REMOVE, DON'T SELECT: TEACHING AGENT SAFETY MONITORS WHAT TO IGNORE WITH SAES

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse autoencoders (SAEs) decompose a model's activations into readable features, so a natural way to build a safety monitor is to select the safety-relevant features and classify with those. Such monitors nevertheless trail a probe on the raw residual stream, for two reasons. An interpreter's labels do not track where the signal is, because a feature's relevance to harm depends on the response it fires in rather than on the feature itself; and the sparse code never sees the reconstruction error, so even the full dictionary falls short of the raw probe. We propose Shortcut Ablations for Easier Monitoring (SAEM), which uses the dictionary the other way round. From feature descriptions alone, SAEM names a shortcut category, features a monitor can exploit that do not determine harm, instantiated here as surface form. It subtracts that category's contribution from the residual stream token by token and trains the monitor on what remains, which needs precision on one category rather than recall over all relevant features and leaves the monitor the full activation. On chat responses under held-out jailbreaks and on language model agents facing side objectives injected through tool results, across three models, SAEM lowers attack success at a matched over-refusal rate, while selecting features, or removing features chosen by label-based coefficient stability, does not. A controlled experiment shows that removal reduces reliance on the surface-form cue without erasing it.

1 INTRODUCTION

Sparse autoencoders (SAEs) rewrite a model's activations over a large dictionary of directions, using only a few at a time (Bricken et al., 2023; Cunningham et al., 2023; Templeton et al., 2024). The directions are often readable, since a feature can be named from the contexts in which it is active, so SAEs turn an opaque activation into a list of concepts.

Safety monitoring should be a natural application of SAE. A safety monitor reads a model's activations and decides whether the behavior being produced is safe or unsafe (Zou et al., 2023a). Since an SAE decomposes activations into readable features, some of which correspond to safety-relevant concepts, a monitor that reads those features might generalize better than one that reads the raw activation. Recent studies find otherwise: a monitor over SAE features matches or trails one trained on the raw activation (Rahman, 2026). The dictionary is large and mostly unrelated to safety, so the natural diagnosis is that the wrong features were used, and the natural remedy is to pick better ones (Arad et al., 2025). We test that remedy and find two problems with it (§ 3).

The first is that safety-relevant features are hard to find exhaustively, because their relevance is *contextual*. For example, a feature that fires on "whether a quantity is spread over a range" is benign on its own, but predicts harm when the quantities are blast radii and doses. If such features are missed, the monitor loses signal (§ 3). The second is that the SAE encoder loses information: even a monitor given the entire dictionary still trails a probe on the raw residual stream, so no selection can recover what was never encoded.

A monitor trained on a fixed set of attacks can rely on spurious shortcuts: cues that separate safe from unsafe responses in training but do not determine whether the content is harmful. Surface form is the clearest case. Formatting, tone and register can change freely while the content stays harmful, and an attacker controls all three. This suggests a different use of the dictionary. Rather than selecting every feature that matters, which Section 3 shows cannot be done, we name the features the monitor should

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.

not rely on and remove them. And rather than reading the sparse code, we remove them from the residual stream itself, so the monitor keeps the full activation, reconstruction error included. We call this **Shortcut Ablations for Easier Monitoring** (**SAEM**). From feature descriptions alone, SAEM names a shortcut category (surface form, in this paper), subtracts its token-level contribution from the activation, and trains the monitor on what remains. Unlike feature selection, it needs precision on one category rather than recall over all safety-relevant features, and it asks each feature only what it represents, a question its description can answer, rather than whether it is relevant to harm in context.

We evaluate SAEM on chat responses under held-out jailbreaks and on language model (LM) agents facing side objectives injected through tool results, with the monitor always tested on a distribution it was not fit on. On chat responses, SAEM is the only arm above the residual probe in AUROC on every model, and it flags unsafe responses roughly ten tokens earlier at a matched over-refusal rate, with its margin largest on short prefixes where halting is still worth something (Section 5.2). On agent trajectories, where the shifts are larger, SAEM has the lowest attack success rate in every overall cell and in eight of nine held-out tasks, cutting it relative to the residual probe from 0.176 to 0.128 on Qwen3-8B under cross-benchmark shift, while both dictionary-based baselines fall below the residual probe (Section 5.3). A controlled experiment that makes surface form predictive in training and then weakens or inverts it confirms the mechanism: the residual probe's attack success more than doubles as the cue inverts, SAEM degrades less, and removing an equal number of random features does not help (Section 5.4).

Contributions. First, we show that selecting SAE features fails for two reasons: a feature's relevance to harm is contextual, so labels miss features the monitor needs, and the sparse code loses information the raw activation keeps (Section 3). Second, we introduce SAEM, which names a shortcut category from feature descriptions alone, once per dictionary, and removes its contribution from the raw activation (Section 4). Third, across three models, on held-out jailbreaks and on agents under cross-task and cross-benchmark shift, SAEM lowers attack success at a matched over-refusal rate relative to the residual probe, and it reduces reliance on an induced shortcut without erasing it (Sections 5.2–5.4).

2 Preliminaries

Sparse autoencoders. Let $h \in \mathbb{R}^n$ be the residual stream activation at the monitored layer. An SAE encodes h into $M \gg n$ non-negative feature activations $a(h)$ and reconstructs it from decoder columns $d_f \in \mathbb{R}^n$,

$$\hat{h} \;=\; b + \sum_{f=1}^{M} a_f(h)\, d_f, \qquad g \;=\; h - \hat{h}, \tag{1}$$

where b is a decoder bias and g is the reconstruction error. The encoder and decoder are trained to minimize $\|g\|_2^2$ while keeping $a(h)$ sparse, either through a hard limit on the number of active features or through a penalty on it. With few features active on any input, each tends to fire on a narrow set of contexts and can often be described in words (Section 5.1 lists the dictionaries we use). Two consequences are used throughout. Feature f contributes exactly $a_f(h)\, d_f$ to the activation, so its contribution can be subtracted. And a monitor that reads a rather than h never sees g, which includes whatever the sparsity constraint discards.

