TEACH THE WAY IT THINKS: GUIDING LLM POST-TRAINING DATA ENGINEERING WITH SPARSE AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Data engineering for large language model (LLM) reinforcement learning (RL) decides which samples to train on, when to present them, and which to batch together. Existing methods mostly rely on external feedback, such as pass rates or difficulty labels, which observes only the outcome on each sample. We argue that model internals offer richer and more model-intrinsic feedback that can guide all three decisions. We propose SAERL, which organizes RL training data in the feature space of Sparse Autoencoders (SAEs), a mechanistic interpretability tool that decomposes model internals into sparse, interpretable features. SAE features enable viewing the model as a set of functional regions, each formed by features that related samples activate together. For each sample, its SAE activations indicate which regions it engages, and its gradients projected onto SAE features indicate how training on it would change these regions. Guided by this view, SAERL keeps samples with strong and non-redundant gradient signals, trains each functional region from easy to hard, and trains one functional region at a time, drawing each batch from a single activation cluster with moderate cross-cluster mixing. The resulting curriculum is constructed once with small models and transfers to larger ones. Experiments on mathematical reasoning show consistent gains across model scales and RL algorithms. For instance, on `Qwen2.5-Math-1.5B`, SAERL improves the average accuracy of GRPO by 3.1% and reaches the target accuracy with 42% fewer training steps. Further analyses reveal that ordering and batching are complementary. These results demonstrate that model internals are a powerful and practical foundation for post-training data engineering. The source code will be publicly released to facilitate future research.

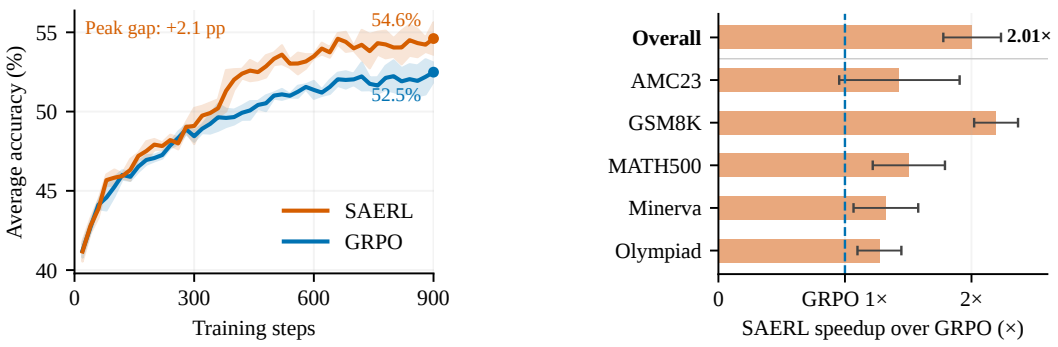
1 INTRODUCTION

Post-training, especially reinforcement learning, has become central to advancing the capabilities of large language models (OpenAI, 2026; Anthropic, 2026; Zeng et al., 2026; DeepSeek-AI, 2026; Kimi Team, 2026). Its effectiveness depends heavily on data engineering, which answers three questions: which samples to train on, when to present them, and which samples to present together in a batch. These choices shape the training signal at every step, making data engineering an important factor for improving both training efficiency and final performance.

Existing post-training data engineering pipelines typically rely on external feedback signals, including human preferences (Ouyang et al., 2022; Lambert et al., 2024), verifier outcomes (DeepSeek-AI et al., 2025; Shao et al., 2024; Yu et al., 2025), rollout pass rates (Sun et al., 2025; Xu et al., 2025; Zheng et al., 2025), and difficulty signals (Narvekar et al., 2020; Shi et al., 2025; Gao et al., 2025; Zhao et al., 2025). These signals have proven useful for data selection and curriculum learning.

However, external feedback is costly to obtain during training (Casper et al., 2023) and observes only the outcome on each sample. A model-intrinsic perspective could provide richer feedback by characterizing which components of the model each sample engages and how training on the sample updates these components. Prior work approximates such a perspective with raw activations and gradients, which have been shown to guide data selection in pre-training (Sam et al., 2025; Rathi & Radford, 2026) and supervised fine-tuning (Ivison et al., 2025; Ma et al., 2025; Chen et al., 2026).

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.



(a) Training performance **(b) Training efficiency**

Figure 1: On `Qwen2.5-Math-1.5B`, SAERL achieves higher average accuracy and reaches GRPO's peak in fewer steps across five math benchmarks: AMC23, GSM8K, MATH500, MinervaMath, and OlympiadBench. Each method is trained with three different seeds; shading and error bars show standard deviations over three runs.

Yet raw activations and gradients superimpose many concepts along the same dimensions (Elhage et al., 2022), offering only an indirect view of which components a sample engages.

Mechanistic interpretability research (Meng et al., 2022; Wang et al., 2022; Somvanshi et al., 2026) continuously explores how to disentangle and understand model internals. As a recent advance, Sparse Autoencoders (SAEs) decompose LLM hidden representations into sparse, fine-grained feature activations (Bricken et al., 2023; Gao et al., 2024; Templeton et al., 2024), providing a disentangled view of model internals: individual features respond to specific concepts, and related inputs activate overlapping groups of features, much as related tasks engage the same functional region of the brain. This makes SAE features a natural representation space for data engineering: sample activations can be encoded as sparse feature vectors, and policy gradients can be projected onto SAE feature directions, placing both internal signals on a shared set of fine-grained features.

We therefore propose SAERL, a data engineering framework that organizes RL post-training data in the SAE feature space. Its design follows three principles, each answering one of the three questions posed earlier: (1) *Train where the model can still improve*: SAERL keeps samples with strong gradient signals on SAE features, drops those whose gradient patterns repeat stronger ones, and retains samples that the small model fails to solve. (2) *Train each functional region from easy to hard*: SAERL clusters samples by their SAE activations, sorts each cluster by a difficulty score predicted from SAE features, and interleaves clusters over training. (3) *Train one functional region at a time*: SAERL draws each batch from a single activation cluster, so that its update concentrates on a shared group of SAE features, and exchanges a few samples across clusters to keep other regions covered. All steps are performed offline with small models and their SAEs, so the resulting curriculum is computed once and reused for training larger models.

Experiments on mathematical reasoning show that SAERL consistently improves both performance and training efficiency across model scales and RL algorithms. On `Qwen2.5-Math-1.5B`, it improves the average accuracy of GRPO by 3.1% and reaches the target accuracy with 42% fewer training steps. Guided by SAEs from small models, SAERL also improves the average accuracy on `Qwen2.5-Math-7B` by 1.9% while achieving the fastest convergence among all methods, showing that the curriculum constructed with small models transfers to larger ones.

Our contributions are as follows. (1) We show that model internals can serve as rich, model-intrinsic feedback for post-training data engineering, complementary to external feedback. (2) We propose SAERL, which selects, orders, and batches RL training data with SAE features, using a curriculum built once on small models and transferable to larger ones. (3) Experiments across model scales and RL algorithms show that SAERL consistently improves both performance and training efficiency, with analyses of how each operation contributes. We hope that this work can facilitate future research on intrinsic data engineering and actionable mechanistic interpretability (Orgad et al., 2026).

2 METHODOLOGY

SAERL reads model-intrinsic feedback for each sample through SAE features: its activations indicate which functional regions of the model it engages, and its gradients projected onto SAE features indicate how training on it would update these regions. We first formalize the problem and outline the pipeline (Section 2.1), and then detail feature extraction, data selection, and curriculum construction (Sections 2.2 to 2.4).

2.1 OVERVIEW

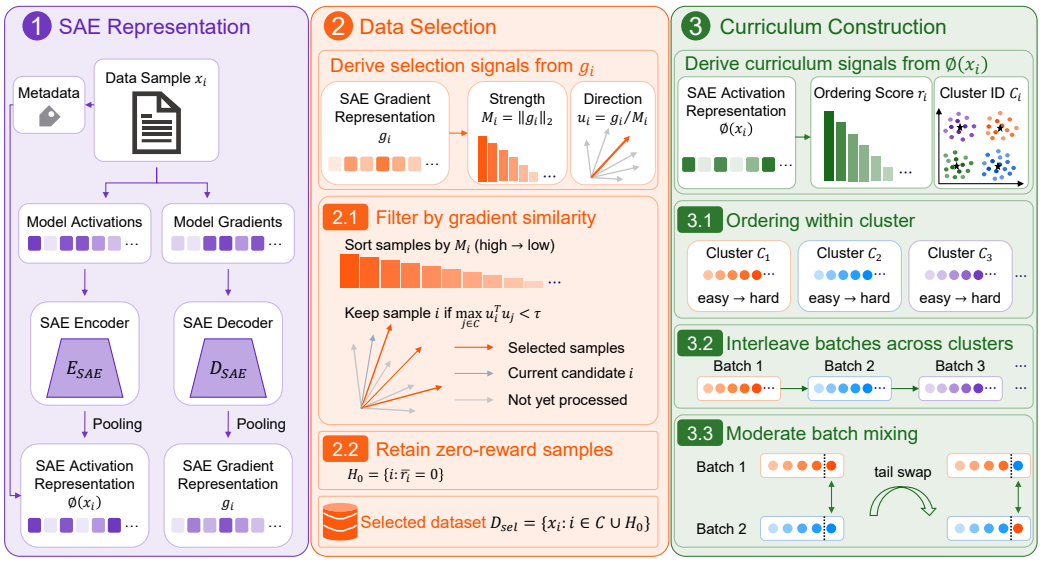


Figure 2: Overview of SAERL. SAE representations of each sample drive data selection by gradient strength and similarity, and curriculum construction by clustering, easy-to-hard ordering, and moderate batch mixing.

