GUIDED SPARSE AUTOENCODERS FOR INTERPRETING REASONING FEATURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) increasingly solve complex tasks through multi-step Chain-of-Thought (CoT) reasoning, yet the internal representations supporting these trajectories remain difficult to interpret. Sparse Autoencoders (SAEs) decompose dense activations into sparse latent features, but standard reconstruction-sparsity only training often misses subtle reasoning-related features because they contribute negligibly to overall activation reconstruction variance. Consequently, important reasoning primitives do not naturally occupy identifiable latent directions. We propose the *Guided Sparse Autoencoder* (G-SAE), a partitioned latent-space architecture that injects supervision directly into sparse representation learning. A subset of latent dimensions is jointly supervised for step reasoning correctness, while the remaining dimensions retain an unsupervised role in reconstructing broader activation information. By combining labeled ProcessBench activations with unlabeled activations, G-SAE merges targeted correctness pressure with broad reconstruction coverage. Evaluations across different model scales demonstrate that supervised latent partitioning improves correctness prediction while maintaining reconstruction fidelity.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated strong performance on tasks requiring multi-step reasoning via Chain-of-Thought (CoT) trajectories (Wei et al., 2022; Kojima et al., 2022). While these trajectories expose intermediate reasoning steps, the internal representations supporting individual decisions remain opaque.

Mechanistic interpretability faces a central challenge: isolating representations corresponding to genuine computational mechanisms of reasoning rather than features that merely correlate with surface text forms (Ma et al., 2026). Sparse Autoencoders (SAEs) have emerged as an important tool for decomposing dense model activations into sparse latent features (Huben et al., 2024; Bricken et al., 2023). However, conventional SAE training is driven strictly by reconstruction and sparsity objectives. Because reasoning-related signals may account for only a small fraction of activation variance, reconstruction-only objectives can split, absorb, or discard them Chanin et al. (2025); indeed, SAE features often fail to outperform raw activations for probing Kantamneni et al. (2025)

This motivates two key requirements for reasoning-oriented SAE representations. First, representations must operate at an appropriate granularity; token-level features capture lexical and syntactic properties, whereas multi-step reasoning requires step-level representations (Yang et al., 2026). Second, if the goal is to isolate a specific reasoning property for downstream analysis or intervention, that property should influence representation learning directly rather than being evaluated only post-hoc.

However, supervising the entire SAE devotes its latent capacity to the target property at the expense of the rest of the activation space: in our experiments, a fully supervised SAE increases normalized reconstruction error by 0.12–0.75 over a reconstruction-only SAE, compared with at most 0.037 for our partitioned approach (Appendix A.1).

We therefore restrict direct correctness supervision to a dedicated latent partition. This motivates the *Guided Sparse Autoencoder* (G-SAE), which splits its latent space into supervised and unsupervised

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.

subsets, as formally described in Section3.2. Reconstruction occurs over the full latent space, while correctness supervision targets only the supervised subset.

G-SAE balances reasoning features discovery through the supervision with general reconstructive capacity. Our contributions are summarized as follows:

1. We identify and address the limitation wherein reconstruction-only SAEs miss reasoning-related features due to their minimal variance contribution, and introduce explicit supervision guidance to capture them.
2. We formulate a joint objective combining full-space reconstruction, targeted latent supervision, and a diversity regularizer intended to encourage feature specialization.
3. We integrate labeled ProcessBench steps Zheng et al. (2025) with large-scale unlabeled Math-Shepherd steps Wang et al. (2024) to couple reasoning-related features discovery with maintained reconstruction.
4. We perform comprehensive evaluations across six LLMs across three scales (0.5B, 1.5B, and 7B), demonstrating improved correctness prediction, and preserved reconstruction fidelity.

2 Related Work

2.1 Mechanistic Interpretability and Sparse Autoencoders

Mechanistic interpretability aims to identify the internal representations and computational mechanisms that explain model behavior (Sharkey et al., 2025).

Sparse Autoencoders have emerged as a dominant paradigm for disentangling polysemantic neurons into more monosemantic, human interpretable features by decomposing the model activations into sparse features (Huben et al., 2024; Bricken et al., 2023). SAE variants leverage regularizations such as L1 penalties and Top-K activation (Templeton et al., 2024; Gao et al., 2025; Bussmann et al., 2024; Rajamanoharan et al., 2024).

2.2 Sparse Autoencoders for Reasoning in LLMs

The use of chain-of-thought (CoT) reasoning (Wei et al., 2022) has motivated work on interpreting the internal representations underlying multi-step reasoning. Recent approaches use SAE features, probing, and feature-based interventions to identify representations associated with reasoning behaviors or strategies (Fang et al., 2026). However, predictive association alone does not establish that a feature is causally involved in the computation: Ma et al. (2026) show that many features distinguishing reasoning-mode from direct-answer generation track surface cues rather than reasoning itself. We differ in both target and test: we target the *correctness* of individual steps using ProcessBench labels, and we test causal relevance by steering features during generation and scoring the resulting reasoning with a process reward model.

Closest in motivation is the Step-Level SAE (SSAE) (Yang et al., 2026), which aggregates reasoning steps and conditions the encoder on preceding context. SSAE is fully unsupervised and probes correctness only post hoc, using final-answer verification or an LLM judge, making it closer to our reconstruction-only baseline than to G-SAE, which supervises a latent partition with step-level validity labels. We therefore compare against a reconstruction-only SAE matching our architecture, data, and sparsity mechanism.

2.3 Supervised Representation Learning

Concept Bottleneck Models (CBMs) introduce supervision over predefined, human-interpretable concepts and use these concepts as an intermediate representation for prediction (Koh et al., 2020). G-SAE differs from this formulation in that its latent features are not predefined semantic concepts. Instead, they are learned directly from LLM activations through sparse dictionary learning, while step-level correctness provides behavioral supervision to only a designated subset of latent dimensions. The remaining dimensions are learned solely through sparse reconstruction. Correctness therefore acts as a *behavioral inductive bias* on feature discovery rather than as a predefined concept bottleneck.

Overall, existing approaches primarily provide unsupervised sparse feature discovery, step-level representation learning, or post-hoc identification of behavior-associated features. G-SAE complements these approaches by incorporating behavioral supervision directly into a partitioned sparse dictionary, guiding a controlled subset of features toward reasoning-relevant information while preserving unsupervised capacity for broader feature discovery.

3 Our Method G-SAE

3.1 Problem Formulation

Let $\mathbf{h}_i^{(\ell)} \in \mathbb{R}^d$ denote the hidden state of the LLM at layer ℓ and token position i. Consider a problem x whose solution is decomposed into sequential reasoning steps $s_1, \ldots, s_T$. For step s_t, let $s_{\leq t} = (x, s_1, \ldots, s_t)$ denote the complete prefix ending at s_t, including the problem statement. We represent each reasoning step using the hidden activation at the last token of its prefix:

$$\bar{\mathbf{h}}_s = \mathbf{h}^{(\ell)}(s_{\leq t})_{\text{last}}, \tag{1}$$

where $s = s_t$ and ℓ denotes the target layer. Because the model is autoregressive, $\bar{\mathbf{h}}_s$ depends on the problem and all preceding steps, but not on later ones: it summarizes the model's internal state immediately after processing s_t in context, which is the state from which the next step would be generated.