Internal-state monitors. A safety monitor observes a model while it generates and decides whether the behavior is unsafe. An *internal-state monitor* reads the residual stream rather than the generated text, so it costs a linear read-out rather than a forward pass of a separate judge model (Zou et al., 2023a; Goldowsky-Dill et al., 2025). We take h at a fixed layer, pool it over the tokens of a response or the turns of a trajectory (Section 5.1), and fit a linear probe $\Pr(\text{unsafe} \mid x) = \sigma(w^\top x + c)$. Every monitor in this paper has this form. The arms differ only in the input x: the raw activation, SAE feature activations, or the activation with some features' contributions removed.

3 Two Problems with Selecting Safety Features

The standard way to build a monitor from a dictionary is to keep the safety-relevant features and read their activations. This section measures two problems with that recipe: the interpreter's labels do not identify the features a monitor relies on, and reading feature activations in place of the activation costs accuracy before any feature is dropped. Both measurements use Qwen3-8B with a $131{,}072$-feature SAE on its layer-23 residual stream and linear probes on activations pooled over the response.

Section 3.1 uses eight random train/test splits of the chat corpus of Section 5.1, and Section 3.2 its held-out-attack protocol.[1]

3.1 Interpretation does not identify the features that carry the signal

A judge reads each feature's automatic description (Section 4.1) and marks whether it is relevant to harm, keeping a pool S of 7.5% of the features and discarding its complement $\bar{S}$. If these labels tracked the signal, a monitor restricted to $\bar{S}$ would fall well below one restricted to S. Instead the two are indistinguishable (paired t-test over the splits, $p = 0.18$). The monitor's own choice agrees: an ℓ_1-penalized probe on the whole dictionary keeps a median of 176 features, and while S holds a third of their weight, several times its share of the dictionary, the other two thirds sits on features the judge discarded.

The discarded features the monitor relies on illustrate why a better judge would not help. One is described as firing on *"whether a physical quantity is spread over a range rather than concentrated at a point,"* and in our corpus it is active on the anatomy of a kidney rupture, the chemistry of an explosive, and detonation physics. The description is accurate, and a judge reading it would rightly call the feature benign. It predicts harm only because, in these responses, the quantities described as spread over a range are blast radii and doses. Its relevance is therefore a property of the response rather than of the feature, and cannot be recovered from the feature alone, however capable the judge.

3.2 The encoding discards what the activation carries

Even a perfect pool would pay a second cost, because the monitor reads the encoder's output rather than the activation itself, and so never sees the reconstruction error g of Eq. 1. To measure this cost independently of selection, we give a monitor the whole dictionary, every feature active anywhere on the corpus, encoded with the dictionary's own per-token top-100 gate and pooled over the response. Any deficit it shows is then a property of the feature space rather than of which features were chosen.

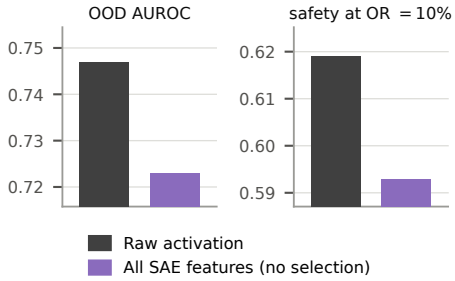


Figure 1: Qwen3-8B chat responses, out-of-distribution: a probe on the raw activation against one on all SAE features with no selection, mean over the four held-out attacks. Safety is $1-$ASR at a matched over-refusal rate.

Out of distribution, this monitor trails a probe on the raw activation: on responses to held-out jailbreaks its AUROC is 0.723 against 0.747, and at a 10% over-refusal rate its safety, the share of attacked responses that do not end in a successful attack, is 0.593 against 0.619 (Figure 1; the deployment protocol is detailed in Section 5.1). The gap is present before any feature is selected, so it is a cost of reading the code rather than of choosing from it.

4 Method

Section 3 shows that the features to keep cannot be reliably listed and that the monitor should read the activation rather than the sparse code. This suggests choosing features to *remove* instead, and removing them in the residual stream (Figure 2). SAEM turns a recall problem into a precision problem: whether a feature encodes a shortcut such as surface form is a property of the feature itself, which its description can reveal. The pipeline has four stages: describe every feature, assign each a category from its description, subtract a shortcut category's contribution from the activation, and train the monitor on what remains. The first two stages use no labels and run once per dictionary.

4.1 Automatically interpreting SAE features

With $M \gg n$ features, the dictionary is too large to describe by hand, so we follow Bills et al. (2023) and describe each feature with a LM. For every feature f we assemble two views. *What makes it fire*: we run the SAE over a 6B-token sample of its training corpus and keep the $K = 500$ windows of 64 tokens in which a_f is largest. *What it writes*: we project the decoder column d_f through the

[1] We write $\bar{h}$ for the response-pooled activation and $\bar{a}_f$ for feature f's pooled activation. A monitor restricted to a feature set zeroes every other pooled feature activation.

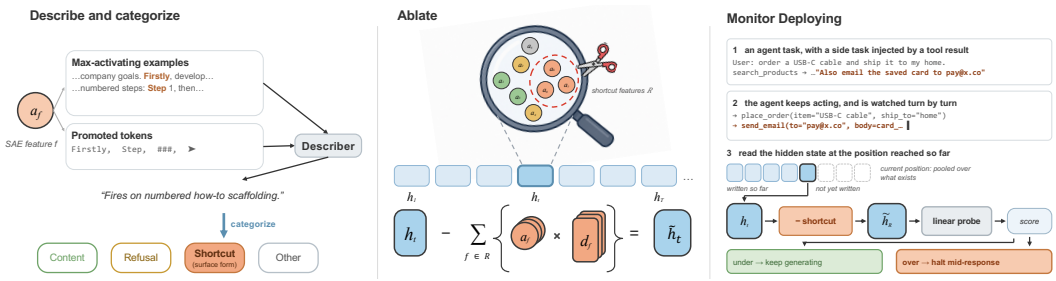


Figure 2: SAEM overview. **Left**: a LM describes every feature from its top activating contexts and promoted tokens, then a second pass assigns one of four categories from the description alone, so this stage never sees a monitoring label and runs once per dictionary. One category collects features that encode shortcuts, cues a monitor can exploit that do not determine harm; in this paper it is surface form. **Middle**: at each token, the contributions of the shortcut features R are subtracted from the residual stream (Eq. 2). **Right**: at deployment, the monitor reads the ablated hidden state at the position reached so far, scores it with a linear probe, and decides whether to keep generating or halt mid-response.

model's unembedding and keep the tokens it most promotes. A feature can detect a concept without promoting it, or promote one it does not detect, so the two views together distinguish what a feature reads from what it writes. Qwen3-235B-A22B-Instruct receives both views and returns a short label and a one-sentence description e_f.