Problem setup. Let $\mathcal{D} = \{x_i\}_{i=1}^N$ be the training pool for RL post-training. Given a batch size b, data engineering produces an ordered sequence of batches $\mathcal{S} = (B_1, \ldots, B_T)$ with $B_t \subseteq \mathcal{D}_{\text{sel}}$ and $|B_t| = b$, which the RL trainer consumes in order. Producing $\mathcal{S}$ involves two operations. *Data selection* chooses a subset $\mathcal{D}_{\text{sel}} \subseteq \mathcal{D}$. *Curriculum construction* partitions $\mathcal{D}_{\text{sel}}$ into batches and orders them, determining when each sample is seen and which samples are seen together.

Framework. As shown in Figure 2, SAERL proceeds in three offline steps. (1) *Feature extraction* (Section 2.2): an SAE trained on a small model encodes each sample into an activation representation, and the small model's rollout gradients are projected onto SAE decoder directions to form a gradient representation. (2) *Data selection* (Section 2.3): samples with strong and non-redundant gradient signals are kept, together with samples that receive zero reward under the small model, yielding $\mathcal{D}_{\text{sel}}$. (3) *Curriculum construction* (Section 2.4): $\mathcal{D}_{\text{sel}}$ is clustered into functional regions in the activation space and sorted from easy to hard within each region; batches are then interleaved across clusters and moderately mixed, yielding $\mathcal{S}$. Because all steps rely only on small models and their SAEs, $\mathcal{S}$ is computed once and reused across training runs and model scales.

2.2 FEATURE EXTRACTION

Both operations in SAERL act on sample representations built from SAE features. An SAE decomposes a model's dense hidden activations into sparse feature activations (Bricken et al., 2023; Gao et al., 2024), providing coordinates in which samples can be compared, grouped, and ordered. We construct two representations, each computed with a frozen SAE trained on a small language model.

Activation representation. Given a sample x_i, we extract token-level SAE activations separately from its prompt and solution spans, and aggregate each span with mean and max pooling to capture both sustained and localized activation patterns. Concatenating the pooled activations over both spans gives z_i (960 dimensions), and appending shallow metadata features m_i such as length statistics, TeX ratio, and digit ratio (26 dimensions) yields the activation representation $\phi(x_i) = [z_i, m_i]$, which is used for curriculum construction.

Gradient representation. To characterize how each sample would update the model, we generate rollouts for x_i with the small model, project token-level response gradients onto SAE decoder directions, and aggregate the projections over tokens and rollouts (Shu et al., 2025). The resulting gradient representation g_i describes how the sample's gradient is distributed across SAE features and is used for data selection. Details of both representations are provided in Appendix A.1.

2.3 Data Selection

Data selection aims to retain samples that provide strong and non-redundant training signals. Since this concerns how each sample would update the model, we perform selection on the gradient representations g_i.

Filtering by gradient similarity. We use the norm of g_i to measure each sample's signal strength, and cosine similarity to determine whether two samples induce similar gradient patterns across SAE features. For samples with nonzero representations, define

$$M_i = \|g_i\|_2, \qquad u_i = g_i/M_i. \tag{1}$$

We process samples in descending order of M_i so that stronger signals are considered first. Starting from an empty core set $\mathcal{C}$, a sample is retained only when its similarity to every selected sample is below a threshold τ:

$$\mathcal{C} \leftarrow \mathcal{C} \cup \{i\} \quad \text{if} \quad \mathcal{C} = \emptyset \quad \text{or} \quad \max_{j \in \mathcal{C}} u_i^\top u_j < \tau. \tag{2}$$

A sample is therefore omitted when a selected sample already provides a similar gradient pattern with an equal or stronger signal; τ controls how similar two samples must be for one to be omitted.

Retaining zero-reward samples. Under GRPO, a sample whose rollouts all receive the same reward has zero group-normalized advantages and hence a zero gradient representation, so it never enters the core set. This includes the hard samples that the small model always fails, which the policy should eventually learn. We therefore add all samples with zero average reward $\bar{r}_i$, computed from the same rollouts used for feature extraction:

$$\mathcal{H}_0 = \{i : \bar{r}_i = 0\}, \qquad \mathcal{D}_{\text{sel}} = \{x_i : i \in \mathcal{C} \cup \mathcal{H}_0\}. \tag{3}$$

The resulting $\mathcal{D}_{\text{sel}}$ is passed to curriculum construction.

2.4 Curriculum Construction

Curriculum construction arranges $\mathcal{D}_{\text{sel}}$ into the batch sequence $\mathcal{S}$ and involves two decisions: *ordering*, which determines when each sample appears, and *batching*, which determines which samples share a batch. Both decisions are made on the activation representation $\phi(x_i)$ and share a common clustering. We cluster samples by $\phi(x_i)$ using MiniBatchKMeans (Sculley, 2010) with K=10 and denote the cluster of x_i by c_i. We treat each cluster as a functional region of the model: its samples activate overlapping groups of SAE features and thus engage similar components of the model. The resulting clusters do not simply reproduce the dataset-provided topic labels; they reflect combinations of topic, problem format, and solution profile (Appendix E).

Ordering. We train a lightweight ElasticNet regressor (Zou & Hastie, 2005) on a small difficulty-labeled subset ($|\mathcal{L}|$=3k), producing a raw score $\hat{d}_i = f_D(\phi(x_i))$ for every sample. Because raw scores may vary in scale across clusters, we calibrate them with a global mapping and shrinkage-based cluster corrections:

$$s_i = \text{Calibrate}\big(\hat{d}_i,\, c_i\big), \tag{4}$$

where s_i is the calibrated difficulty score (Appendix A.2). Within each cluster, samples are sorted by s_i in ascending order, forming an easy-to-hard trajectory for each functional region. These per-cluster sequences are then interleaved into a single global order, so that overall difficulty rises over training while every stage of training draws on multiple clusters.

Batching. Each cluster's sorted sequence is partitioned into batches of size b, so each batch initially focuses on a single functional region with samples of similar difficulty, concentrating its update on a shared group of SAE features. To keep other regions covered, we apply moderate batch mixing: each batch is paired with a partner from a nearby stage that has similar average difficulty score and sequence length but a different dominant cluster, and the last h samples of the two batches are exchanged ($h \ll b$). The mixing strength h trades off focus on a single region against coverage of other regions; we find that moderate mixing performs best (Section 4.3), consistent with a bias–variance view (Appendix B). Further details are provided in Appendix A.3.

3 Main Experiment

We evaluate SAERL in the mathematical reasoning domain from two perspectives: downstream performance, compared with data selection and curriculum learning baselines, and training efficiency, measured by the number of training steps needed to reach a target accuracy. We also examine whether the curriculum constructed with small models transfers to a larger model.

3.1 Experiment Setup

Models and training. We train two model scales, `Qwen2.5-Math-1.5B` and `Qwen2.5-Math-7B` (Yang et al., 2024), on `DeepMath-103K` (He et al., 2025) with a batch size of 128 to test the generality of SAERL. We use GRPO (Shao et al., 2024) as the default RL algorithm. We denote the selection-only variant as SAERL$_{\text{Sel}}$, which trains on the selected subset in random order; the curriculum-only variant as SAERL$_{\text{Curr}}$, which applies both ordering and batching to the full dataset; SAERL$_{\text{Ord}}$ and SAERL$_{\text{Batch}}$, which apply only difficulty-based ordering or only SAE-space batching to the full dataset, respectively; and the complete framework as SAERL, which applies the full curriculum to the selected subset. We train SAEs on layer-27 activations of `Qwen2.5-Math-1.5B` and `Qwen3-1.7B` (Yang et al., 2024; 2025a) for data selection and curriculum construction, respectively. This setup tests whether SAE representations from small models can organize training data for other model families and larger scales. Additional details are provided in Appendix C.2.

Evaluation. We evaluate on six benchmarks spanning a wide difficulty range: `GSM8K` (Cobbe et al., 2021) and `AMC23` (lower), `MATH500` (Lightman et al., 2024) and `MinervaMath` (Lewkowycz et al., 2022) (mid), and `OlympiadBench` (He et al., 2024) and `AIME24` (competition-level), which are referred to as `GSM8K`, `AMC`, `MATH`, `MNV`, `OLPD`, and `AIME`, respectively. We report `Pass@8` for `AIME24` and `Avg@8` for the remaining five benchmarks.