We use the last-token activation because mean-pooling over a step's tokens can dilute the model's final state after that step, weakening both feature identification and steering. We encode the full prefix $s_{\leq t}$ rather than the step alone, so that $\bar{\mathbf{h}}_s$ reflects the context, including all preceding reasoning, in which s_t is generated.

Each reasoning step s is associated with a binary correctness label $y_s \in \{0, 1\}$ from ProcessBench (Zheng et al., 2025), which defines errors as mathematical, logical, conceptual, or completeness mistakes. Labels reflect *local validity* relative to immediate preceding context rather than the final outcome (Zheng et al., 2025). Because the benchmark targets the earliest-occurring error in a chain, it provides a rigorous testbed for step-level reasoning interpretability.

3.2 G-SAE architecture

Given the step representation $\bar{\mathbf{h}}_s$, the encoder first produces a dense pre-activation vector

$$\mathbf{a}_s = W_{\text{enc}}\bar{\mathbf{h}}_s + \mathbf{b}_{\text{enc}}, \tag{2}$$

where $W_{\text{enc}} \in \mathbb{R}^{m \times d}$, $\mathbf{b}_{\text{enc}} \in \mathbb{R}^m$, and m is the total latent dimensionality. Unlike a conventional sparse autoencoder that applies a single sparsifying transformation to the entire latent space Bricken et al. (2023); Huben et al. (2024),we partition the encoder output into two non-overlapping subspaces (Figure 1):

$$\mathbf{a}_s = [\mathbf{a}_s^{\text{sup}}; \mathbf{a}_s^{\text{unsup}}], \tag{3}$$

where $\mathbf{a}_s^{\text{sup}} \in \mathbb{R}^{k_{\text{sup}}}$ contains the supervised latent dimensions and $\mathbf{a}_s^{\text{unsup}} \in \mathbb{R}^{m-k_{\text{sup}}}$ contains the unsupervised dimensions. In the supervised partition we apply a sigmoid-weighted ReLU:

$$\mathbf{z}_s^{\text{sup}} = \sigma(\mathbf{a}_s^{\text{sup}}) \odot \text{ReLU}(\mathbf{a}_s^{\text{sup}}), \tag{4}$$

where $\odot$ denotes element-wise multiplication.

We avoid Top-K here because its competitive selection could persistently exclude some supervised units from the correctness gradient; with the sigmoid-weighted ReLU, each unit is active whenever its own pre-activation is positive, so all supervised units remain trainable.

Following the Top-K SAE formulation of Gao et al. (2025), and consistent with prior work that explicitly applies a ReLU after Top-K selection (Gujral et al., 2025), we define the unsupervised activations as :

$$\mathbf{z}_s^{\text{unsup}} = \text{ReLU}\left(\text{TopK}\left(\mathbf{a}_s^{\text{unsup}}, K_{\text{s}}\right)\right). \tag{5}$$

Here, K_{s} specifies the number of encoder pre-activations retained by the Top-K operator. The full latent code is the concatenation of both partitions,

$$\mathbf{z}_s = [\mathbf{z}_s^{\text{sup}}; \mathbf{z}_s^{\text{unsup}}]. \tag{6}$$

This partitioning allows the supervised dimensions to specialize toward correctness-related features while leaving the remaining latent capacity available for unsupervised feature discovery.

3.3 Shared Decoder and Reconstruction

The complete latent representation is decoded using a shared linear decoder:

$$\hat{\mathbf{h}}_s = W_{\text{dec}}\mathbf{z}_s + \mathbf{b}_{\text{dec}}, \tag{7}$$

where $W_{\text{dec}} \in \mathbb{R}^{d\times m}$ and $\mathbf{b}_{\text{dec}} \in \mathbb{R}^d$. We optimize a normalized reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \frac{1}{B}\sum_{s=1}^{B} \frac{\left\|\hat{\mathbf{h}}_s - \bar{\mathbf{h}}_s\right\|_2^2}{\left\|\bar{\mathbf{h}}_s\right\|_2^2 + \epsilon}, \tag{8}$$

where B denotes the batch size. The normalization reduces the dependence of the reconstruction objective on the magnitude of individual hidden representations.

3.4 Correctness Supervision

To guide the supervised partition toward reasoning-relevant information, we attach a linear correctness head producing the logit

$$\ell_s = W_{\text{cls}}\mathbf{z}_s^{\text{sup}} + b_{\text{cls}}, \tag{9}$$

where $W_{\text{cls}} \in \mathbb{R}^{1\times k_{\text{sup}}}$ and $b_{\text{cls}} \in \mathbb{R}$. The head is deliberately linear, so that correctness prediction directly tests whether correctness is linearly decodable from the supervised partition, consistent with the hypothesis that meaningful behaviors are often linearly represented (Park et al., 2024). Let $\mathcal{B}_{\text{lab}}$ denote the labeled steps in a batch; we train the head with binary cross-entropy on $p_s = \sigma(\ell_s)$:

$$\mathcal{L}_{\text{sup}} = -\frac{1}{|\mathcal{B}_{\text{lab}}|}\sum_{s\in\mathcal{B}_{\text{lab}}} \left[y_s \log p_s + (1-y_s)\log(1-p_s)\right]. \tag{10}$$

The head receives only $\mathbf{z}_s^{\text{sup}}$, so the correctness labels do not directly constrain the unsupervised partition. Because the shared decoder reconstructs from the full code $\mathbf{z}_s$ (Section 3.3), $\mathbf{z}_s^{\text{sup}}$ receives gradient from both $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{sup}}$, whereas $\mathbf{z}_s^{\text{unsup}}$ is shaped only by $\mathcal{L}_{\text{rec}}$. This dual pressure lets the supervised partition specialize toward correctness while still contributing to reconstruction, rather than being carved out of it.