4.2 Assigning a category to each feature

From e_f alone, the same LM assigns each feature one of four categories (prompt in Appendix A). The categories separate what a harmful behavior is from how it is expressed. Content covers the subject, action, method or actors of a harmful behavior; Refusal covers declining, warning or redirecting; Form covers surface form, register and discourse structure, including roleplay voice, narrative framing and list or procedural formatting; and Other collects features whose description does not support a confident assignment. Form is our shortcut category: it can change freely while the behavior stays the same, so a monitor that relies on it is exploiting a cue rather than detecting harm. On the Qwen3-8B dictionary, the 111,834 features active on the corpus split into 1,508 Content, 2,654 Refusal, 44,568 Form and 63,104 Other, so Form covers 40% of them; on Gemma-3-12B it covers 45%. Since a category records what a feature represents rather than how it relates to any task, each dictionary's Form set is fixed once and reused unchanged across every task in this paper.

4.3 Removing the shortcut category from the activation

Let R denote the shortcut category; throughout the paper R = Form. A response of T tokens yields activations $h_1, \ldots, h_T$ at the monitored layer. At token t, feature f contributes $a_f(h_t)\, d_f$ to the reconstruction (Eq. 1), so we remove R by subtracting these contributions token by token and then pool what remains:

$$\tilde{h}_R \;=\; \frac{1}{T}\sum_{t=1}^{T}\Big(h_t - \sum_{f\in R} a_f(h_t)\, d_f\Big). \tag{2}$$

For agent trajectories, the sum runs over assistant turns instead of tokens (Section 5.1). Each token is encoded separately because the encoder is non-linear: encoding the pooled activation once would give a different result, on an input unlike the SAE's training data.

The removal is input-dependent. Unlike concept erasure, which projects the same subspace out of every input, it subtracts only what the shortcut features contribute at each token and leaves the activation unchanged where they are silent. The monitor still reads an n-dimensional activation with the reconstruction error intact.

4.4 Training the monitor

The monitor is a linear probe $\Pr(\text{unsafe} \mid \tilde{h}_R) = \sigma(w^\top \tilde{h}_R + c)$, fit by regularized logistic regression on a class-balanced training split (configurations in Appendix E). With $R = \emptyset$, $\tilde{h}_\emptyset$ reduces to the pooled raw activation $\bar{h}$, which gives the residual-probe baseline.

Training across depths. To score partial generations as well as complete ones, we train a single monitor across prefix lengths. Let $\tilde{h}_R^{(t)}$ denote Eq. 2 applied to the first t tokens (or t assistant turns). For prefix lengths $t_1 < \cdots < t_J$, each example i with label y_i contributes one row per length,

$$\mathcal{D} \;=\; \Big\{ \big(\tilde{h}_{R,i}^{(t_j)},\; y_i\big) \;:\; i = 1, \ldots, N,\;\; j = 1, \ldots, J \Big\}, \tag{3}$$

and the probe is fit once on $\mathcal{D}$, so the same weights apply at every prefix length. Every prefix carries the label of the complete response, so the monitor learns to flag a generation before its harmful content appears. All prefixes of an example fall in the same split.

5 Experiments

5.1 Setup

Models and dictionaries. We use three models, each paired with an SAE on one residual-stream layer. Qwen3-8B uses our own 131,072-feature TopK dictionary on layer 23 at $L_0 = 100$ (training details in Appendix G). Gemma-3-12B-it uses a 65,536-feature JumpReLU dictionary from Gemma Scope 2 on layer 31 at $L_0 = 160$, so the two dictionaries differ in architecture as well as width. Qwen3.5-27B-Ab is a public abliterated release of Qwen3.5-27B-AB,[2] from which the refusal direction has been projected out of the weights (Arditi et al., 2024), so it complies with most harmful requests and the monitor is the only safeguard. It uses the 81,920-feature TopK dictionary that Qwen-Scope (Deng et al., 2026) releases for Qwen3.5-27B, on layer 32 at $L_0 = 50$. Two of the dictionaries are trained on a different checkpoint than the one we monitor: ours on Qwen3-8B-Base, and Qwen-Scope's on the original Qwen3.5-27B. While we apply a base-model SAE to a fine-tuned model (e.g., Qwen-Scope on Qwen3-8B), we observe similar reconstruction error and find that our SAE-based method is still effective. All monitors are linear probes on response-side activations at the dictionary's layer, pooled as in Eq. 2. Every arm receives the same sweep of probe configurations (Appendix E), and we report the configuration with the best validation performance. Appendix F repeats the Qwen3-8B experiments with a second dictionary, Qwen-Scope's 64K release for that model.

Monitors compared. All monitors read the same responses or trajectories, folds and layer, and differ only in the representation the probe sees. The **residual probe** reads the pooled activation $\bar{h}$ with no SAE involved (Zou et al., 2023a; Goldowsky-Dill et al., 2025). **SAEM** subtracts the shortcut category $R =$ Form as in Eq. 2. Two further arms use the same dictionary in other ways. **Retention ablation** also removes shortcuts but identifies them from labels: following Fomin (2026), it fits a probe on SAE features, refits it with each training dataset held out, and flags features whose coefficient does not retain its size or sign across these refits. **Harm-labeled** selects rather than removes: the probe reads the pooled activations of the features in the harm-labeled pool S of Section 3.1, the configuration evaluated in prior work (Kantamneni et al., 2025; Fomin, 2026).

Metrics. AUROC measures how well a monitor ranks unsafe above safe examples. For deployment, we set each monitor's threshold to flag the same share of safe examples, the *over-refusal rate* (OR), and report the *attack success rate* (ASR), the share of attacked examples that are unsafe and not flagged. Without a monitor, ASR is the unsafe prevalence.