Baselines. We compare SAERL against seven baselines. Vanilla **GRPO** (Shao et al., 2024) serves as an RL algorithm baseline without selection or curriculum. For data selection, **LearnAlign** (Li et al., 2025a) combines rollout success rates with gradient alignment to select learnable and representative samples, while **CROPI** (Zhu et al., 2025) uses compressed off-policy gradients to estimate each sample's contribution to the current policy update. Both provide gradient-based comparisons. **IRDS** (Li et al., 2026a) selects samples through verifier-coupled coverage of SAE feature clusters, providing an SAE-based selection comparison. For curriculum construction, **Difficulty Curriculum Learning** (Narvekar et al., 2020) orders samples by the difficulty labels provided with the dataset, representing curricula built on external annotations. **ADARFT** (Shi et al., 2025) estimates difficulty from rollout accuracy, representing rollout-based curriculum methods. **GAINRL** (Wang et al., 2025a) schedules training samples by the angle concentration of their hidden-state representations, representing curricula built on model internals.

Model	Type	Method	Accuracy (%) ↑							Steps to Target ↓						
			AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**
Qwen2.5 Math-1.5B	-	GRPO	30.0	53.4	81.1	71.4	26.1	34.8	49.5	40	340	560	700	780	800	537
	Selection	LearnAlign	**36.7**	52.2	80.5	70.7	25.0	34.6	49.9	**20**	420	520	640	720	780	517
		CROPI	33.3	**55.3**	80.4	70.5	25.3	34.9	50.0	40	520	660	720	800	800	590
		IRDS	33.3	53.8	81.6	71.3	25.6	35.2	50.1	**20**	620	460	560	640	700	500
	Curriculum	DIFF	33.3	55.0	81.7	72.2	26.5	34.7	50.6	60	340	460	600	700	620	463
		ADARFT	33.3	52.8	81.7	72.3	25.8	34.8	50.1	40	620	460	540	680	800	523
		GAINRL	33.3	53.1	79.2	70.9	25.6	35.0	49.5	40	600	780	800	780	840	640
	Joint	**SAERL**	**36.7**	**55.3**	**85.4**	**72.7**	**29.4**	**36.0**	**52.6**	**20**	**200**	**320**	**380**	**380**	**560**	**310**
Qwen2.5 Math-7B	-	GRPO	46.7	68.1	90.4	79.2	33.7	42.1	60.0	40	480	300	280	500	280	313
	Selection	LearnAlign	50.0	66.6	**91.4**	79.4	**36.1**	42.3	61.0	120	880	320	300	260	300	363
		CROPI	**56.7**	67.5	88.8	77.5	33.0	39.4	60.5	40	380	700	480	780	680	510
	Curriculum	DIFF	50.0	68.8	90.9	79.1	33.0	**42.7**	60.7	**20**	760	340	300	900	260	430
		ADARFT	53.3	66.9	88.9	79.0	32.8	40.9	60.3	**20**	**320**	640	300	420	240	323
		GAINRL	53.3	68.4	90.1	**79.8**	34.7	41.8	61.4	80	440	260	**200**	**240**	280	250
	Joint	**SAERL**	53.3	**70.6**	90.3	**79.8**	34.8	42.3	**61.9**	**20**	400	**240**	**200**	260	**220**	**223**

Table 1: Training performance and efficiency on six mathematical reasoning benchmarks. Accuracy (%) is reported at training step 900. For each model–benchmark pair, the target accuracy is defined as the minimum final accuracy across all compared methods, ensuring that every method reaches the target by step 900. Methods are categorized into selection, curriculum, and joint data-engineering approaches. DIFF denotes Difficulty Curriculum Learning. Higher accuracy and fewer training steps are better. **Bold** and underline denote the best and second best results, respectively.

3.2 TRAINING PERFORMANCE

Table 1 shows that SAERL consistently improves average accuracy across baseline families and model scales. At the 1.5B scale, SAERL outperforms both gradient-based selectors and the SAE-based IRDS baseline, as well as curricula based on external difficulty labels, rollout accuracy, or hidden-state geometry. The ablation in Table 2 further shows that each operation alone already surpasses the corresponding baselines, indicating that organizing data in the SAE feature space benefits both selection and curriculum construction. At the 7B scale, SAERL achieves the highest average accuracy, suggesting that data organized with SAEs from smaller models remains effective for training larger models. Notably, SAERL uses only about 43% of the full dataset yet surpasses full-data GRPO in average accuracy at both model scales, suggesting that SAE-gradient selection retains a compact subset with more useful training signals than the full unfiltered dataset.

3.3 TRAINING EFFICIENCY

SAERL improves training efficiency by reducing both training steps and preparation cost.

Table 1 demonstrates that SAERL is the most efficient method at both model scales. At the 1.5B scale, SAERL reaches the target accuracy in the fewest training steps on every benchmark, suggesting that SAE-guided data engineering yields consistent convergence gains. At the 7B scale, it achieves the fastest average convergence, indicating that the curriculum constructed with smaller models also accelerates training on a larger model.

SAERL also reduces preparation cost. SAERL performs selection once offline and reuses the selected subset throughout RL training, avoiding the repeated gradient computation and data reranking required by CROPI at every curriculum round. Moreover, it constructs the subset using SAE gradients from the 1.5B model and transfers it directly to 7B training, avoiding selection-time rollouts and gradient computation on the larger model. Regarding curriculum construction, the Difficulty baseline and ADARFT achieve comparable convergence speed but require LLM-generated labels or multiple rollouts per problem at substantial cost. In contrast, SAERL trains the difficulty proxy from a small labeled subset of 3,000 samples, and SAE encoding for the full dataset of 103,022 samples takes about 1 A800 GPU hour. Thus, SAERL obtains its convergence gains with substantially lower preprocessing overhead.

To test robustness beyond the default setup, we vary the RL algorithm and batch size for SAERL and the SAE input span, source model, and the difficulty supervision source for $\text{SAERL}_{\text{Curr}}$. Across these

settings on `Qwen2.5-Math-1.5B`, both variants improve average accuracy and convergence over their respective baselines, demonstrating that SAE-guided data engineering remains effective across training configurations, SAE representations, and difficulty supervision sources (Appendix D).

4 ANALYSIS

We analyze the sources of SAERL's gains from four aspects: component contributions, representation space, batch composition, and interpretability.

4.1 ABLATION STUDY

SAERL combines data selection with a curriculum consisting of ordering and batching. Data selection retains samples with strong, non-redundant gradient signals, ordering defines the easy-to-hard trajectory, and batching balances within-batch coherence and cross-batch coverage in SAE space.

Method	Accuracy (%) ↑							Steps to Target ↓						
	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**
GRPO	30.0	53.4	81.1	71.4	26.1	34.8	49.5	40	340	840	800	780	800	600
SAERL$_{\text{Sel}}$	36.7	55.3	85.3	**73.4**	27.8	35.8	52.4	**20**	440	460	520	540	580	427
SAERL$_{\text{Curr}}$	**40.0**	**56.3**	83.6	72.0	27.3	35.8	52.5	**20**	220	520	580	660	620	437
SAERL$_{\text{Batch}}$	33.3	52.2	81.7	71.4	25.1	35.1	49.8	60	340	800	780	760	800	590
SAERL$_{\text{Ord}}$	36.7	55.0	82.3	71.3	25.3	34.5	50.9	80	240	840	860	740	900	610
SAERL	36.7	55.3	**85.4**	72.7	**29.4**	**36.0**	**52.6**	**20**	**200**	**360**	**480**	**380**	**560**	**333**

Table 2: Ablation results on `Qwen2.5-Math-1.5B`, reported in accuracy (%) at step 900 and training steps to target. **Bold** and underline denote the best and second best results, respectively.

Table 2 shows that both SAERL$_{\text{Sel}}$ and SAERL$_{\text{Curr}}$ improve accuracy and convergence over vanilla GRPO, confirming that data selection and curriculum construction are each effective on their own. Combining them in SAERL further reduces the average steps to target from 427 and 437 to 333 while slightly improving accuracy, indicating that the two operations are complementary. Within the curriculum, SAERL$_{\text{Ord}}$ and SAERL$_{\text{Batch}}$ yield limited gains alone (50.9 and 49.8), whereas combining them in SAERL$_{\text{Curr}}$ improves both accuracy and convergence. The two play complementary roles: ordering provides an easy-to-hard progression, whereas batching maintains within-batch coherence and cross-batch coverage at each stage.

4.2 REPRESENTATION SPACE ANALYSIS

To test whether SAE decomposition provides more useful data engineering signals than the underlying hidden states, Table 3 compares hidden-state and SAE gradients for selection, and hidden-state and SAE activations for curriculum construction. Both types of signals improve over GRPO, but the SAE variants achieve higher average accuracy in each operation. By decomposing dense hidden states into sparse features, SAEs can mitigate superposition and expose clearer feature-level signals (Elhage et al., 2022). This feature-level structure makes gradient signals more discriminative for selecting valuable samples and activation patterns more informative for curriculum construction, accounting for the gains over dense hidden-state representations.