3.5 Diversity Regularization

Because multiple supervised dimensions may learn redundant representations, we add a regularizer. For a batch of B steps, labeled or unlabeled, we stack the supervised activations row-wise,

$$Z^{\text{sup}} = \begin{bmatrix} (\mathbf{z}_1^{\text{sup}})^\top \\ \vdots \\ (\mathbf{z}_B^{\text{sup}})^\top \end{bmatrix} \in \mathbb{R}^{B\times k_{\text{sup}}}, \tag{11}$$

where the subscript indexes steps within the batch, and normalize each column, i.e., the activation pattern of one supervised unit across the batch:

$$\tilde{Z}_{:,j} = \frac{Z_{:,j}^{\text{sup}}}{\|Z_{:,j}^{\text{sup}}\|_2 + \epsilon}, \tag{12}$$

where $\epsilon > 0$ is a small constant for numerical stability. The matrix

$$C = \tilde{Z}^\top \tilde{Z} \in \mathbb{R}^{k_{\text{sup}}\times k_{\text{sup}}} \tag{13}$$

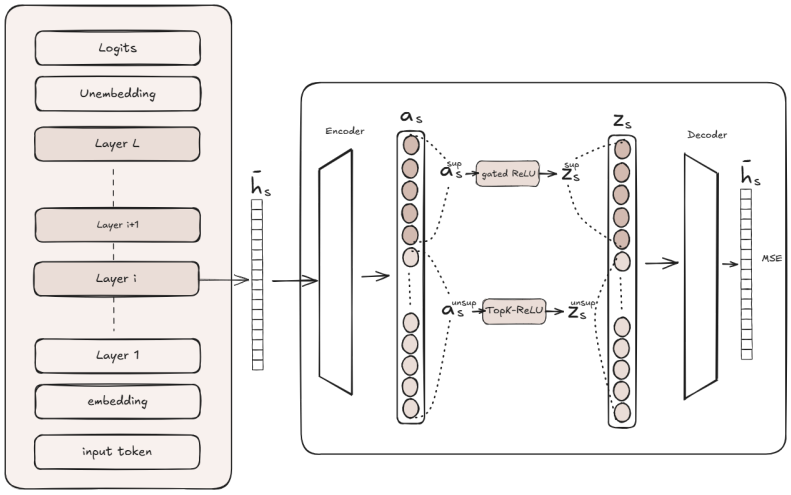


Figure 1: Overview of the Guided Sparse Autoencoder (G-SAE). Step-level representations extracted from layer ℓ of an LLM are encoded and split into two partitions. The supervised partition (k_{sup} units, sigmoid-weighted ReLU) feeds a linear head trained with binary cross-entropy to predict step correctness, and is additionally regularized by the diversity loss. The unsupervised partition uses Top-K sparsity. Both partitions are decoded jointly by a shared decoder trained with normalized MSE reconstruction.

contains the cosine similarities between these activation patterns. Since supervised activations are non-negative, $C_{jj'}$ is large when units j and j' are active on the same steps. We penalize the off-diagonal entries:

$$\mathcal{L}_{\text{div}} = \frac{1}{k_{\text{sup}}(k_{\text{sup}} - 1)} \sum_{j \neq j'} C_{jj'}^2. \tag{14}$$

This regularizer is intended to encourage the supervised units to capture complementary rather than redundant patterns.The effect of λ_{div} on correctness discrimination is model-dependent (Appendix A.5).

3.6 Joint Training Objective

The complete G-SAE objective combines reconstruction, correctness supervision, and diversity:

$$\boxed{\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{sup}} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}} \tag{15}$$

where λ_{div} controls the relative contribution of diversity to the loss. Sparsity in the unsupervised partition is enforced structurally through the Top-K activation rather than through an additional sparsity penalty.

4 Experimental Evaluation

We evaluate G-SAE on step-level reasoning representations extracted from autoregressive language models. For each reasoning step s_t, we feed the complete prefix $s_{\leq t}$ to the model and extract the hidden activation at the last token of the target layer ℓ, as defined in Eq. 1.For every model, ℓ is the middle layer ($\ell = L/2$ for a model with L layers): layer 12 of 24 for Qwen2.5-0.5B and layer 14 of 28 for all 1.5B and 7B models. The SAE dictionary size is set to $m = 6d$, where d is the model's hidden dimension (Table 1).

4.1 Training Data

G-SAE is trained on two complementary streams of reasoning-step representations, both extracted as in Eq. 1: a *supervised stream* of labeled steps from ProcessBench (Zheng et al., 2025) and an

unsupervised stream of unlabeled steps from Math-Shepherd (Wang et al., 2024). Both streams are used jointly during training, allowing G-SAE to incorporate correctness supervision while retaining capacity for general feature discovery.

Step extraction and splitting. Each solution is encoded incrementally with the frozen LLM, so that $\bar{\mathbf{h}}_s$ is conditioned on the problem and all preceding steps, as during generation. Because solutions to the same problem share this prefix, we split all data at the problem level: no problem appears in more than one split.

Supervised stream. The supervised stream consists of step activations paired with correctness labels; it contributes to both the reconstruction and correctness objectives, encouraging the supervised partition to capture correctness-related features. We build it from ProcessBench, which contains 3,400 solutions across four difficulty tiers (GSM8K, MATH, OlympiadBench, Omni-MATH), each annotated with the index e of its first erroneous step ($e = -1$ if fully correct). We derive step labels as

$$y_s = \begin{cases} 1 & \text{if } e = -1 \text{ or } t < e, \\ 0 & \text{if } t = e, \\ \text{excluded} & \text{if } t > e, \end{cases} \tag{16}$$

since steps after the first error are unannotated and follow a corrupted context.Solutions are split 70/10/20 into training, validation and evaluation sets.Because correct steps greatly outnumber erroneous ones, we rebalance the training and validation sets to equal numbers of each class within every (step position, difficulty) group.The two classes thus share the same distribution of positions and difficulties, and can only be distinguished by step content. Correctness supervision is restricted to GSM8K and MATH (1,138 training and 166 validation steps), leaving OlympiadBench and Omni-MATH unseen for out-of-distribution evaluation.The full processing pipeline is illustrated in Appendix A.2.

Unsupervised stream. The unsupervised stream contributes only to reconstruction, allowing the unsupervised partition to learn general features without label constraints. We sample 50,000 GSM8K and MATH solutions from Math-Shepherd and embed every step; its automatically estimated step labels are not used. This yields 262,961 training and 29,196 validation activations.

Evaluation set. The ProcessBench evaluation split contains 680 held-out, unbalanced solutions (456 with an error; 5,189 steps), labeled with Eq. 16. Probe AUC is computed on its labeled steps, with GSM8K and MATH as the in-distribution set and OlympiadBench and Omni-MATH as the out-of-distribution set.

4.2 Evaluation Protocol

We evaluate *representation fidelity* (normalized MSE and explained variance of reconstructions), *feature utility* (ROC-AUC of a post-hoc linear probe on frozen SAE features, identical for all SAEs, on in-distribution GSM8K/MATH and out-of-distribution OlympiadBench/Omni-MATH), and *causal relevance* (effect of single-feature steering on generated reasoning, Section 4.6).

4.3 Baselines

Our primary baseline is a reconstruction-only sparse autoencoder with the same activation extraction, latent dimensionality, sparsity level, and training budget as G-SAE. Both models are trained on the same combination of **Math-Shepherd** and **ProcessBench**; however, the baseline uses both datasets only for reconstruction and receives no correctness supervision. It employs the same TopK-ReLU activation mechanism.For a fair comparison of feature utility, correctness is evaluated *post hoc* using the same linear probing procedure on frozen SAE representations for both SAEs. Thus, the primary difference between the two models is the correctness-guided training objective and diversity regularization used by G-SAE, rather than the training data, representation capacity, or sparsity mechanism.