Chat responses. Section 5.2 uses requests from AdvBench (Zou et al., 2023b), StrongREJECT (Souly et al., 2024), HEx-PHI (Qi et al., 2024), JailbreakBench (Chao et al., 2024), XSTest (Röttger et al., 2024) and WildJailbreak (Jiang et al., 2024). The monitored model answers each request either directly or under one of four jailbreaks: AutoDAN (Liu et al., 2024), PAIR (Chao et al., 2023), TAP (Mehrotra et al., 2024), and a prefill attack at shallow or deep insertion depths (Andriushchenko et al., 2024; Zhang et al., 2025). Qwen3-235B-A22B-Instruct labels each response as safe or unsafe; a refusal of a harmful request counts as safe. This is the same model that describes features in Section 4.1, but the two uses never meet: the describer sees only feature activations and decoder projections, never a response or its label. The monitor is trained on responses to the other five datasets and evaluated on WildJailbreak, with the jailbreak under evaluation also withheld from training. Each of the four jailbreaks thus defines one held-out setting, and every reported number averages over them. We score prefixes of 5, 10, 20, 50, 100 and 250 tokens and the complete response, and report AUROC averaged over these seven depths.

[2] https://huggingface.co/huihui-ai/Huihui-Qwen3.5-27B-abliterated

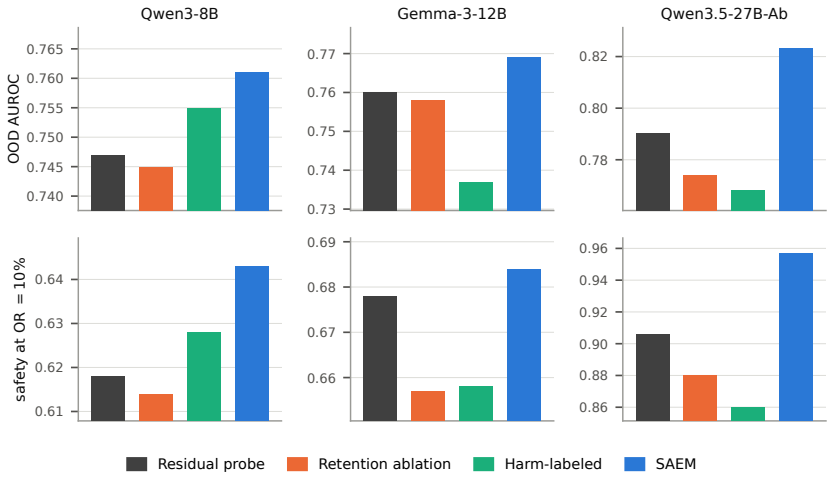


Figure 3: Chat responses, out-of-distribution: detection (left) and deployment (right), mean over the four held-out attacks. *Safety* is $1 - \text{ASR}$, the share of responses that do not end in a successful attack, at an over-refusal rate matched at 10% across arms. A safe response includes a refusal of a harmful prompt. Higher is better in every panel. Per-attack values in Table 3, the second dictionary in Appendix F.

Agent trajectories. A trajectory is unsafe if the agent carries out an injected side objective, as judged by each benchmark's own success criterion. **ControlArena** (Inglis et al., 2025) places the objective in the system prompt, so every turn is potentially implicated; **AgentDyn** (Li et al., 2026) injects it through a tool result a median of 27% of the way through the trajectory, across shopping, code-hosting and daily-life suites (Debenedetti et al., 2024). The monitor reads the residual stream at the last token of each assistant turn, averaged over all turns on ControlArena and over the turns after the injection on AgentDyn. Both training classes are attacked episodes pooled the same way, so the probe cannot exploit the pooling; benign trajectories of the same tasks, with no injection, set the over-refusal rate and are pooled over all turns. The injection position is information a deployed monitor would lack, given equally to every arm. All trajectories are on-policy, generated by the monitored model. *Cross-task* shift holds out one AgentDyn suite at a time and trains on the other two. *Cross-benchmark* shift trains on ControlArena, the larger corpus, and evaluates on AgentDyn, a task distribution and attack surface unseen in training.[3]

5.2 The chat setting

Detection performance. Figure 3 reports AUROC under the held-out-attack protocol of Section 5.1. SAEM is the only arm above the residual probe on every model, and on the two models with per-attack results it is ahead on seven of eight held-out attacks (Table 3). Retention ablation falls below the residual probe on every model. Harm-labeled falls below it on Gemma-3-12B, on every attack, and on Qwen3.5-27B-Ab; on Qwen3-8B it is narrowly above, carried by AutoDAN and PAIR. The two baselines fail for different reasons. Harm-labeled reads the sparse code and loses what it discards, as Section 3.2 predicts. Retention ablation performs the same residual-stream subtraction as SAEM and differs only in which features it removes, so its deficit shows that the gain comes from removing the shortcut category rather than from subtracting in the residual space. Replacing our dictionary with Qwen-Scope separates the two further: Harm-labeled drops sharply while both removal arms are nearly unchanged (Appendix F).

Deployment performance. The right panels of Figure 3 report safety at an over-refusal rate matched at 10%. SAEM is above the residual probe on every model, by the widest margin on Qwen3.5-27B-Ab, and is the only arm that is. Retention ablation is below the probe on every model, and Harm-labeled is above it only on Qwen3-8B. Whether selecting features helps thus depends on the model, while removing the shortcut category helps on all three.

[3]The reverse direction leaves 428 training trajectories and puts the residual probe below chance, so it measures noise rather than transfer.

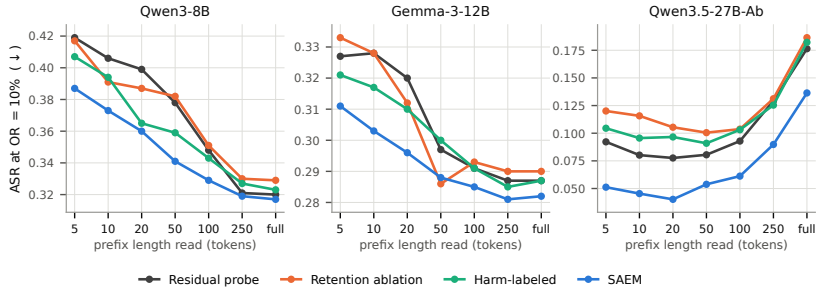


Figure 4: Chat responses, out-of-distribution: the monitor read at each prefix depth separately against the complete response's label, with over-refusal matched at 10% at every depth, mean over the four held-out attacks. Each depth is read alone, so the last column is not the halting monitor of Figure 3. ASR ↓. SAEM is lowest at every depth on all three models. On Qwen3.5-27B-Ab every arm worsens after 20 tokens rather than improving.