4.3 BATCH DIVERSITY ANALYSIS

The cluster-first curriculum introduces moderate cross-cluster batch mixing to balance within-batch gradient coherence and cross-cluster coverage. The mixing strength is controlled by the number of tail samples swapped between batches. To examine how mixing strength affects downstream performance, we compare five curriculum variants that differ only in this parameter (`mix0`, `mix4`, `mix8`, `mix16`, and `mix32`), where `mix0` is the cluster-first curriculum with no mixing, and larger indices correspond to stronger cross-cluster mixing. All other components of SAERL are held fixed.

We quantify batch diversity by the mean in-batch k-NN distance (k=5) computed in the two-dimensional SAE projection space, and measure downstream performance by the average `Avg@8`

Operation	Representation	AIME	AMC	GSM8K	MATH	MNV	OLPD	Avg
GRPO	–	30.0	53.4	81.1	71.4	26.1	34.8	49.5
Selection	Hidden-state gradient	36.7	54.4	81.9	71.5	25.6	34.5	50.8
	SAE gradient	36.7	55.3	**85.3**	**73.4**	**27.8**	**35.8**	52.4
Curriculum	Hidden-state activation	33.3	55.0	82.2	71.4	25.5	34.2	50.3
	SAE activation	**40.0**	**56.3**	83.6	72.0	27.3	**35.8**	**52.5**

Table 3: Comparison of hidden-state and SAE representations for data selection and curriculum construction on `Qwen2.5-Math-1.5B` at step 900. Selection compares hidden-state gradients with SAE gradients, and curriculum construction compares hidden-state activations with SAE activations. **Bold** and underline denote the best and second-best results in each benchmark, respectively.

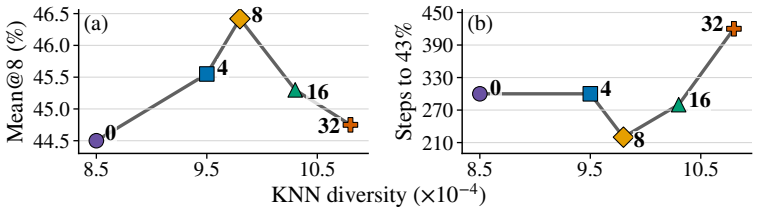


Figure 3: SAE-space batch diversity versus downstream reinforcement learning performance. **(a)** Average `Avg@8` at step 800 as a function of the mean in-batch k-NN distance in SAE space, with $k = 5$. **(b)** Number of training steps required to reach the fixed average `Avg@8` threshold $\alpha = 43.0\%$. Moderate diversity, represented by `mix8`, achieves the best step-800 performance and the fastest threshold crossing.

across the six evaluation benchmarks used in the main experiments. Figure 3 reports both peak performance at step 800 and the number of steps required to reach a fixed threshold $\alpha = 43.0\%$.

The results reveal a clear non-monotonic relationship. Performance improves steadily from `mix0` to `mix8`, and `mix8` reaches α in the fewest training steps. Beyond this point, further increasing the mixing strength to `mix16` and `mix32` degrades both step-800 accuracy and convergence speed—despite `mix32` achieving the highest measured diversity. These results favor moderate mixing among the evaluated configurations and show that increasing the measured diversity alone does not ensure better training outcomes.

A bias–variance analysis (Appendix B) suggests that partial mixing can reduce estimation error, while excessive mixing biases the update away from the reference gradient.

4.4 Interpretability Analysis

Beyond aggregate performance, we examine how SAERL organizes data in the SAE feature space over the course of its curriculum. Figure 4 visualizes three 32-batch windows from the early, middle, and late stages of the curriculum on a shared t-SNE map of SAE representations, with protocol and cluster-audit details in Appendix E.

Shifting cluster composition. Figure 4 highlights representative clusters whose shares shift across stages, computed over the complete windows. C2 (abstract algebra and proof) is most prevalent early (15.5%) and drops to 2.8% in the middle; C9 (integration procedures) and C4 (integral reasoning) peak in the middle (23.6% and 14.4%); and C5 (calculus transforms) rises steadily from 6.9% to 25.6%. No single cluster dominates any stage, consistent with the interleaved design of SAERL: each batch focuses on one functional region, while successive batches rotate across regions.

5 Related Work

We discuss two trends behind our work: post-training data engineering is becoming more adaptive, and model internals are increasingly used to guide training.

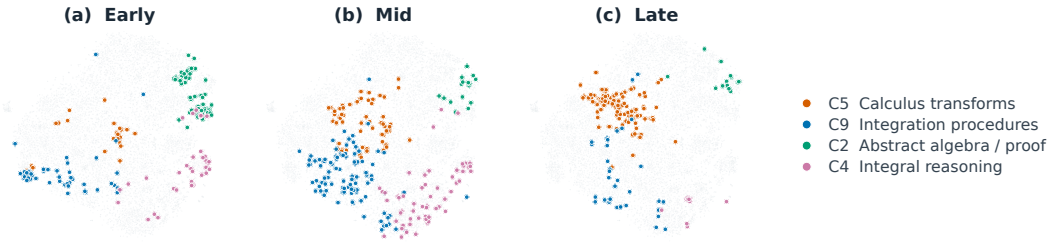


Figure 4: Curriculum order in SAE space. Examples from the **(a)** early (batches 1–32), **(b)** middle (batches 159–190), and **(c)** late (batches 317–348) windows of the curriculum, with four representative clusters highlighted in color; gray points show the fixed background sample. Cluster names are abbreviated from the semantic audit in Table 11 and are conservative summaries rather than one-to-one topic labels.

Post-training data engineering. Post-training data engineering shapes model behavior through two operations, data selection and curriculum construction (Ouyang et al., 2022; Lambert et al., 2024; DeepSeek-AI et al., 2025). For data selection, many methods filter prompts by rollout pass rates or difficulty (Li et al., 2025b; Chen et al., 2025; Sun et al., 2025; Zheng et al., 2025; Zhao et al., 2025), while others score samples by gradient alignment (Li et al., 2025a; Yang et al., 2026; Zhu et al., 2025). For curriculum construction, most methods order samples from easy to hard by difficulty labels, rollout accuracy, or learning progress (Narvekar et al., 2020; Tzannetos et al., 2024; Shi et al., 2025; Gao et al., 2025; Rajaraman et al., 2026), schedule them by hidden-state geometry (Wang et al., 2025a), or reweight data distributions during training (Wang et al., 2025b). Others control batch composition through task ratios (Ramesh et al., 2026), variance-based sampling (Leng et al., 2025), prompt replay (Baroian & Berger, 2026), or explore–exploit allocation (Liang et al., 2026). These methods rely on external feedback or raw model signals, whereas SAERL makes all three decisions with SAE features that indicate which functional regions each sample engages and how training on it would update them.

Model internals for data engineering. Model internals have moved from post-hoc analysis toward guiding training. In pre-training and supervised fine-tuning, losses, gradients, and hidden representations guide data filtering and selection (Li et al., 2024; Lin et al., 2024; Xia et al., 2024; Ivison et al., 2025; Ma et al., 2025; Sam et al., 2025; Rathi & Radford, 2026; Chen et al., 2026). In RL post-training, raw policy gradients and hidden states likewise guide prompt selection and sampling (Li et al., 2025a; Zhu et al., 2025; Wang et al., 2025a; Liang et al., 2026). SAEs decompose hidden representations into sparse features, many of which correspond to human-interpretable concepts (Bricken et al., 2023; Cunningham et al., 2023; Templeton et al., 2024; Gao et al., 2024). More recently, SAE activations have been used to select or synthesize training data (Yang et al., 2025b; Li et al., 2026b;a). SAERL extends them to ordering and batching, combines SAE activations with SAE-projected gradients, and builds the curriculum once with small models for reuse across scales.

6 Conclusion

We present SAERL, which uses model internals, read through SAE features, as feedback for organizing RL post-training data. Treating groups of co-activated SAE features as functional regions, SAERL keeps samples with strong, non-redundant gradient signals, trains each region from easy to hard, and trains one region at a time with moderate mixing. Experiments on mathematical reasoning show consistent gains in accuracy and efficiency across model scales and RL algorithms, with the curriculum built once on small models and transferred to larger ones. These results suggest that model internals offer rich, model-intrinsic feedback for post-training data engineering, complementary to external feedback.

Future work will extend SAERL beyond mathematical reasoning to broader post-training settings, better understand how batch composition across functional regions shapes RL updates, and explore how model internals can further reduce reliance on external supervision.

AI USE STATEMENT

We use AI assistants in three ways: (1) to aid and polish the writing of this paper; (2) to support literature retrieval and discovery, such as finding related work, with all cited papers checked by the authors; and (3) to assist research ideation and execution, such as discussing ideas and developing code. All AI-assisted content is reviewed and edited by the authors, who remain responsible for the final scientific claims, experiments, and writing.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure that our results are reproducible. Section 2 describes each stage of SAERL, and Section A provides the technical details of SAE sample representation, clustering, difficulty calibration, curriculum ordering, and batch mixing. Section 3.1 specifies the models, training data, RL algorithm, benchmarks, and evaluation metrics, all of which are publicly available. Section C reports the SAE training details (Section C.1), all hyperparameters (Section C.2), baseline implementations (Section C.3), and preprocessing cost (Section C.4). We will release the code for SAE feature extraction, data selection, curriculum construction, training, and evaluation upon publication.