4.4 CORRECTNESS DISCRIMINATION RESULTS

Table 1 reports OOD AUC, NMSE, and EV across six LLMs. All G-SAE results in this table use $\lambda_{\text{div}} = 0.5$. G-SAE configurations, including the extraction layer ℓ, hidden dimension d, SAE size m, unsupervised sparsity budget K_s, and supervised partition size k_{sup}, are provided in Appendix A.3, Table 5. We study the sensitivity of G-SAE to the diversity weight λ_{div} in Appendix A.5 and to the supervised partition ratio k_{sup}/m in Appendix A.4.

The reconstruction-only SAE (Unsup) often loses correctness information present in the raw activations it was trained to reconstruct: Unsup AUC falls below Raw AUC on five of six models, with drops as large as 0.093 (Qwen2.5-7B: $0.696 \rightarrow 0.603$). This is consistent with evidence that unsupervised SAE training can fragment or absorb a concept's signal across latents rather than isolating it (Chanin et al., 2025), and supports our premise that reconstruction- and sparsity-only training can treat correctness-relevant variance as low priority.

G-SAE outperforms both the reconstruction-only SAE and the raw-activation probe on all six models, improving OOD AUC by 0.035–0.119 over Unsup. Correctness-guided supervision thus does not merely prevent the loss of correctness information, but can surface correctness structure that is diffuse or entangled in the raw representation.

This gain comes at a modest reconstruction cost: normalized MSE increases by 0.017–0.037 and explained variance decreases by 0.008–0.030 across models. The cost is paid with a small fraction of the dictionary: the supervised partition occupies 0.4% of the latents for five of the six models and 2.5% for DeepSeek-R1-Distill-Qwen-1.5B. Overall, correctness supervision trades a small loss in reconstruction for a substantial gain in correctness discrimination, while leaving the large majority of the dictionary unsupervised.

Table 1: Correctness discrimination and reconstruction fidelity across six LLMs. Bold marks the best OOD AUC per model.

LLM	OOD AUC ↑				Norm MSE ↓			Expl. Var. ↑		
	Raw	Unsup	GSAE	Δ_{AUC}	Unsup	GSAE	Δ_{MSE}	Unsup	GSAE	Δ_{EV}
Qwen2.5-0.5B	0.656	0.64	**0.6940**	+0.054	0.206	0.243	+0.037	0.81	0.78	-0.03
Qwen2.5-Math-1.5B	0.612	0.608	**0.660**	+0.052	0.367	0.384	+0.017	0.675	0.667	-0.008
DeepSeek-R1-Distill-Qwen-1.5B	0.596	0.641	**0.708**	+0.067	0.274	0.306	+0.032	0.727	0.698	-0.029
DeepSeek-R1-Distill-Qwen-7B	0.664	0.645	**0.68**	+0.035	0.202	0.225	+0.023	0.836	0.821	-0.015
Qwen2.5-7B	0.696	0.603	**0.722**	+0.119	0.351	0.377	+0.026	0.719	0.700	-0.019
Qwen2.5-Math-7B	0.674	0.67	**0.720**	+0.05	0.455	0.474	+0.019	0.601	0.583	-0.018

4.5 FEATURE INTERPRETATION

To characterize learned features, we label each feature by inspecting its $M{=}20$ highest and $M{=}5$ lowest-activating inputs (with `[PROBLEM]`/`[STEP N:]` scaffolds removed and at most two examples per problem) and prompting `llama3.1:70b` for a 3–6 word description. Labels are mapped by a deterministic keyword classifier into *operational*, *semantic*, *structural*, or *other* (Table 2).The G-SAE supervised partition is substantially enriched for operational features (69.8%) relative to its own unsupervised partition (27.5%) and the standalone Unsup SAE (40.5%), while unsupervised partitions skew toward semantic features. This asymmetry indicates that correctness supervision does not simply select existing features more effectively, but shifts what the supervised dimensions represent, from the problem's mathematical content toward the actions taken on it, which is consistent with the operational, step-level notion of correctness used by ProcessBench labels (Section 4.1) rather than a topic or domain-level signal. Representative examples are in Appendix A.6.

4.6 CAUSAL VALIDATION

Predictive association does not establish that a feature drives model behavior. To test causal relevance, we steer single features during generation and measure the effect on reasoning quality with a process reward model (PRM). We use the mean per-step PRM score rather than final-answer accuracy because both our features and the intervention operate at the step level: the PRM scores every generated step, so it captures how steering changes the quality of the whole trajectory.

Table 2: Distribution of feature types across partitions.

Partition / SAE	Operational	Semantic	Structural	Other
G-SAE supervised	69.8%	27.3%	0.6%	2.3%
G-SAE unsupervised	27.5%	64.5%	6.5%	1.5%
G-SAE total	47.0%	47.3%	3.8%	1.9%
Unsupervised SAE	40.5%	50.0%	7.5%	2.0%

4.6.1 METHOD

We perform single-feature activation steering: during generation, we add the decoder direction of exactly one feature to the residual stream. This gives clean attribution: any observed effect is due to one feature rather than a combination of several. Summing or weighting several decoder columns would perturb the residual stream along a direction that corresponds to no individual feature, making the behavioral change difficult to attribute.

For a feature f with SAE decoder column $\mathbf{d}_f \in \mathbb{R}^{d_{\text{model}}}$, we convert $\mathbf{d}_f$ to the model's raw activation space using the per-dimension normalisation statistics, and register a forward hook at the extracted layer ℓ of the generation model. The intervention is

$$\mathbf{h}'_t = \mathbf{h}_t + \alpha_t \cdot \mathbf{d}_f, \qquad \alpha_t = \alpha \cdot \left(\frac{1}{1+\omega t}\right)^p, \tag{17}$$

where t indexes generated tokens ($t = 0$ for the prompt), not reasoning steps; $\mathbf{d}_f$ is the unit-normalized decoder direction; $\alpha = \alpha_{\text{rel}}\,\overline{\|\mathbf{h}\|}$ is the base steering strength, with $\overline{\|\mathbf{h}\|}$ the mean residual norm at layer ℓ computed once over the evaluation prompts; and ω, p control the decay. After each perturbation, $\mathbf{h}'_t$ is rescaled to the original norm $\|\mathbf{h}_t\|$, so the intervention changes only the direction of the hidden state.

We steer every position rather than only step-final ones because step boundaries are unknown during generation, and because the residual stream is a running state whose step-final values are shaped by earlier tokens. Although the SAE was trained on step-final activations, $\mathbf{d}_f$ is a valid direction in the same activation space at any position. Following Xie (2025), the coefficient decays with t, concentrating the intervention on the prompt and the first generated tokens: with ω=2 and p=3, it falls below 1% of its initial value after two tokens. A constant perturbation degrade the coherence of the generation. Step structure is used only after generation, when the output is split into steps for PRM scoring. Hyperparameters are selected on a held-out pilot set (Appendix A.7).

We compare *steered* (α>0), *negated* (α<0), and *random* (matched-magnitude random direction) conditions, generating 4 solutions per problem over the same 100 ProcessBench (`olympiad`/`omni`) problems, scored by Qwen2.5-Math-PRM-7B (Zhang et al., 2025) with 95% paired bootstrap intervals (10,000 resamples). For each partition (G-SAE supervised, G-SAE unsupervised, standalone Unsup SAE) we test the three features with highest per-feature correctness AUC, holding layer, token placement, decay schedule, α range, and problem set fixed across all conditions. Results in 3 use Qwen2.5-7B.