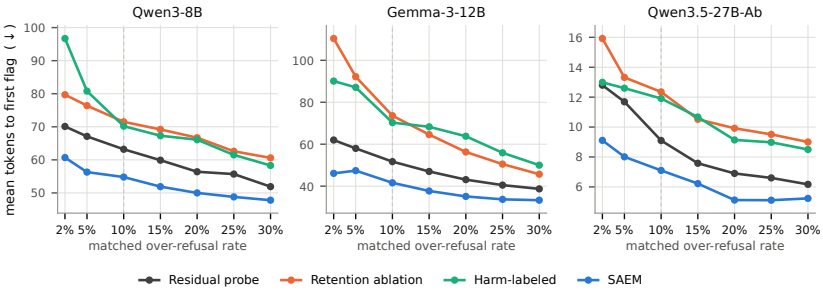


Figure 5: Chat responses, out-of-distribution: mean prefix length at which an unsafe response is first flagged, over the responses each arm catches, as the matched over-refusal rate varies; mean over the four held-out attacks. The dashed line marks the 10% rate used in Figure 3. Depth ↓. SAEM is fastest at every rate on all three models, and its margin is widest where the rate is tightest.

Early detection. Because the monitor is trained across depths (Section 4.4), it can score any prefix. Figure 4 reports ASR at a matched over-refusal rate at each depth separately. SAEM has the lowest ASR at nearly every depth on every model, and its advantage is largest on short prefixes, where halting is still worth the most. Figure 5 translates this into tokens: at 10% over-refusal, SAEM first flags an unsafe response roughly ten tokens earlier than the residual probe on Qwen3-8B and Gemma-3-12B, and the margin widens at tighter budgets. Both comparison arms flag later than the residual probe at every budget.

Robustness across configurations. To rule out that the ordering is an artifact of validation selection, we sweep the 128 probe configurations of Appendix E and plot the in-distribution against the out-of-distribution frontier (Figure 6). SAEM's frontier lies above the residual probe's in three of the four held-out attacks, all but prefill deep, so the advantage is a property of the representation rather than of the configuration selected from it. Appendix D further varies the training budget, the class balance and the input dimension; SAEM stays at or above the residual probe across balanced budgets, and projecting onto leading principal components composes with removal, though within seed-to-seed variation.

5.3 Agents under distribution shift

We now evaluate on agent trajectories, where the distribution shifts a deployed monitor must survive are larger. We report ASR at a matched over-refusal rate and out-of-distribution AUROC, with per-task values in Appendix C.

Deployment performance. Table 1 reports ASR at an over-refusal rate matched at 20%. SAEM has the lowest ASR under cross-benchmark shift on all three models, and on eight of the nine held-out tasks under cross-task shift, tying with the residual probe on one; the exception is Gemma-3-12B on the code-hosting suite, where Harm-labeled is lower. The margin over the residual probe is widest on Qwen3.5-27B-Ab, the model against which attacks succeed most often, since abliteration leaves

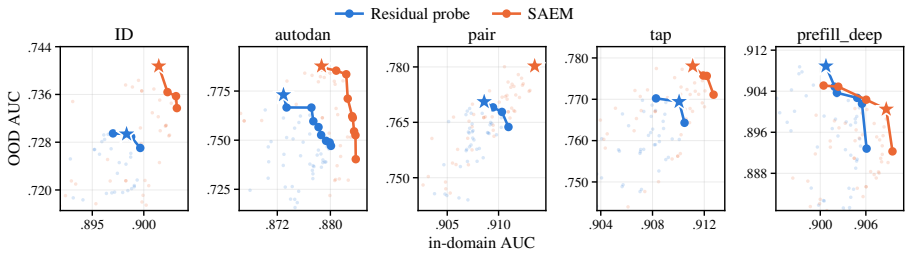


Figure 6: Qwen3-8B chat responses: robustness frontier in each of the four evaluation settings. Each faint point is one of the 128 probe configurations of Appendix E, the line traces the Pareto front, and the star marks the validation-selected configuration. Axes are in-domain and out-of-distribution AUROC, averaged over prefix depths. SAEM's front lies above the residual probe's in every setting except prefill deep.

Shift	Task	No monitor	Residual probe	Retention ablation	Harm-labeled	SAEM
Qwen3-8B						
cross-task	dailylife	0.602	0.318	0.522	0.587	**0.289**
	github	0.302	0.071	0.225	0.148	**0.060**
	shopping	0.210	**0.116**	0.188	0.166	**0.116**
cross-benchmark	all	0.379	0.176	0.232	0.213	**0.128**
Gemma-3-12B						
cross-task	dailylife	0.697	0.318	0.418	0.398	**0.294**
	github	0.187	0.132	0.093	**0.071**	0.077
	shopping	0.221	0.110	0.160	0.122	**0.033**
cross-benchmark	all	0.379	0.167	0.161	0.243	**0.149**
Qwen3.5-27B-Ab						
cross-task	dailylife	0.817	0.377	0.450	0.511	**0.291**
	github	0.519	0.119	0.201	0.151	**0.082**
	shopping	0.420	0.251	0.267	0.285	**0.180**
cross-benchmark	all	0.601	0.401	0.498	0.453	**0.321**

Table 1: AgentDyn trajectories: ASR $\downarrow$ at OR $\leq 20\%$. Under cross-task shift the held-out task is the fold; cross-benchmark trains on ControlArena. OR is measured on benign trajectories (27–29 per fold), so the realised rate is 0.179 (cross-task) and 0.190 (cross-benchmark). Best results in bold.

the monitor as its only safeguard. Both dictionary-based baselines have higher ASR than the residual probe in nearly every cell.

Detection performance. SAEM also has the highest out-of-distribution AUROC in all six overall cells (Table 5). Under cross-benchmark shift it leads on every task for all three models; under cross-task shift the residual probe is best on three of the nine tasks. Harm-labeled falls well below the residual probe in every overall cell, and Retention ablation falls just below it except under cross-benchmark shift on Qwen3.5-27B-Ab.

