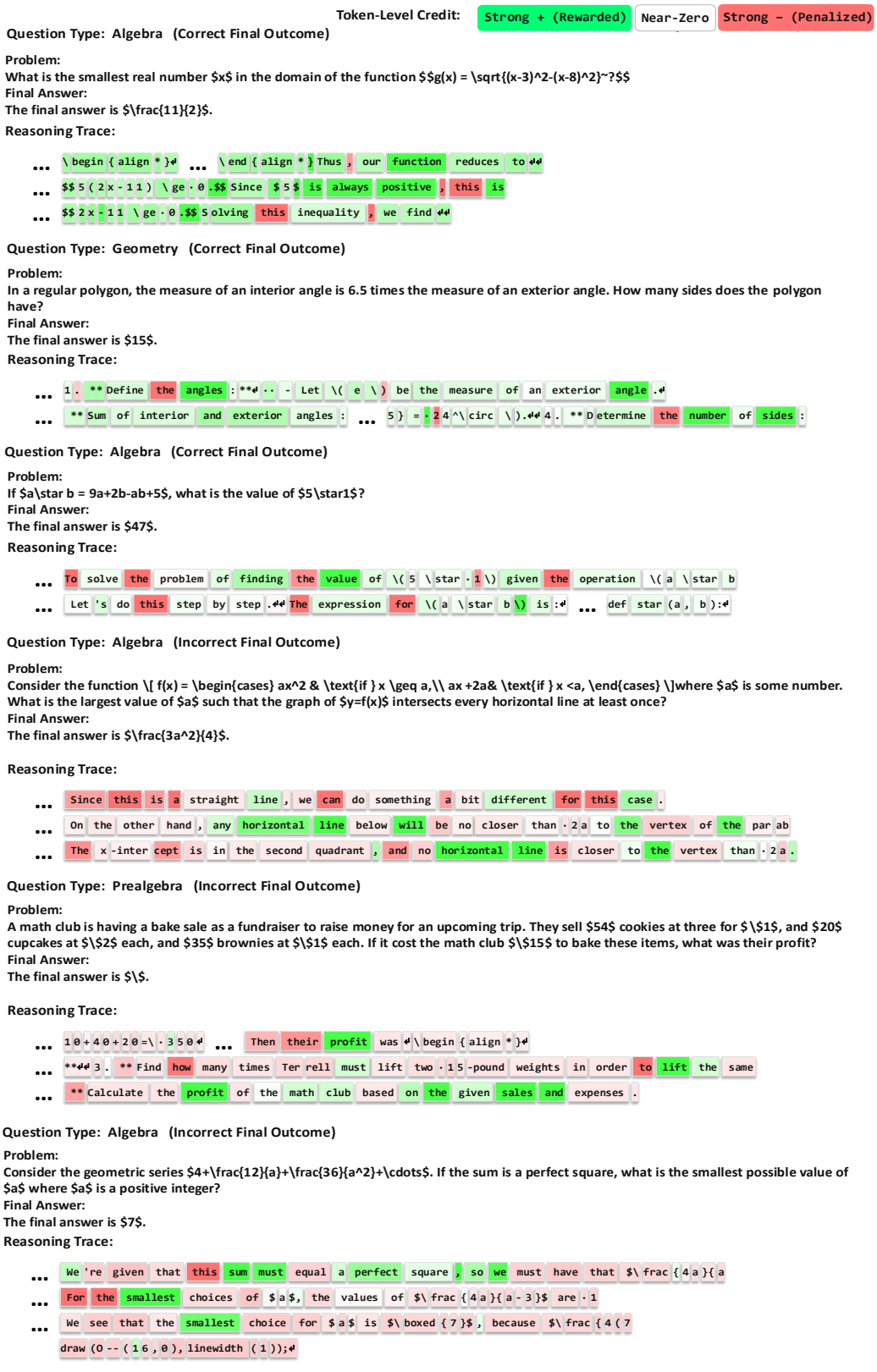

Figure 6: More Case Study Examples.