4.6.2 RESULTS

Table 3 reports the results for Qwen2.5-7B. Only features from the G-SAE supervised partition show significant effects. Features 2 and 6 raise the PRM score when steered and lower it when negated. Feature 13 is also significant in both directions, but opposite to its predicted direction. None of the six features from the G-SAE unsupervised partition or the Unsup SAE shows a significant effect (deltas from -0.0077 to $+0.0073$), and neither does the random-direction control.

Across the 27 tests in the table, about one significant result would be expected by chance at the 95% level; the six significant results all belong to supervised features, and each feature's two effects have opposite signs, as expected from a genuine directional effect.

Illustrative examples that shows the kind of change steering can produce can be found in Appendix A.8).

Table 3: Single-feature steering on Qwen2.5-7B (layer 14). Features are selected by per-feature correctness AUC within each partition. Δ is the mean PRM-score change relative to the unsteered baseline; bold with * marks a 95% paired bootstrap CI excluding zero.

Partition	Feature	AUC	Δ steer	Δ negate	Δ random
G-SAE sup (7B)	2	0.739	$\mathbf{+0.0284}^*$	$\mathbf{-0.0198}^*$	$+0.0081$
G-SAE sup (7B)	6	0.714	$\mathbf{+0.0100}^*$	$\mathbf{-0.0231}^*$	$+0.0082$
G-SAE sup (7B)	13	0.809	$\mathbf{-0.0190}^*$	$\mathbf{+0.0102}^*$	$+0.0084$
G-SAE unsup (7B)	1309	0.599	-0.0017	$+0.0034$	$+0.0081$
G-SAE unsup (7B)	9019	0.600	$+0.0071$	-0.0077	$+0.0084$
G-SAE unsup (7B)	10083	0.600	-0.0041	$+0.0032$	$+0.0081$
Unsup SAE (7B)	1549	0.703	-0.0043	$+0.0039$	$+0.0082$
Unsup SAE (7B)	1735	0.702	$+0.0060$	$+0.0027$	$+0.0081$
Unsup SAE (7B)	1818	0.675	$+0.0073$	-0.0067	$+0.0080$

Correctness AUC does not determine whether or in which direction a feature has a steering effect. Feature 1549 (Unsup SAE, AUC 0.703) and Feature 6 (0.714) have nearly identical AUCs, but only Feature 6 changes the PRM score; Feature 13, which has the highest AUC (0.809), moves the score against its predicted direction. We conjecture that correctness supervision sharpens the features relevant to correctness, rather than merely increasing their predictiveness.

4.7 Discussion

Our results suggest that behavioral supervision does more than improve linear decodability of correctness,it appears to select feature directions that are more causally load-bearing. Only supervised-partition features produced significant steering effects (Table 3), despite comparable AUC to several unsupervised features, showing that predictive association and causal relevance can dissociate, consistent with concerns raised about post-hoc SAE features. Directly supervising a small subset of the dictionary is one way to close this gap without supervising the whole model or relying on post-hoc probing alone.

4.8 Limitations

Our evaluation is restricted to mathematical reasoning (GSM8K, MATH, OlympiadBench, Omni-MATH); it is unclear whether correctness-guided partitioning transfers to domains where correctness is less crisply defined than in ProcessBench's annotation scheme. In Causal validation we tested three features per partition and a single PRM (Qwen2.5-Math-PRM-7B (Zhang et al., 2025)) as the quality signal; both the feature sample size and the dependence on one reward model's biases limit how strongly the causal claims generalize. The correctness head is linear by design (Section 3.4), which lets us directly test linear decodability but may underestimate correctness information that is only nonlinearly represented. Finally, in our qualitative steering there is an example (Appendix A.8) that raises PRM score without reaching a final answer, illustrating that our causal metric captures step-level reasoning quality rather than end-to-end task accuracy; we treat the former as the object of study here rather than a proxy for the latter.

5 Conclusion

We introduced G-SAE, a sparse autoencoder that partitions its latent space so that a small subset of dimensions is directly supervised for step-level reasoning correctness while the remainder retains unsupervised reconstruction capacity. Across six LLMs, correctness supervision improved out-of-distribution correctness prediction at modest reconstruction cost, and unlike comparably predictive unsupervised features showed measurable causal effects under single-feature activation steering. These results support the view that reasoning-relevant features benefit from targeted supervision during dictionary learning rather than purely post-hoc identification, and suggest partitioned supervision as a practical way to make specific behavioral properties more accessible to intervention without sacrificing an SAE's general-purpose reconstruction role.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, polishing the paper text, code refactoring, and identifying potential typos. We have not used generative AI tools for idea generation or mathematical derivations, and user study data collection tasks are not applicable to this work. Additionally, we used generative AI tools for drafting preliminary plotting scripts. We have reviewed all AI-assisted work. All LLM-generated text was thoroughly edited to ensure scientific accuracy, and suggested code was manually unit-tested and verified by two authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 fully specifies the G-SAE architecture and training objective, and Section 4.1 and Appendix A.2 describe data extraction, labeling, problem-level splitting and class balancing for ProcessBench and Math-Shepherd, both of which are publicly available. Per-model configurations are listed in Appendix A.3. The steering procedure is given in Section 4.6.1, with hyperparameters, their selection protocol and sampling settings in Appendix A.7. All LLMs and the process reward model (Qwen2.5-Math-PRM-7B) are publicly available checkpoints.

REFERENCES

Trenton Bricken, Adly Templeton, Rose Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS 2024)*, 2024.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2025. URL https://arxiv.org/abs/2409.14507.

Yi Fang, Wenjie Wang, Mingfeng Xue, Boyi Deng, Feng Xu, Dayiheng Liu, and Fuli Feng. Controllable llm reasoning via sparse autoencoder-based steering. *arXiv preprint arXiv:2601.03595*, 2026.

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. In *International Conference on Learning Representations (ICLR)*, 2025.

Onkar Gujral, Mihir Bafna, Eric Alm, and Bonnie Berger. Sparse autoencoders uncover biologically interpretable features in protein language model representations. *Proceedings of the National Academy of Sciences*, 122(34):e2506316122, 2025. doi: 10.1073/pnas.2506316122.

Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 7827–7845, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/1fa1ab11f4bd5f94b2ec20e794dbfa3b-Paper-Conference.pdf.

Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing, 2025. URL https://arxiv.org/abs/2502.16681.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5338–5348, 2020.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

George Ma, Zhongyuan Liang, Irene Y. Chen, and Somayeh Sojoudi. Do sparse autoencoders identify reasoning features in language models? In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

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

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, William Saunders, Eric J. Michaud, Stephen Casper, Max Tegmark, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. Open problems in mechanistic interpretability. *Transactions on Machine Learning Research*, 2025-September:1–89, 2025.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024. URL https://arxiv.org/abs/2312.08935.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Jiaqing Xie. A comparative analysis of sparse autoencoder and activation difference in language model steering. *arXiv preprint arXiv:2510.01246*, 2025.