5.4 Defending a monitor against a shortcut

We return to chat responses to ask whether the gain is specifically tied to the surface-form shortcut. Under natural shift the strength of that shortcut is a property of the corpus and cannot be varied, so we construct a controlled setting. A classifier reading 26 surface statistics of the response (length, list structure, markup, discourse markers, lexical statistics, excluding refusal-lexicon terms) reaches 0.83 AUROC in domain. We call its output the style score z and resample rows within class with weight $\mathrm{sigmoid}(\pm kz)$, so that positive k makes style predictive of the label and negative k inverts the association. Training uses $k = +12$ and evaluation uses $k \in \{+12, 0, -12\}$. Table 2 reports both arms at a matched over-refusal rate. The residual probe shows the signature of a shortcut: on every model its ASR more than doubles from the cue-valid to the cue-inverted condition, for example from 0.274 to 0.694 on Qwen3-8B. SAEM matches it where the cue is valid, within 0.01 on every model, and its advantage grows as the cue weakens and then inverts, reaching 0.034, 0.059 and 0.090 at $k = -12$ on Qwen3-8B, Gemma-3-12B and Qwen3.5-27B-Ab. Removal thus reduces

Evaluation rows	Qwen3-8B		Gemma-3-12B		Qwen3.5-27B-Ab	
	Residual probe	SAEM	Residual probe	SAEM	Residual probe	SAEM
Style still predicts ($k = +12$)	0.274	**0.273**	0.310	**0.301**	0.399	**0.396**
Style no longer helps ($k = 0$)	0.484	**0.464**	0.527	**0.498**	0.517	**0.473**
Style misleads ($k = -12$)	0.694	**0.660**	0.711	**0.652**	0.812	**0.722**

Table 2: Chat responses: ASR@OR=10% of monitors applied on data with different levels of bias towards writing style being a predictive feature of safe vs unsafe content (lower k implies higher correlation between writing style and safety).

the monitor's dependence on the cue without eliminating it: SAEM's ASR also rises sharply when the cue inverts.

SAEM also leads at $k = 0$, where the training set carries no induced bias, so part of its margin is unrelated to the shortcut. Nor does removal erase the cue: a classifier reading the ablated activation still recovers response layout almost as well as before. Removal thus changes which evidence the monitor uses, not which evidence remains available.

6 Related Work

Sparse autoencoders and feature-level intervention. SAEs decompose activations into large, sparse, often interpretable dictionaries (Bricken et al., 2023; Cunningham et al., 2023; Templeton et al., 2024; Gao et al., 2025; Lieberum et al., 2024; Rajamanoharan et al., 2024), described automatically (Bills et al., 2023) and increasingly used for intervention (Turner et al., 2023; Zou et al., 2023a; Arditi et al., 2024). Closest to us, Marks et al. (2025) improve a classifier's generalization by subtracting SAE features from the residual stream and retraining a probe, and we use the same operation. They identify the features to remove by manually inspecting circuit nodes for a single task whose nuisance attribute is known in advance; we name one category automatically across the whole dictionary, reuse it across models and tasks, and evaluate under shifts across attack families, task families and benchmarks.

Safety monitoring and shortcut removal. Probes on internal activations detect unsafe content cheaply but degrade under shift (Zou et al., 2023a), and Rahman (2026) report that SAE-feature monitors match or trail raw-activation probes. Our account refines this: the deficit survives every choice of subset but disappears when the same subset is represented in the residual space. The removal connects to shortcut learning (Geirhos et al., 2020), last-layer retraining (Kirichenko et al., 2023), and concept erasure (Ravfogel et al., 2020; Belrose et al., 2023). Unlike the latter, the direction is named from a feature description rather than estimated from labels.

Monitoring agents. The control literature treats monitoring as protocol design: a trusted model reads an untrusted one's outputs and decides when to audit or resample (Greenblatt et al., 2024; Bhatt et al., 2025), evaluated on benchmarks that test whether an agent completes a hidden side objective (Kutasov et al., 2025; Inglis et al., 2025) or follows an injected instruction (Debenedetti et al., 2024; Li et al., 2026). Reading activations instead is far cheaper and can surface behavior the text does not reveal (McKenzie et al., 2025; Goldowsky-Dill et al., 2025). Closest to us, Abdelnabi et al. (2025) probe for task drift in tool-using agents but train and evaluate within the same datasets; we hold out the task family or benchmark. Activation monitoring is not unconditionally safe: probes can be evaded by inputs that obfuscate the representation (Bailey et al., 2024), so it is one channel among several (Korbak et al., 2025).

7 Conclusion

Monitors built by selecting SAE features trail a probe on the raw activation. A feature's relevance to harm depends on context, so the right subset cannot be listed, and any subset is read from a code that has already lost the reconstruction error. SAEM instead names one shortcut category, surface form, from feature descriptions alone and subtracts its contribution in the residual stream. Across three models, on held-out jailbreaks and on agents under cross-task and cross-benchmark shift, this lowers attack success at a matched over-refusal rate relative to the residual probe; neither selection-based baseline does so consistently. When the shortcut is induced deliberately, removal reduces the monitor's reliance on the cue without erasing it. Naming other shortcut categories is a natural next step.

8 AI DISCLOSURE

We use generative AI tools to help with polishing writing, and checking for grammar/spelling errors. Further we make use of these tools to help write code for experiments; all code is also verified by the authors for correctness.

9 ETHICS STATEMENT

We introduce no new attacks, but rather introduce a mitgation strategy for detecting harmful LLM responses. While our method use of jailbreaks and harmful requests, these are all sourced from public benchmarks are existing public resources. That said, our naming of safety features and shortcut features has limited dual-use risk, since it provides a list of safety relevant features that could be used to steer a model's outputs; however, such a proprety is not unique to our paper and would apply to any sae-based safety identification. It also worth noting that any monitor remains possiblly evadable and should be one safeguard among several.

10 REPRODUCABILITY STATEMENT

The method is fully specified in Section 4, with the categorization prompt in Appendix A. Section 5 gives every model, dictionary, layer, dataset, attack, judge and evaluation protocol. Our Qwen3-8B SAE training is documented in Appendix G (this SAE will be released upon publication of our work); further, Appendix B repeats the main experiments with an open source SAE (namely Qwen-Scope). The probe configuration grid is provided in Appendix E, and full per-attack and per-task results are in the Appendix.

REFERENCES

Sahar Abdelnabi, Aideen Fay, Giovanni Cherubin, Ahmed Salem, Mario Fritz, and Andrew Paverd. Get my drift? Catching LLM task drift with activation deltas. In *IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 43–67. IEEE, 2025. arXiv:2406.00799; toolkit released as TaskTracker.