REFERENCES

Anthropic. System Card: Claude Opus 5.5. https://www.anthropic.com/claude-opus-5-5, 2026. Published September 22, 2026.

Andrei Baroian and Rutger Berger. Prompt replay: Speeding up GRPO with on-policy reuse of high-signal prompts, 2026. URL https://arxiv.org/abs/2603.21177.

Trenton Bricken et al. Towards monosemanticity: Decomposing language models with dictionary learning. Transformer Circuits Thread, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features/.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback, 2023. URL https://arxiv.org/abs/2307.15217.

Xin Chen, Junchao Wu, Shu Yang, Runzhe Zhan, Zeyu Wu, Min Yang, Shujian Huang, Lidia S. Chao, and Derek F. Wong. Neuron-aware data selection in instruction tuning for large language models, 2026. URL https://arxiv.org/abs/2603.13201.

Xinjie Chen, Minpeng Liao, Guoxin Chen, Chengxi Li, Biao Fu, Kai Fan, and Xinggao Liu. From data-centric to sample-centric: Enhancing LLM reasoning via progressive optimization, 2025. URL https://arxiv.org/abs/2507.06573.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression, 2026. URL https://arxiv.org/abs/2609.19969.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Boyi Deng, Xu Wang, Yaoning Wang, Yu Wan, Yubo Ma, Baosong Yang, Haoran Wei, Jialong Tang, Huan Lin, Ruize Gao, et al. Qwen-scope: Turning sparse features into development tools for large language models. *arXiv preprint arXiv:2605.11887*, 2026.

Nelson Elhage et al. Toy models of superposition. Transformer Circuits Thread, 2022. URL https://transformer-circuits.pub/2022/toy_model/.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, 2024. URL https://arxiv.org/abs/2406.04093.

Zhaolin Gao, Joongwon Kim, Wen Sun, Thorsten Joachims, Sid Wang, Richard Yuanzhe Pang, and Liang Tan. Prompt curriculum learning for efficient LLM post-training, 2025. URL https://arxiv.org/abs/2510.01135.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025. URL https://arxiv.org/abs/2504.11456.

Hamish Ivison, Muru Zhang, Faeze Brahman, Pang Wei Koh, and Pradeep Dasigi. Large-scale data selection for instruction tuning, 2025. URL https://arxiv.org/abs/2503.01807.

Kimi Team. Kimi K3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

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, 2024. URL https://arxiv.org/abs/2411.15124.

Sicong Leng, Jing Wang, Jiaxi Li, Hao Zhang, Zhiqiang Hu, Boqiang Zhang, Yuming Jiang, Hang Zhang, Xin Li, Lidong Bing, Deli Zhao, Wei Lu, Yu Rong, Aixin Sun, and Shijian Lu. MMR1: Enhancing multimodal reasoning with variance-aware sampling and open resources, 2025. URL https://arxiv.org/abs/2509.21268.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14255–14273. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.769. URL https://aclanthology.org/2024.acl-long.769/.

Shipeng Li, Zhiqin Yang, Shikun Li, Xiaobo Xia, Hengyu Liu, Xinghua Zhang, Gaode Chen, Dong Fang, Ying Tai, and Zhe Peng. LearnAlign: Data selection for LLM reinforcement learning with improved gradient alignment, 2025a. URL https://arxiv.org/abs/2506.11480.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. LIMR: Less is more for RL scaling, 2025b. URL https://arxiv.org/abs/2502.11886.

Yuhan Li, Mingxu Zhang, Dazhong Shen, and Ying Sun. IRDS: Interpretable RLVR data selection via verifier-coupled sparse autoencoder coverage, 2026a. URL https://arxiv.org/abs/2605.28247.

Zhongzhi Li, Xuansheng Wu, Yijiang Li, Lijie Hu, and Ninghao Liu. Less is enough: Synthesizing diverse data in feature space of LLMs, 2026b. URL https://arxiv.org/abs/2602.10388.

Shuang Liang, Haoyang Zhou, Yifan Gong, Guowei Wang, and Xiting Wang. LEEPS: Latent-guided explore-exploit prompt sampling for efficient RLVR in large language models, 2026. URL https://arxiv.org/abs/2607.28077.

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. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Rho-1: Not all tokens are what you need, 2024. URL https://arxiv.org/abs/2404.07965.

Da Ma, Gonghu Shang, Zhi Chen, Libo Qin, Yijie Luo, Lei Pan, Shuai Fan, Lu Chen, and Kai Yu. Task-specific data selection for instruction tuning via monosemantic neuronal activations, 2025. URL https://arxiv.org/abs/2503.15573.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT, 2022. URL https://arxiv.org/abs/2202.05262.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020. URL https://www.jmlr.org/papers/v21/20-212.html.

OpenAI. GPT-6 Astra System Card. https://deploymentsafety.openai.com/gpt-6-astra, 2026. Published September 3, 2026.

Hadas Orgad, Fazl Barez, Tal Haklay, Isabelle Lee, Marius Mosbach, Anja Reusch, Naomi Saphra, Byron Wallace, Sarah Wiegreffe, Eric Wong, et al. Interpretability can be actionable. *arXiv preprint arXiv:2605.11161*, 2026.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Nived Rajaraman, Audrey Huang, Miro Dudik, Robert Schapire, Dylan J. Foster, and Akshay Krishnamurthy. Learning to reason with curriculum I: Provable benefits of autocurriculum, 2026. URL https://arxiv.org/abs/2603.18325.

Shyam Sundhar Ramesh, Xiaotong Ji, Matthieu Zimmer, Sangwoong Yoon, Zhiyong Wang, Haitham Bou Ammar, Aurelien Lucchi, and Ilija Bogunovic. Multi-task GRPO: Reliable LLM reasoning across tasks, 2026. URL https://arxiv.org/abs/2602.05547.

Neil Rathi and Alec Radford. Shaping capabilities with token-level data filtering, 2026. URL https://arxiv.org/abs/2601.21571.

Dylan Sam, Ayan Chakrabarti, Afshin Rostamizadeh, Srikumar Ramalingam, Gui Citovsky, and Sanjiv Kumar. Analyzing similarity metrics for data selection for language model pretraining, 2025. URL https://arxiv.org/abs/2502.02494.

D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, pp. 1177–1178. Association for Computing Machinery, 2010. doi: 10.1145/1772690.1772862.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Taiwei Shi, Yiyang Wu, Linxin Song, Tianyi Zhou, and Jieyu Zhao. Efficient reinforcement finetuning via adaptive curriculum learning, 2025. URL https://arxiv.org/abs/2504.05520.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Mengnan Du, and Ninghao Liu. Beyond input activations: Identifying influential latents by gradient sparse autoencoders. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1673–1682, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.87. URL https://aclanthology.org/2025.emnlp-main.87/.

Shriyank Somvanshi, Md Monzurul Islam, Amir Rafe, Anannya Ghosh Tusti, Arka Chakraborty, Anika Baitullah, Tausif Islam Chowdhury, Nawaf Alnawmasi, Anandi Dutta, and Subasish Das. Bridging the black box: a survey on mechanistic interpretability in ai. *ACM Computing Surveys*, 58(8):1–35, 2026.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for LLM reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay, 2025. URL https://arxiv.org/abs/2506.05316.

Adly Templeton et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. Transformer Circuits Thread, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/.

Georgios Tzannetos, Parameswaran Kamalaruban, and Adish Singla. Proximal curriculum with task correlations for deep reinforcement learning, 2024. URL https://arxiv.org/abs/2405.02481.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small, 2022. URL https://arxiv.org/abs/2211.00593.

Qinsi Wang, Jinghan Ke, Hancheng Ye, Yueqian Lin, Yuzhe Fu, Jianyi Zhang, Kurt Keutzer, Chenfeng Xu, and Yiran Chen. Angles don't lie: Unlocking training-efficient RL through the model's own signals, 2025a. URL https://arxiv.org/abs/2506.02281.

Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. DUMP: Automated distribution-level curriculum learning for RL-based LLM post-training, 2025b. URL https://arxiv.org/abs/2504.09710.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54104–54132. PMLR, 2024. URL https://proceedings.mlr.press/v235/xia24c.html.

Yixuan Even Xu, Yash Savani, Fei Fang, and J. Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in LLM reinforcement learning, 2025. URL https://arxiv.org/abs/2504.13818.

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, 2024. URL https://arxiv.org/abs/2409.12122.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Ningyuan Yang, Weihua Du, Weiwei Sun, Sean Welleck, and Yiming Yang. GradAlign: Gradient-aligned data selection for LLM reinforcement learning, 2026. URL https://arxiv.org/abs/2602.21492.