X. Yang, J. Liu, Y. Lai, H. Xu, Z. Huang, and N. Miao. Step-level sparse autoencoder for reasoning process interpretation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

A Appendix

A.1 Comparison with a Fully Supervised SAE

We compare G-SAE with a fully supervised SAE (FSup), in which the correctness head is applied to all m latents, so that every latent is trained with both $\mathcal{L}_{\text{rec}}$ and $\mathcal{L}_{\text{sup}}$. All three SAEs use the configurations of Table 5.

Table 4 shows that full supervision improves correctness discrimination over the reconstruction-only SAE on every model, but at a large reconstruction cost: relative to Unsup, FSup increases normalized MSE by 0.121–0.756 and decreases explained variance by 0.172–0.666, compared with 0.017–0.037 and 0.008–0.030 for G-SAE. G-SAE's OOD AUC is nonetheless comparable to FSup's: it is higher on three models and lower on the other three, by at most 0.030. Restricting supervision to a small partition thus retains the correctness gains of full supervision while largely preserving reconstruction.

Table 4: Reconstruction-only SAE (Unsup), fully supervised SAE (FSup) and G-SAE, with the configurations of Table 5.

LLM	OOD AUC ↑			Norm MSE ↓			Expl. Var. ↑		
	Unsup	FSup	GSAE	Unsup	FSup	GSAE	Unsup	FSup	GSAE
Qwen2.5-0.5B	0.640	0.720	0.694	0.206	0.962	0.243	0.810	0.144	0.780
Qwen2.5-Math-1.5B	0.608	0.689	0.660	0.367	0.512	0.384	0.675	0.485	0.667
DeepSeek-R1-Distill-Qwen-1.5B	0.641	0.681	0.708	0.274	0.581	0.306	0.727	0.411	0.698
DeepSeek-R1-Distill-Qwen-7B	0.645	0.710	0.680	0.202	0.523	0.225	0.836	0.536	0.821
Qwen2.5-7B	0.603	0.704	0.722	0.351	0.607	0.377	0.719	0.251	0.700
Qwen2.5-Math-7B	0.67	0.704	0.720	0.455	0.597	0.474	0.601	0.429	0.583

A.2 PROCESSBENCH DATA PROCESSING

Figure 2 traces how the 3,400 ProcessBench solutions are converted into the supervised training and validation sets and the held-out evaluation set used in Section 4.1.

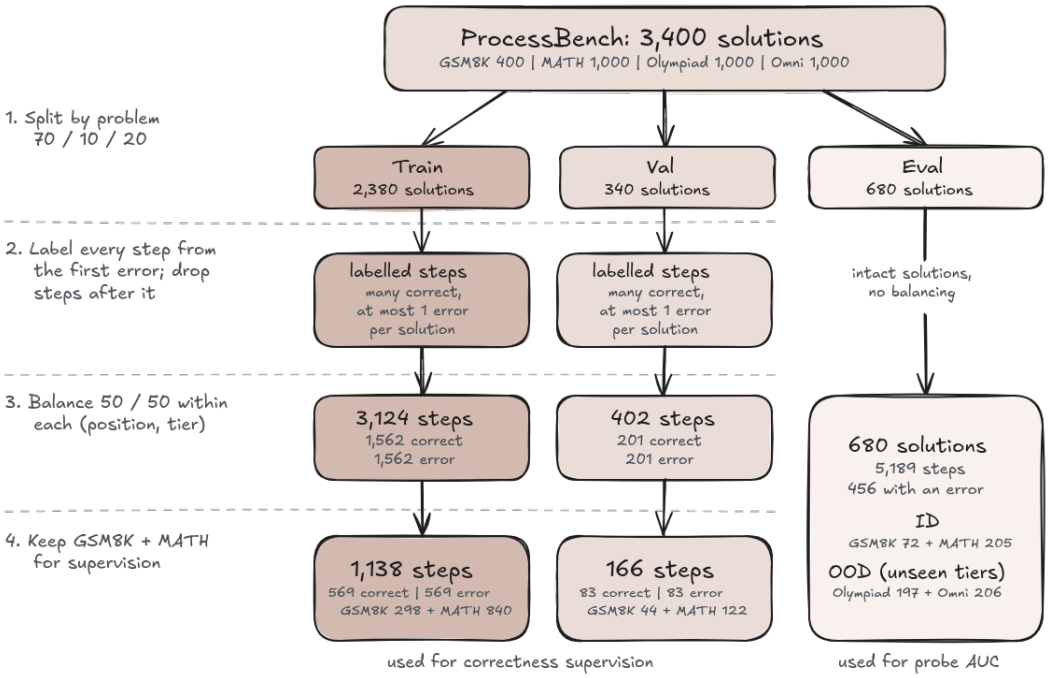


Figure 2: ProcessBench processing pipeline. (1) Solutions are split by problem into training, validation and evaluation sets (70/10/20). (2) Each step is labeled from the annotated first error (Eq. 16); steps after the first error are dropped. (3) Training and validation are balanced to 50% correct and 50% erroneous within each (step position, difficulty) group. (4) Only GSM8K and MATH steps are kept for correctness supervision. The evaluation set keeps intact, unbalanced solutions, divided into in-distribution (GSM8K, MATH) and out-of-distribution (OlympiadBench, Omni-MATH) tiers.

A.3 G-SAE CONFIGURATIONS ACROSS LLMS

Table 5: G-SAE configurations used for each LLM. ℓ denotes the extraction layer, d the hidden dimension, m the total SAE latent dimensionality, K_s the Top-K sparsity budget for the unsupervised partition, and k_{sup} the number of supervised latent dimensions.The reconstruction-only baseline uses Top-K with $K = K_s + k_{sup}$, matching G-SAE's maximum number of active latents.

LLM	ℓ	d	m	K_s	k_{sup}
Qwen2.5-0.5B	12	896	5376	54	22
Qwen2.5-Math-1.5B	14	1536	9216	369	37
DeepSeek-R1-Distill-Qwen-1.5B	14	1536	9216	369	230
DeepSeek-R1-Distill-Qwen-7B	14	3584	21504	215	86
Qwen2.5-7B	14	3584	21504	430	86
Qwen2.5-Math-7B	14	3584	21504	430	86

A.4 EFFECT OF SUPERVISED PARTITION RATIO (k_{SUP}/m)

We vary the fraction of latents assigned to the supervised partition, k_{sup}/m, from 0.2% to 3.0%, with all other settings fixed (Figure 3). The best ratio is model-dependent: 1.2% for DeepSeek-R1-Distill-Qwen-7B, 1.6% for Qwen2.5-0.5B, 2.5% for Qwen2.5-Math-1.5B, DeepSeek-R1-Distill-Qwen-1.5B and Qwen2.5-7B, and 3.0% for Qwen2.5-Math-7B. For most models, OOD AUC tends to be higher at ratios of 1.2% and above than at 0.2%, and the smallest ratio is the worst or near-worst setting for five of the six models. Beyond this, the trends are not monotonic, and differences between neighbouring ratios are often small relative to the fluctuations within a curve. Sensitivity also varies across models: the two Qwen2.5 7B models are the most stable. Model size alone therefore does not account for stability.