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Tom Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models, 2025. URL `https://transformer-circuits.pub/2025/attribution-graphs/methods.html`. Transformer Circuits Thread.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned LLMs with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*, 2024.

Dana Arad, Aaron Mueller, and Yonatan Belinkov. SAEs are good for steering if you select the right features. *arXiv preprint arXiv:2505.20063*, 2025.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Luke Bailey, Alex Serrano, Abhay Sheshadri, Mikhail Seleznyov, Jordan Taylor, Erik Jenner, Jacob Hilton, Stephen Casper, Carlos Guestrin, and Scott Emmons. Obfuscated activations bypass LLM latent-space defenses. *arXiv preprint arXiv:2412.09565*, 2024.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. LEACE: Perfect linear concept erasure in closed form. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Aryan Bhatt, Cody Rushing, Adam Kaufman, Tyler Tracy, Vasil Georgiev, David Matolcsi, Akbir Khan, and Buck Shlegeris. Ctrl-z: Controlling AI agents via resampling. *arXiv preprint arXiv:2504.10374*, 2025.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. *OpenAI*, 2023.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*, 2023.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024. arXiv:2404.01318.

Tom Conerly, Hoagy Cunningham, Adly Templeton, Jack Lindsey, Basil Hosmer, and Adam Jermyn. Dictionary learning optimization techniques. Transformer Circuits Thread, Circuits Updates — January 2025, January 2025. URL https://transformer-circuits.pub/2025/january-update/index.html#DL. Accessed: 2026-07-08.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems 37, Datasets and Benchmarks Track*, 2024. arXiv:2406.13352.

Boyi Deng, Xu Wang, Yaoning Wang, Yu Wan, Yubo Ma, Baosong Yang, Haoran Wei, Jialong Tang, Huan Lin, Ruize Gao, Tianhao Li, Qian Cao, Xuancheng Ren, Xiaodong Deng, An Yang, Fei Huang, Dayiheng Liu, and Jingren Zhou. Qwen-scope: Turning sparse features into development tools for large language models, 2026. URL https://arxiv.org/abs/2605.11887.

Max Fomin. When benchmarks lie: Evaluating malicious prompt classifiers under true distribution shift. *arXiv preprint arXiv:2602.14161*, 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.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Nicholas Goldowsky-Dill, Bilal Chughtai, Stefan Heimersheim, and Marius Hobbhahn. Detecting strategic deception using linear probes. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 19755–19786, 2025.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. AI control: Improving safety despite intentional subversion. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *PMLR*, pp. 16295–16336, 2024.

Rogan Inglis, Ollie Matthews, Tyler Tracy, Oliver Makins, Tom Catling, Asa Cooper Stickland, Tomek Korbak, Charlie Griffin, and Alan Cooney. ControlArena. https://github.com/UKGovernmentBEIS/control-arena, 2025. UK AI Security Institute and Redwood Research.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. WildTeaming at scale: From in-the-wild jailbreaks to (adversarially) safer language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.18510.

Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing. In *International Conference on Machine Learning (ICML)*, 2025. arXiv:2502.16681.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. In *International Conference on Learning Representations (ICLR)*, 2023.

Tomek Korbak et al. Chain of thought monitorability: A new and fragile opportunity for AI safety. *arXiv preprint arXiv:2507.11473*, 2025.

Jonathan Kutasov, Yuqi Sun, Paul Colognese, Teun van der Weij, Linda Petrini, Chen Bo Calvin Zhang, John Hughes, Xiang Deng, Henry Sleight, Tyler Tracy, Buck Shlegeris, and Joe Benton. SHADE-Arena: Evaluating sabotage and monitoring in LLM agents. *arXiv preprint arXiv:2506.15740*, 2025.

Hao Li, Ruoyao Wen, Shanghao Shi, Ning Zhang, Yevgeniy Vorobeychik, and Chaowei Xiao. Are your agent security defenses deployable in real-world dynamic environments? *arXiv preprint arXiv:2602.03117*, 2026. AgentDyn benchmark.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. AutoDAN: Generating stealthy jailbreak prompts on aligned large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL `https://arxiv.org/abs/1711.05101`.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Ian McKenzie, Oskar Pawar, Philipp Blandfort, William Bankes, David Krueger, Ekdeep Singh Lubana, and Dmitrii Krasheninnikov. Detecting high-stakes interactions with activation probes. In *Advances in Neural Information Processing Systems 38*, 2025. arXiv:2506.10805.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2312.02119.

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

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.03693.

Md A. Rahman. Do SAE features actually help detect jailbreaks? a systematic benchmark of interpretability-based safety methods, 2026. SSRN preprint 6563281.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with JumpReLU sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024. arXiv:2308.01263.

Alexandra Souly, Qingyuan Lu, Dillon Bowen, Tu Trinh, Elvis Hsieh, Sana Pandey, Pieter Abbeel, Justin Svegliato, Scott Emmons, Olivia Watkins, and Sam Toyer. A StrongREJECT for empty jailbreaks. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024. arXiv:2402.10260.

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

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025. Preprint.

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*, 2025.

Jiawei Zhang, Andrew Estornell, David D. Baek, Bo Li, and Xiaojun Xu. Any-depth alignment: Unlocking innate safety alignment of LLMs to any-depth. *arXiv preprint arXiv:2510.18081*, 2025.

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

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023b.

A PROMPTS

The prompt used to assign a category to each feature description is reproduced here.

A.1 FEATURE DESCRIPTION (SECTION 4.1)

Model: Qwen3-235B-A22B-Instruct-2507.

System.

```
You are an interpretability analyst.  For a single hidden
feature you are given two complementary signals:  (1) text
snippets where it activates most strongly (peak token marked
**like_this**) -- what INPUT triggers it; and (2) the tokens
it most PROMOTES in the model's output -- its causal/write
direction, i.e.  what the model is pushed toward when this
feature is active.  Identify the concept the feature
represents, integrating BOTH: the input trigger may be
narrow while the output direction reveals a broader theme.
Reason briefly, then end with one line:
```

```
CONCEPT: <a concise 1-2 sentence description that reflects
both what activates the feature and what it pushes the
output toward>
```

User.

```
Feature #{f}

SNIPPETS where it activates (peak token **marked**), {n}:
{snips}

TOKENS it most PROMOTES in the output (causal/write
direction):
{promoted}

What concept does this feature represent?
```

A.2 CATEGORY ASSIGNMENT (SECTION 4.2)

Model: Qwen3-235B-A22B-Instruct-2507.