Xianjun Yang, Shaoliang Nie, Lijuan Liu, Suchin Gururangan, Ujjwal Karn, Rui Hou, Madian Khabsa, and Yuning Mao. Diversity-driven data selection for language model tuning through sparse autoencoder, 2025b. URL https://arxiv.org/abs/2502.14050.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan 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, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Yang Zhao, Kai Xiong, Xiao Ding, et al. UFO-RL: Uncertainty-focused optimization for efficient reinforcement learning data selection, 2025. URL https://arxiv.org/abs/2505.12457.

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

Erle Zhu, Dazhi Jiang, Yuan Wang, Xujun Li, Jiale Cheng, Yuxian Gu, Yilin Niu, Aohan Zeng, Jie Tang, Minlie Huang, and Hongning Wang. Data-efficient RLVR via off-policy influence guidance, 2025. URL https://arxiv.org/abs/2510.26491.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. doi: 10.1111/j.1467-9868.2005.00503.x.

A Technical Details of the SAERL Pipeline

This appendix provides additional technical details for the offline data-processing steps used in Section 2.

A.1 SAE Sample Representation

To construct the activation representation, each sample x_i is divided into a prompt span S_i^p and a solution span S_i^s. Let $\mathcal{F}$ denote the retained SAE feature set, and let $a_t \in \mathbb{R}^{|\mathcal{F}|}$ be the SAE activation vector at token t. For each span, we summarize token-level SAE activations using mean pooling and max pooling. The pooled SAE activations z_i are obtained by concatenating the mean-pooled and max-pooled activations from both the prompt and solution spans:

$$z_i = \left[\bar{a}_{S_i^p}; a_{S_i^p}^{\max}; \bar{a}_{S_i^s}; a_{S_i^s}^{\max}\right]. \tag{5}$$

We further append shallow metadata features m_i, including length statistics, TeX ratio, and digit ratio, and use $\phi(x_i) = [z_i; m_i]$ as the activation representation for clustering and difficulty estimation. In our experiments, z_i is 960-dimensional and m_i is 26-dimensional.

To construct the SAE gradient representation, we generate R rollouts for each sample x_i and denote the gradient at response token t of rollout r by $\gamma_{irt} \in \mathbb{R}^H$. We project these gradients onto unit-normalized SAE decoder directions $D \in \mathbb{R}^{F \times H}$, retain the TopK coordinates of each projection by magnitude, while preserving their signs, and average them over response tokens and rollouts:

$$q_i = \frac{1}{R} \sum_{r=1}^{R} \frac{1}{T_{ir}} \sum_{t=1}^{T_{ir}} \mathrm{TopK}(\gamma_{irt} D^\top). \tag{6}$$

Here T_{ir} is the number of response tokens in rollout r. We use $R = 8$ rollouts per sample, and TopK retains the 128 largest-magnitude coordinates of each projection. We retain the 256 decoder features with the highest mean squared value across ℓ_2-normalized q_i, forming a shared feature set $\mathcal{G}$. To compare how gradient magnitude is distributed across these features, we take the element-wise absolute value. The SAE gradient representation used for selection is

$$g_i = |q_i[\mathcal{G}]| \in \mathbb{R}^{256}_{\geq 0}. \tag{7}$$

A.2 Clustering and Difficulty Calibration

We cluster samples in the feature space using MiniBatchKMeans with $K = 10$. Each sample is assigned to the nearest cluster centroid based on its activation representation $\phi(x_i)$.

To estimate sample difficulty, we use a small difficulty-labeled subset $\mathcal{L}$ with $|\mathcal{L}| = 3{,}000$. For each labeled sample, the difficulty label is denoted by $d_i^\star$. We train an ElasticNet difficulty proxy f_D on the labeled subset and use it to produce the raw difficulty prediction $\hat{d}_i = f_D(\phi(x_i))$.

Because only a limited number of difficulty labels are available, the raw proxy score is calibrated using a global calibration map fitted on the small labeled subset, together with a shrinkage-based cluster residual. Let κ denote the global calibration map fitted on $\mathcal{L}$.

For each cluster c, we compute the average residual Δ_c between the labeled difficulty $d_i^\star$ and the globally calibrated prediction $\kappa(\hat{d}_i)$ over labeled samples in that cluster. If a cluster has no labeled samples, we set $\Delta_c = 0$. The residual is then scaled by a shrinkage weight $\lambda_c = n_c/(n_c + \tau_{\mathrm{sh}})$, where n_c is the number of labeled samples in cluster c and $\tau_{\mathrm{sh}} > 0$ controls the strength of shrinkage toward the global calibration.

The final difficulty score is

$$s_i = \kappa(\hat{d}_i) + \lambda_{c_i} \Delta_{c_i}. \tag{8}$$

A larger s_i corresponds to a higher estimated difficulty.

A.3 Curriculum Ordering and Moderate Batch Mixing

Within each cluster, samples are sorted by the calibrated difficulty score s_i in ascending order, resulting in a local easy-to-hard curriculum. The ordered samples in each cluster are then partitioned into fixed-size batches. The global curriculum interleaves these batches across clusters stage by stage, which preserves local easy-to-hard trajectories while maintaining coverage across different clusters.

After the cluster-first curriculum has been constructed, we apply moderate batch mixing. For an ordered batch of size b, we keep the first $b - h$ samples fixed and exchange only the last h tail samples with another batch. The partner batch is selected from a local curriculum window and must satisfy three conditions: it should have similar average calibrated difficulty, similar average sequence length, and a different dominant cluster. The dominant cluster of a batch is defined as the most frequent cluster label among its samples.

Since only the tail block is exchanged, this operation introduces limited cross-cluster mixing while largely preserving the local curriculum structure within each batch. In our experiments, the curriculum uses 16-batch difficulty windows and $h = 8$, corresponding to `mix8` in Section 4.3.

B A BIAS–VARIANCE VIEW OF MODERATE BATCH MIXING

We provide a local bias–variance perspective on the potential benefit of moderate cross-cluster batch mixing. The analysis illustrates when partial mixing can improve gradient estimation under a simplified sampling model.

Setup. Fix the model parameters θ and denote the per-sample training gradient by $\tilde{g}_i = \nabla_\theta \ell(x_i; \theta)$. As a modeling assumption, suppose $\tilde{g}_i = F(\phi(x_i)) + \varepsilon_i$, where $\phi(x_i)$ is the activation representation used for clustering, F is locally Lipschitz, and ε_i captures variation not explained by $\phi(x_i)$. When residual differences are small, proximity in SAE space can therefore correspond to similar gradients, motivating cluster-based batching. Let μ_c and Σ_c be the gradient mean and covariance within cluster c, with finite second moments. We idealize sampling as independent within and across clusters, and let G_t be a fixed reference gradient for the data distribution of interest at the current curriculum stage.

Pure-cluster bias. For a batch of size b drawn entirely from cluster c, the mean squared error of $\hat{G}_c = b^{-1} \sum_{i=1}^{b} \tilde{g}_i$ relative to G_t decomposes as

$$\mathrm{MSE}_0 = \|G_t - \mu_c\|^2 + \frac{1}{b} \operatorname{tr}(\Sigma_c), \tag{9}$$

where the two terms are squared cluster-local bias and estimation variance, respectively. A cluster with small $\operatorname{tr}(\Sigma_c)$ yields a low-variance estimate, but its mean may still deviate from the reference gradient.

Effect of mixing. Replace $h \in \{0, \ldots, b\}$ samples with independent draws from cluster d and write $\rho = h/b$. With these fixed sample counts, the mixed estimator has

$$\mathbb{E}[\hat{G}_\rho] = (1-\rho)\mu_c + \rho\mu_d, \qquad \mathrm{Cov}(\hat{G}_\rho) = \frac{(1-\rho)\Sigma_c + \rho\Sigma_d}{b}. \tag{10}$$

Let $r_c = G_t - \mu_c$ and $v = \mu_d - \mu_c$. Substituting these moments into the bias–variance decomposition gives the MSE change relative to equation 9:

$$\Delta(\rho) = -2\rho\langle r_c, v\rangle + \rho^2\|v\|^2 + \frac{\rho}{b} \operatorname{tr}(\Sigma_d - \Sigma_c), \tag{11}$$

When partial mixing helps. Write $\Delta(\rho) = -A\rho + C\rho^2$, where

$$A = 2\langle r_c, v\rangle - \frac{1}{b} \operatorname{tr}(\Sigma_d - \Sigma_c), \qquad C = \|v\|^2. \tag{12}$$

For $C > 0$, the continuous relaxation is minimized at $\rho^\dagger = \mathrm{clip}(A/(2C), 0, 1)$; feasible mixing counts are obtained by evaluating $\rho = h/b$. An interior minimum exists when $0 < A < 2C$, and mixing improves on the pure-cluster estimate whenever $0 < \rho < A/C$ within the feasible range. If $C = 0$, the change is linear and a boundary point is optimal, with all choices equivalent when $A = 0$.

The alignment term $\langle r_c, v\rangle$ measures whether the partner cluster moves the expected update toward G_t, while the covariance difference accounts for the change in sampling variance. Favorable alignment can make a small amount of mixing beneficial. Beyond the optimal balance, further mixing increases the combined error through added variance or excessive displacement of the mean.