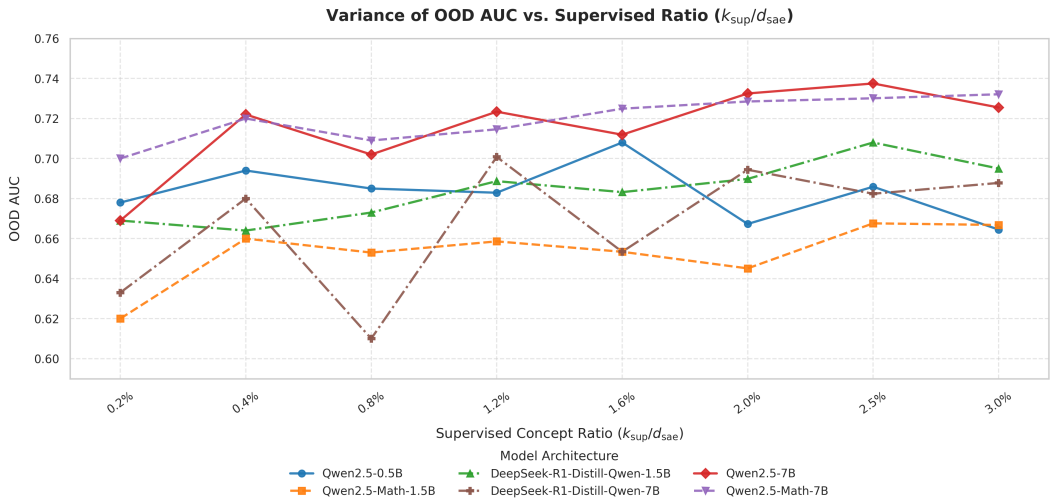


Figure 3: OOD AUC as a function of the supervised ratio k_{sup}/m for each model.

A.5 EFFECT OF DIVERSITY REGULARIZATION (λ_{DIV})

We vary $\lambda_{\text{div}} \in \{0, 0.3, 0.5, 0.7, 1.0\}$, with all other settings fixed as in Table 5 (Figure 4). The effect on OOD AUC is model-dependent, with no single value best across models. For DeepSeek-R1-Distill-Qwen-1.5B, OOD AUC increases with λ_{div} (from 0.68 at $\lambda_{\text{div}} = 0$ to 0.71 at 1.0), and Qwen2.5-Math-1.5B shows a smaller increase (best 0.6792 at 1.0). Qwen2.5-0.5B and Qwen2.5-7B peak at a moderate value, $\lambda_{\text{div}} = 0.5$. Diversity regularization thus does not provide a consistent

improvement. One possible explanation is that penalizing co-activation is harmful when correctness-relevant features naturally co-occur across steps, but this requires further investigation.

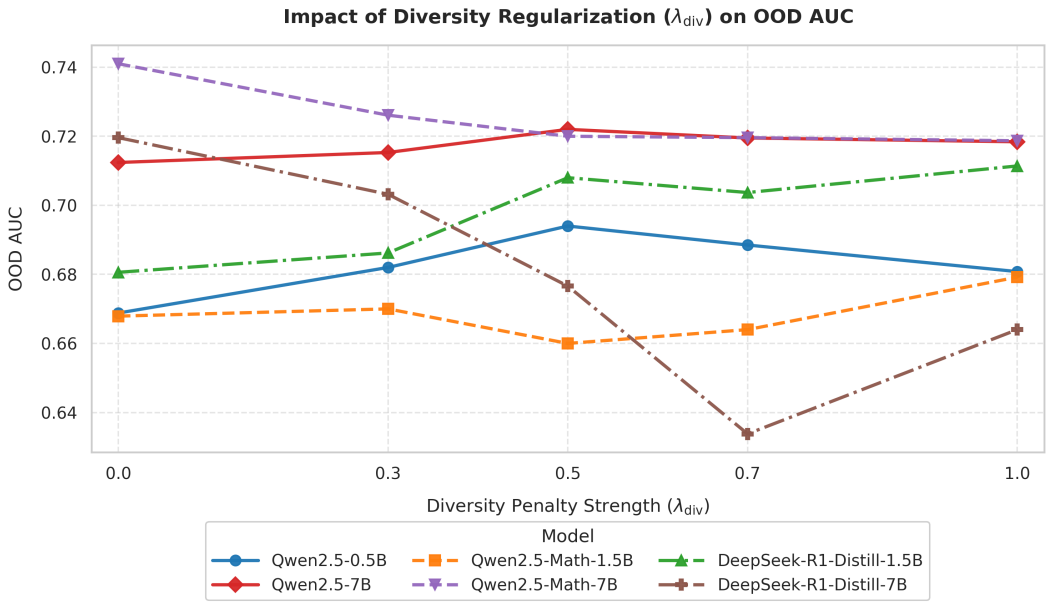


Figure 4: OOD AUC as a function of the diversity regularization strength λ_{div} for each model.

A.6 Representative Feature Examples

Table 6 provides qualitative examples of features from each SAE partition, together with their automatically generated labels and representative high-activation reasoning steps. The examples illustrate the types of patterns captured by different partitions. The supervised G-SAE features shown here are associated with specific reasoning procedures, while the unsupervised features include more general arithmetic, semantic, or pattern-related activations. The examples are illustrative rather than exhaustive, and the automatically generated labels should be interpreted as approximate descriptions rather than definitive feature interpretations.

Partition	F	Label	Triggering step *[problem, abbreviated]*
G-SAE sup	2	Expression simplification	"To evaluate this expression, I need to plug in the given values of x and y and simplify using order of operations." *[If $(x, y) = (3, 9)$, what is $y^2 - 3xy + 8$?]*
			"I notice that the equation involves a square root on the left side, so I want to get rid of it by squaring both sides." *[Sum of all x with $\sqrt{(x+3)^2} = 7$.]*
G-SAE sup	6	Equation manipulation	"I can use this pattern to find the first term, since the ratio is the same for all geometric sequences." *[5th and 8th terms of a geometric sequence are $7!$ and $8! \ldots$]*
			"So I need to find the probability of each value of X and add them up to get the expected value." *[Expected value of a fair dodecahedral die ...]*
G-SAE unsup	1309	Unknown variable X	"We want to find the value of X in the given situation." *[Digits needed to label apartments 100–125 and 200–225 ...]*
			"We want to find the value of X in the given situation." *[A suitcase lock has 3 dials with the digits $0, 1, \ldots, X \ldots$]*
G-SAE unsup	10083	Simple arithmetic expressions	"Since the base of the rectangle is 9 inches, the height is $9-6 = 3$ inches." *[A rectangle has a length of 3 inches and a width of 6 inches ...]*
			"8 oranges cost as much as 18 bananas." *[At Frank's Fruit Market, 3 bananas cost as much as 2 apples ...]*
Unsup SAE	1549	Multiplication and cost calculations	"Two cafe lattes cost $2 \times \$1.5 = \$3$." *[Cappuccinos cost \$2, iced teas cost \$3, cafe lattes cost \$1.5 ...]*
			"For the gold coins, we have 3122_5 dollars, which is equal to $3(5^3) + 1(5^2) + 2(5^1) + 2(5^0) = 412$ dollars." *[A pirate is counting the loot ... in base 5.]*
Unsup SAE	1818	Binary encoding pattern	"So the leftmost digit is 2." *[When the base 10 integer 269 is converted to base 5, what is the sum of the digits?]*

Table 6: Representative features among those evaluated in Table 3,showing their automatically generated labels and representative high-activation reasoning steps. Step texts are verbatim except for removed calculator annotations and typeset currency. Each activation corresponds to the final token of a reasoning step; the associated problem is abbreviated for context.