H More Top Feature Analysis

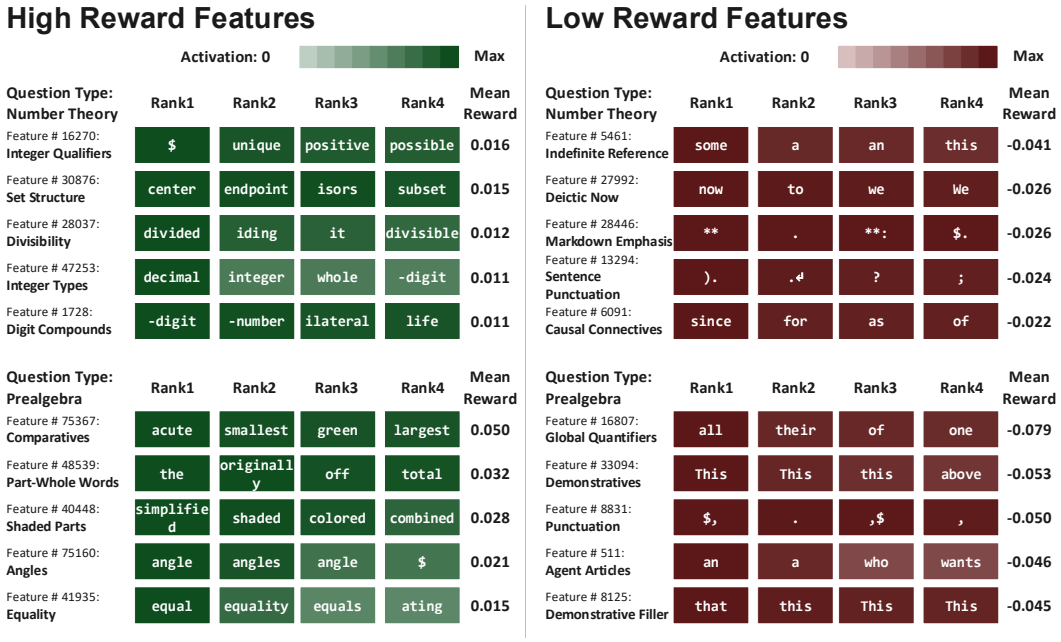

Figure 7: More feature analysis results on "Number Theory" and "Prealgebra" subjects in MATH-500.

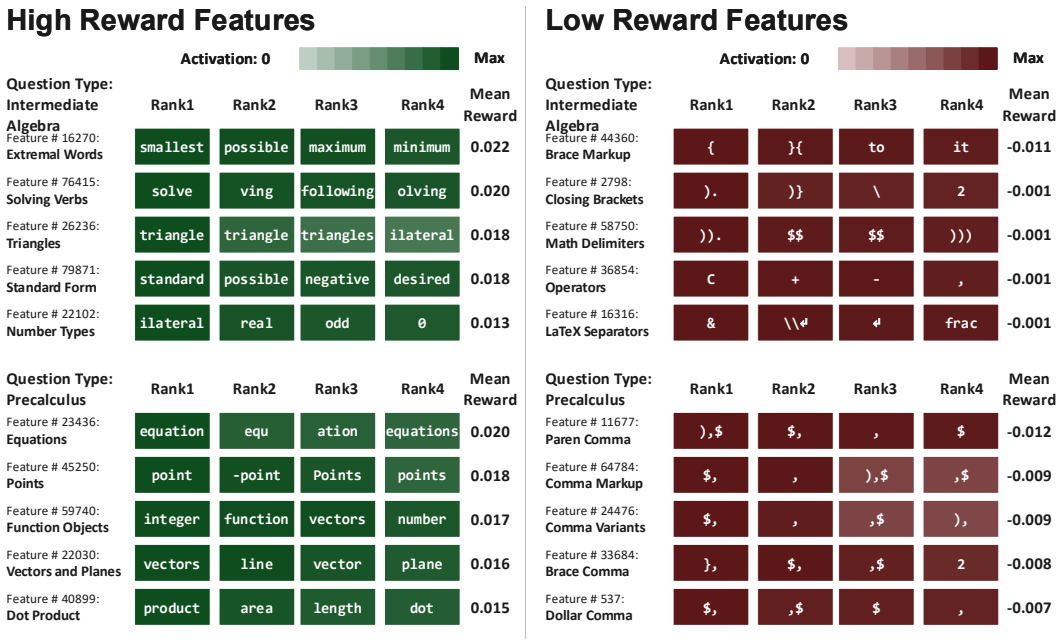

Figure 8: More feature analysis results on "Intermediate Algebra" and "Precalculus" subjects in MATH-500.