Connection to the curriculum. This local argument motivates introducing cross-cluster information while retaining most of a batch's original composition. The implementation exchanges difficulty-ordered tails between paired batches rather than drawing independent samples, and subsequent updates change both gradients and optimizer state. The analysis therefore provides a heuristic explanation for the empirical benefit of moderate mixing by clarifying the balance between directional bias correction and changes in estimation variance.

C IMPLEMENTATION DETAILS

This appendix provides the training hyperparameters and baseline implementation details used in the experiments.

Hyperparameter	GRPO	DAPO
Framework	`verl`	`verl`
Rollout engine	vLLM	vLLM
Learning rate	10^{-6}	10^{-6}
Train batch size	128	128
Max prompt length	1024	1024
Max response length	3072	3072
Sampling temperature	0.6	0.6
Rollouts per sample	8	8
Candidate generation batch	128	384
PPO clip lower / upper	0.2 / 0.2	0.2 / 0.28
Actor KL coefficient	0.001	0

Table 4: Core training hyperparameters for GRPO and DAPO. DAPO uses dynamic sampling to retain 128 valid prompts from a candidate generation batch of 384.

C.1 Sparse Autoencoder Training Details

We train the SAE using the OpenSAE framework on layer-27 activations of `Qwen3-1.7B`, with an expansion factor of 64 to support fine-grained feature extraction. The training corpus consists of `FineWeb-Edu` (Penedo et al., 2024) and `Wikipedia`, totaling 80 GB. Training takes approximately 29 hours on 4 A100 GPUs, corresponding to 116 A100 GPU hours.

We also train an SAE on layer-27 activations of `Qwen2.5-Math-1.5B`, with an expansion factor of 64, using the same training corpus. Training takes approximately 44 hours on 8 A100 GPUs, corresponding to 352 A100 GPU hours.

C.2 Hyperparameters

We train GRPO on both the `Qwen2.5-Math-1.5B` and `Qwen2.5-Math-7B`, and DAPO only on `Qwen2.5-Math-1.5B`. Table 4 lists the core training hyperparameters.

C.3 Baseline Implementations

For the Difficulty Curriculum Learning baseline (Narvekar et al., 2020), we sort the `DeepMath-103K` dataset (He et al., 2025) by its provided difficulty labels in ascending order and sample progressively throughout training.

To ensure a fair comparison, we train the ADARFT baseline (Shi et al., 2025) using GRPO (Shao et al., 2024). Following the original method, we estimate problem difficulty with `Avg@128` rollout accuracy and use the resulting scores to construct the curriculum.

C.4 Preprocessing Cost

Table 5 compares the preprocessing cost of gradient-based data selection at the 1.5B scale. SAERL extracts and projects SAE gradients once, including rollout generation, which takes 32.6 A800 GPU hours. LearnAlign requires 61.7 GPU hours, including warm-up training before gradient extraction, while CROPI requires 277.4 GPU hours because it recomputes gradients and reranks data at every curriculum round. At the 7B scale, SAERL reuses the subset selected with the 1.5B model and incurs no additional selection cost, whereas LearnAlign and CROPI must repeat rollouts and gradient computation on the larger model, so their costs grow further with model size.

SAE training cost. When this work began, no public SAEs were available for the Qwen family, so we trained both SAEs used by SAERL ourselves (Appendix C.1). Open-source SAEs trained on Qwen models have since been released by Qwen-Scope (Deng et al., 2026), so future work along this line can build directly on public SAEs. We therefore treat SAE training as a one-time cost, exclude it from the comparison above, and report it separately in Appendix C.1.

Method	Selection cost (GPU hours)
LearnAlign	61.7
CROPI	277.4
SAERL	**32.6**

Table 5: Data selection cost on `Qwen2.5-Math-1.5B` in A800 GPU hours.

Metric	B	Method	Accuracy (%) ↑							Steps to Target ↓						
			AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**
Avg @8	128	GRPO	13.8	53.4	81.1	71.4	26.1	34.8	46.8	**520**	680	840	800	780	800	737
		SAERL	**15.4**	**55.3**	**85.4**	**72.7**	**29.4**	**36.0**	**49.0**	660	**480**	**360**	**500**	**520**	**560**	**513**
	512	GRPO	12.5	51.9	80.4	70.0	24.5	33.9	45.5	120	**80**	290	220	260	260	205
		SAERL	**14.6**	**54.7**	**82.6**	**71.1**	**26.7**	**34.8**	**47.4**	**10**	**80**	**180**	**130**	**200**	**230**	**138**
Pass @8	128	GRPO	30.0	**85.0**	94.2	86.2	43.4	52.7	65.3	40	340	600	360	**260**	840	407
		SAERL	**36.7**	**85.0**	**95.1**	**87.2**	**44.1**	**54.8**	**67.1**	**20**	**200**	**320**	**320**	400	**320**	**263**
	512	GRPO	**36.7**	**85.0**	93.4	84.6	40.4	52.5	65.4	70	**210**	170	100	**50**	120	120
		SAERL	**36.7**	**85.0**	**94.2**	**86.0**	**42.3**	**53.2**	**66.2**	**30**	280	**100**	**70**	70	**110**	**110**

Table 6: Accuracy and training efficiency across batch sizes and evaluation metrics. Results are evaluated on `Qwen2.5-Math-1.5B` at step 900 for $B = 128$ and step 300 for $B = 512$, where B denotes the training batch size. **Bold** indicates the best result for each batch size and metric.

Method	Accuracy (%) ↑							Steps to Target ↓						
	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**
DAPO	40.0	**55.6**	82.0	70.8	26.7	34.5	51.6	**40**	580	580	580	540	**580**	483
SAERL$^{\text{DAPO}}$	**43.3**	54.1	**84.6**	**72.2**	**28.5**	**35.3**	**53.0**	100	**560**	**360**	**500**	**400**	**580**	**417**

Table 7: Training performance with DAPO on `Qwen2.5-Math-1.5B`. SAERL$^{\text{DAPO}}$ denotes the complete SAERL data engineering pipeline trained with DAPO. **Bold** marks the best result.

D Additional Robustness Experiments

We assess robustness to training batch size, RL algorithm, SAE input, SAE source, and the source of difficulty supervision. Unless noted otherwise, accuracy is measured at step 900 using Pass@8 for AIME and Avg@8 for the other benchmarks. Steps to target are computed against the lowest final accuracy within each benchmark comparison.

D.1 Training Batch Size

Table 6 shows that SAERL improves average accuracy and convergence over GRPO at both batch sizes under both evaluation metrics, indicating that its gains are not tied to a particular batch size or metric.

D.2 DAPO Training

As shown in Table 7, applying SAERL to DAPO improves average accuracy and reduces mean steps to target, demonstrating that the gains extend beyond GRPO.

D.3 Prompt-Only SAE Representation

Table 8 shows that constructing the curriculum from SAE features extracted only from the prompt span improves both average accuracy and convergence over GRPO. Incorporating features from the response span further improves final accuracy, indicating that response information adds useful structure for curriculum construction.

Method	Accuracy (%) ↑							Steps to Target ↓						
	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**
GRPO	30.0	53.4	81.1	71.4	26.1	34.8	49.5	40	680	840	800	780	800	657
SAERL$_{\text{Curr}}$														
+ Prompt&Response	**40.0**	**56.3**	**83.6**	**72.0**	**27.3**	**35.8**	**52.5**	**20**	**580**	520	**540**	840	**760**	543
+ Prompt	36.7	53.4	83.1	71.0	25.6	35.4	50.9	**20**	600	**500**	560	**620**	900	**533**

Table 8: Curriculum performance on `Qwen2.5-Math-1.5B` with SAE features extracted from different token spans. The indented rows use features from both the prompt and response spans or from the prompt span alone, respectively. **Bold** marks the best result.

Method	Accuracy (%) ↑							Steps to Target ↓						
	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**
GRPO	30.0	53.4	81.1	71.4	26.1	34.8	49.5	40	680	840	800	**780**	800	657
SAERL$_{\text{Curr}}$														
+ Qwen3-1.7B	**40.0**	**56.3**	**83.6**	**72.0**	**27.3**	**35.8**	**52.5**	**20**	580	**520**	**620**	840	**760**	**557**
+ Llama-3.1-8B	33.3	53.8	82.3	71.5	26.2	35.3	50.4	80	**520**	680	780	820	900	630

Table 9: Effect of the SAE source for curriculum construction on `Qwen2.5-Math-1.5B`. The two indented rows use SAEs trained on `Qwen3-1.7B` and `Llama-3.1-8B` activations, respectively. **Bold** marks the best result.

Method	Accuracy (%) ↑							Steps to Target ↓						
	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**	AIME	AMC	GSM8K	MATH	MNV	OLPD	**Avg**
GRPO	30.0	53.4	81.1	71.4	26.1	34.8	49.5	40	680	840	800	**780**	800	657
SAERL$_{\text{Curr}}$														
+ OOD	36.7	53.1	81.7	**72.0**	26.1	35.5	50.9	40	**540**	760	740	900	**740**	620
+ In-domain	**40.0**	**56.3**	**83.6**	**72.0**	**27.3**	**35.8**	**52.5**	**20**	580	**520**	**620**	840	760	**557**

Table 10: Training performance and efficiency of out-of-domain difficulty information for curriculum construction on `Qwen2.5-Math-1.5B`. **Bold** marks the best result.