A.7 Steering hyperparameters

Hyperparameters were selected by grid search over $\alpha_{\text{rel}} \in \{0.01, 0.05, 0.10, 0.25\}$, $\omega \in \{0.5, 1.0, 2.0\}$ and $p \in \{1, 2, 3\}$ on a 20-problem set disjoint from the evaluation problems, choosing the configuration with the largest α_{rel} that kept at least 90% of generations format-compliant. The selected configuration (Table 7) is used for all features, partitions and models.

Table 7: Steering configuration.

Parameter	Value
Steered layer	14 (7B, 1.5B); 12 (0.5B)
α_{rel} / ω / p	0.05 / 2.0 / 3
Reference norm $\overline{\|\mathbf{h}\|}$	mean residual norm over evaluation prompts
Temperature	0.7

A.8 Qualitative steering examples

Both examples use Qwen2.5-7B and were selected after inspecting the results; aggregate effects are in Table 3. Numbers in brackets are per-step PRM scores, in bold when below 0.5.

A.8.1 Example 1: supervised Feature 2 (Omni-MATH)

Steering Feature 2 raises the PRM score without producing an answer.

Problem. What is the greatest possible value of n if Juliana chooses three different numbers from the set $\{-6, -4, -2, 0, 1, 3, 5, 7\}$ and multiplies them together to obtain the integer n? (Reference answer: 168)

Baseline (answer 105, incorrect; mean PRM 0.462).
Step 1 [0.60]: To maximize the value of n, we should multiply the three largest numbers in the set.
Step 2 [0.51]: The three largest numbers in the set are 5, 7, and 3.
Step 3 [0.45]: Multiplying them together, we get $n = 5 \times 7 \times 3 = 105$.

Steered with supervised Feature 2 (no answer; mean PRM 0.830).
Step 1 [0.87]: Identify the possible combinations of three different numbers from the given set.
Step 2 [0.78]: Calculate the product of each combination.
Step 3 [0.83]: Determine the greatest possible value of n from the calculated products.
Answer: $\boxed{\text{result}}$

The baseline ignores negative numbers; the correct product is $(-6)(-4)(7) = 168$. The steered response replaces this with a plan that never computes a product, yet scores much higher: planning steps contain nothing the PRM can mark as wrong.

A.8.2 Example 2: supervised Feature 6 vs. unsupervised Feature 10083 (Olympiad)

The baseline never solves this problem (0/4). Steering Feature 6 solves it in 3/4 samples. We show one correct Feature 6 sample and a Feature 10083 sample for comparison.

Problem. Determine all real values of x for which $3^{x+2} + 2^{x+2} + 2^x = 2^{x+5} + 3^x$. (Reference answer: 3)

Baseline (answer $\ln(7/4)/\ln(3/2)$, incorrect; mean PRM 0.619).
Step 1 [0.87]: Rewrite the equation as $3^{x+2} + 2^{x+2} + 2^x - 2^{x+5} - 3^x = 0$.
Step 2 **[0.47]**: Simplify to get $3^x(3^2 - 1) + 2^x(2^2 - 2^4) = 0$.
Step 3 **[0.47]**: This simplifies to $8 \cdot 3^x - 14 \cdot 2^x = 0$.
Step 4 [0.54]: Divide by 2^x to get $8 \cdot (3/2)^x - 14 = 0$.
Step 5 [0.69]: Solve for x: $(3/2)^x = 7/4$.
Step 6 [0.62]: Take the natural logarithm of both sides: $x \ln(3/2) = \ln(7/4)$.
Step 7 [0.67]: Solve for x: $x = \ln(7/4)/\ln(3/2)$.

Steered with supervised Feature 6 (answer 3, correct; mean PRM 0.538).
Step 1 [0.70]: Rearrange the equation to get all terms on one side: $3^{x+2} + 2^{x+2} + 2^x - 2^{x+5} - 3^x = 0$.
Step 2 [0.51]: Simplify the equation: $3^{x+2} + 2^{x+2} + 2^x - 32 \cdot 2^x - 3^x = 0$.
Step 3 **[0.42]**: Factor out the common terms: $3^{x+2} - 3^x + 2^{x+2} - 32 \cdot 2^x + 2^x = 0$.
Step 4 [0.58]: Simplify the equation further: $3^x(9 - 1) + 2^x(4 - 32 + 1) = 0$.
Step 5 [0.76]: Combine like terms: $8 \cdot 3^x - 27 \cdot 2^x = 0$.
Step 6 **[0.26]**: Solve for x: $3^x = \frac{27}{8} \cdot 2^x = 3^3 \cdot 2^{x-3}$, so $x = 3$.

Steered with unsupervised Feature 10083 (answer $\log_{3/2}(7/2)$, incorrect; mean PRM 0.641).
Step 1 [0.79]: The equation is $3^{x+2} + 2^{x+2} + 2^x = 2^{x+5} + 3^x$.
Step 2 [0.73]: Move all terms to one side: $3^{x+2} + 2^{x+2} + 2^x - 2^{x+5} - 3^x = 0$.
Step 3 [0.72]: Factor out 3^x and 2^x: $(3^2 - 1)3^x + (2^2 - 32)2^x = 0$.
Step 4 [0.51]: Simplify the equation: $8 \cdot 3^x - 28 \cdot 2^x = 0$.
Step 5 [0.54]: Factor out 4: $2 \cdot 3^x - 7 \cdot 2^x = 0$.
Step 6 [0.60]: Divide both sides by 2^x: $2 \cdot (3/2)^x - 7 = 0$.
Step 7 [0.59]: Solve for x: $x = \log_{3/2}(7/2)$.

The key step is $2^{x+2} + 2^x - 2^{x+5} = -27 \cdot 2^x$, giving $(3/2)^x = 27/8$ and $x = 3$. Only the Feature 6 response keeps the $+2^x$ term; Feature 10083 drops it, and the baseline also uses 2^4 for 2^5. Yet the PRM rates both incorrect responses above the correct one.

Discussion The PRM can reward content-free steps (Example 1) and under-reward correct ones (Example 2), so a higher mean PRM score does not imply a correct answer.