D.4 Alternative SAE Source

Table 9 compares SAE sources under the same curriculum. The Qwen3 SAE yields stronger accuracy, as expected from its closer match to the Qwen2.5-Math training model. Nevertheless, the Llama-3.1-8B SAE also improves average accuracy and convergence over GRPO, showing that SAE-guided curriculum construction remains effective across model families.

D.5 OOD Difficulty Curriculum

We further examine whether the difficulty signal used for curriculum construction can transfer across datasets. We train the difficulty predictor on 3K labeled samples from DeepScaleR and use the resulting difficulty estimates to construct the SAERL$_{\text{Curr}}$ curriculum on DeepMath. The remaining curriculum construction procedure is kept unchanged.

As shown in Table 10, the OOD difficulty curriculum improves the average accuracy over GRPO, while reducing the average steps to target. Although it does not match the default SAERL$_{\text{Curr}}$ setting, the OOD variant retains improvements in both final performance and training efficiency. These results suggest that the difficulty signal derived from SAE representations exhibits a degree of cross-dataset transferability and can still provide useful guidance for curriculum construction when the predictor is trained on out-of-domain data.

Cluster	Conservative label	Main evidence	Interpretation
0	Derivative-centered calculus	Derivative applications; medium-high difficulty; moderate length	Local change, optimization, and derivative-based transformations.
1	Number-theoretic symbolic reasoning	Congruences; long solutions; medium difficulty	Congruence-style reasoning with symbolic manipulation and extended derivations.
2	Abstract algebra and proof structure	Group theory; high topic entropy; medium-high difficulty	Algebraic structures and proof-oriented reasoning beyond a single subfield.
3	Discrete and combinatorial reasoning	Combinatorics; highest mean difficulty	Counting, construction, and combinatorial proof patterns with high difficulty.
4	Integral and continuous reasoning	Integral applications; long solutions; high difficulty	Integration-related problems with continuous reasoning and longer derivations.
5	Broad calculus and transformations	Derivative-related topics; high entropy; long solutions	A heterogeneous continuous-math group involving functions and transformations.
6	Limits and sequence-style reasoning	Limits; highest top-topic share; lowest entropy	The most topic-concentrated cluster, centered on limits and sequences.
7	High-load limits and analysis	Limits; higher difficulty than Cluster 6; long solutions	More complex and heterogeneous variants of limit or analysis-style reasoning.
8	Short-form elementary algebra	Simple equations; lowest mean difficulty; shortest solutions; highest entropy	A mixed low-to-mid difficulty group with short algebraic structure.
9	Integration and symbolic procedures	Integration techniques; medium difficulty; moderate length	Procedural symbolic transformation, especially integration-related reasoning.

Table 11: Cluster-level semantic audit of SAE activation groups on DeepMath. Summaries are generated by a GPT-5.4-based agent and manually reviewed by the authors. Labels are conservative summaries rather than one-to-one topic annotations.

Feature group	Top 20	Top 100
`sol_mean`	10	36
`prompt_max`	5	26
`prompt_mean`	2	24
`sol_max`	2	11
`meta`	1	3

Table 12: Feature-group composition of the top gain-ranked features in an auxiliary LightGBM model trained for difficulty prediction. This model is used only for interpretability analysis.

E Interpretability Details of SAE-Guided Curriculum Construction

This appendix supplements the interpretability analysis in Section 4.4. We document the curriculum-map protocol, the relation between SAE clusters and external topic annotations, the feature groups that provide predictive signal for difficulty estimation, and the intermediate variables exposed by the curriculum construction procedure.

E.1 Curriculum-Map Protocol

The shared background map in Figure 4 uses a fixed sample of $10{,}000$ DeepMath examples. We preprocess the pooled SAE representation without metadata, retain 50 principal components, and apply t-SNE with perplexity 50 and seed 42. The same coordinates are reused in all three panels. The projection is used only to display where active examples lie; we do not interpret island area or pairwise distance as quantitative evidence.

The curriculum order contains the selected $44{,}510$-sample subset arranged using calibrated difficulty scores and cluster assignments. Its construction uses a batch size of 128, 16-batch difficulty windows, and adjacent tail-8 mixing. The early, middle, and late views correspond to batches 1–32, 159–190, and 317–348 of the curriculum order. Their complete-window mean difficulties are 3.92, 5.86, and 8.22, respectively. The fixed 10K visualization sample intersects these windows in 414, 432, and 395 unique examples; among these intersections, only examples from the four highlighted clusters (C2, C4, C5, and C9) are shown in color. Cluster proportions reported in the main text are computed from all positions in each complete window rather than from the displayed subset.

The curriculum order fixes the sample multiset, difficulty signal, outer-window membership, and mixing budget before RL optimization begins.

High-gain feature	Observed high-activation pattern	Interpretation
sol_mean/50831	Abstract algebra, rings, groups, and proof-heavy problems	Solution-side activation associated with abstract algebraic structure and proof load.
sol_mean/28006	Measure, integration, and high-level analysis	Solution-side pattern related to advanced analysis and integration-style reasoning.
prompt_max/16071	Graphs, geometry, and Asymptote-style prompts	Prompt-side activation capturing localized diagrammatic or formatting cues.
sol_mean/122476	Combinatorics, graphs, and discrete structures	Solution-side signal associated with discrete reasoning and combinatorial structure.
sol_mean/60349	Simple equations and algebraic word problems	Lower- to mid-difficulty algebraic patterns involving short symbolic reasoning.
sol_mean/88548	Congruences, primes, and numeric expressions	Solution-side number-theoretic and numeric reasoning patterns.

Table 13: Semantic audit of individual high-gain features using high-activation samples. Summaries are produced by a GPT-5.4-based agent and manually reviewed by the authors.

E.2 Cluster-Level Structure

We compare the SAE cluster assignments on DeepMath with the dataset-provided topic labels. The overall cluster–topic alignment is low: cluster-topic purity is 0.1095, NMI is 0.0881, and ARI is 0.0394. These results indicate that SAE clusters do not simply reproduce the dataset's topic taxonomy.

We therefore further inspect each cluster using top leaf topics, topic entropy, difficulty distribution, solution length, and representative samples. Table 11 provides a summary of the resulting cluster structure.

Some clusters exhibit clear topical enrichment, such as limits, combinatorics, group theory, and integration. At the same time, clusters also differ in problem format, solution length, symbolic density, proof orientation, and mean difficulty. These summaries should therefore be understood as coarse descriptions of activation-space structure, rather than as one-to-one topic labels.

This distinction is necessary because the external topic taxonomy and the SAE representation characterize different aspects of a sample. Topic labels describe the subject category assigned by the dataset, whereas SAE clusters are formed according to model-internal activation patterns. A cluster may therefore contain multiple mathematical topics while still preserving structure relevant to curriculum construction, such as similar solution profiles, reasoning formats, or difficulty ranges.

E.3 Feature-Level Signals

We next examine which input feature groups provide predictive signal for difficulty estimation. The curriculum pipeline uses prompt-side and solution-side SAE features, together with a small set of metadata features. For feature-group analysis, we train an auxiliary LightGBM model on the same difficulty-labeled subset and rank input columns by gain. Because gradient-boosted trees capture nonlinear feature interactions, this model serves as a probe of which features carry difficulty-relevant information. It is used only for interpretability analysis; the actual curriculum ordering uses the calibrated difficulty scores from the ElasticNet proxy described in Section 2.

Table 12 reports the source distribution of the top-20 and top-100 gain-ranked features.

The top-ranked features come primarily from SAE activations rather than metadata. Among the top-20 features, 19 are SAE-derived and 1 is metadata; among the top-100 features, 97 are SAE-derived and 3 are metadata. Within these features, solution-side mean features form the largest group, which is consistent with sustained solution-side activations providing correlational signals for difficulty estimation. Prompt-side max features also appear frequently, indicating that localized strong activations in the problem statement provide additional predictive cues.

To complement the feature-group statistics, we further inspect high-activation samples for several high-gain SAE features. Table 13 summarizes the semantic tendencies observed in these samples.

These high-activation samples exhibit several recurring patterns, including abstract algebra, measure and integration, discrete combinatorics, geometric formats, elementary algebra, and number-

theoretic expressions. These descriptions indicate semantic tendencies only. SAE features are not strictly monosemantic: the same SAE feature may be activated in different contexts, and the same mathematical pattern may be distributed across multiple SAE features.

E.4 PROCEDURE-LEVEL INSPECTABILITY

The final curriculum order is constructed from a sequence of explicit intermediate variables. For each sample, the pipeline records its SAE representation, metadata, cluster assignment, raw difficulty prediction, calibrated difficulty score, batch assignment, and, when moderate mixing is applied, its tail-swap partner. These variables allow researchers to inspect how a sample moves from the representation space to its final position in the curriculum.