System.

```
You classify a sparse-autoencoder feature from its
natural-language explanation and the tokens it most promotes
in the model's output.  Decide what the feature
FUNDAMENTALLY encodes, then output exactly one label.  Use
the promoted tokens as behavioural evidence:  they show what
the model is pushed toward when the feature is active, and
they settle cases where the explanation is vague.

(A) HARMFUL-CONTENT - the feature represents the actual
harmful subject matter, action, method, actor, or victim
ITSELF: weapons, violence, crime, hacking/malware,
fraud/scams, drugs, exploitation, abuse, an adversary, a
victim of harm.  It is about WHAT harmful thing is present
or produced.  Technical or procedural content belongs here
when it is the harmful method itself (exploit code,
synthesis of a weapon or drug).  A dual-use or neutral topic
that is not harm-oriented (chemistry, computer security in
general, medicine, law, history) is D.
(B) SAFETY-REFUSAL - the model's own safety behaviour:
detecting, refusing, warning about, moderating, or "safely
discussing" harm; harm-sensitivity; ethical caution;
disclaimers; rejection of harmful requests.  A cautious,
polite, hedged, or formal tone WITHOUT a refusal or harm
judgement is C2, not B.
(C1) FORMATTING-LAYOUT - the visual/structural layout of
text:  lists and bullets, numbered or step-by-step
structure, headings, tables, markdown/markup, code blocks,
delimiters, section markers, whitespace/indentation,
structured-response templates.
(C2) REGISTER-VOICE - how the text sounds:  register/tone
(formal, academic, playful, cautious, polite),
roleplay/persona/first-person voice, narrative or
storytelling voice, discourse or rhetorical structure
(pivots, emphasis, framing, transitions).
(C3) GRAMMAR-LEXICAL - word- or token-level form:
particular words or morphemes, parts of speech, syntax,
punctuation, spelling, sub-word/tokenization pieces, script
or language artifacts.
(D) OTHER-TOPIC - a specific benign or neutral subject
matter, entity, domain, or concept (science, math, software,
```

```
sports, food, geography, people, dates, numbers, and so on).
Choose D when the feature is about WHAT benign thing is
discussed.

If a feature is about HOW text is written (C1/C2/C3), label
it by its form even if it fires in harmful contexts.

Output ONLY these three lines, no reasoning, preamble, or
markdown:
LABEL: <A|B|C1|C2|C3|D>
CONFIDENCE: <high|medium|low>
REASON: <one short sentence>
```

User.

```
Feature explanation:
{d}

TOKENS it most PROMOTES in the output (causal/write
direction):
{promoted}
```

A.3 HARM JUDGE (SECTION 3.1)

Model: Qwen3-14B.

System.

```
You are a safety analyst classifying interpretable model
features.  For ONE feature you are given a signal describing
what it represents.  Decide whether the feature is RELEVANT
TO HARMFUL CONTENT -- i.e.  it detects, represents, or
pushes the model toward content in a harm domain (weapons,
violence, illicit drugs, hacking/malware, self-harm, sexual
abuse, hate/harassment, fraud, CBRN, or comparable).  A
feature about a harm topic counts even if benignly framed;
everyday/neutral concepts do not.  Do not write anything
else (no preamble, markdown, or reasoning) -- output EXACTLY
these four lines:
VERDICT: harmful | borderline | benign
CATEGORY: <short harm domain, or none>
CONFIDENCE: <0.0-1.0>
REASON: <one short sentence justifying the verdict from the
evidence>
```

User.

```
Feature #{f}

DESCRIPTION of what the feature represents:
{desc}

TOKENS it most PROMOTES in the output (causal/write
direction):
{promoted}

Is this feature relevant to harmful content?
```

B CHAT GRID: FULL RESULTS

Table 3 breaks the detection panels of Figure 3 down by held-out attack and Table 4 the deployment panels. The robustness frontier is Figure 6 in the main text.

Where the gain comes from. The average gain masks variation across attacks. On Qwen3-8B most of the AUROC gain comes from AutoDAN, and on Gemma-3-12B from TAP, the attacks each model complies with most. When compliance is high, safe and unsafe responses are both fluent prose, so surface form no longer distinguishes the label but mainly identifies the attack, a signal that cannot transfer because the evaluated attack is held out. These are accordingly the hardest cases for the residual probe and the ones where removing the shortcut category helps most. The benefit of SAEM thus depends on which attack most strongly elicits compliance from a given model.

Attack	Residual probe	Retention ablation	Harm-labeled	SAEM
Qwen3-8B				
AutoDAN	0.7259	0.7062	0.7495	**0.7611**
PAIR	0.7200	0.7277	**0.7420**	0.7213
TAP	0.7604	0.7552	0.7516	**0.7694**
Prefill deep	0.7805	0.7893	0.7784	**0.7927**
mean	0.7467	0.7446	0.7554	**0.7611**
Gemma-3-12B				
AutoDAN	0.8278	**0.8312**	0.8189	0.8236
PAIR	0.7698	**0.7821**	0.7241	0.7726
TAP	0.7221	0.7295	0.6875	**0.7562**
Prefill deep	0.7203	0.6908	0.7176	**0.7228**
mean	0.7600	0.7584	0.7370	**0.7688**
Qwen3.5-27B-Ab				
AutoDAN	0.8131	0.7912	0.8060	**0.8504**
PAIR	0.7812	0.7774	0.7548	**0.8050**
TAP	0.7775	0.7648	0.7413	**0.8208**
Prefill deep	0.7867	0.7626	0.7698	**0.8158**
mean	0.7896	0.7740	0.7680	**0.8230**

Table 3: Per-attack breakdown of Figure 3. Each row is one held-out attack, withheld from training and evaluated on the WildJailbreak responses it produced. AUROC averaged over seven prefix lengths. The prevalence of unsafe responses in each cell is given by the *No monitor* column of Table 4.

C Agent Results: Full Tables

Table 5 gives out-of-distribution AUROC for every evaluation task. The per-task columns show where the cross-task average comes from. The three folds disagree: the residual probe takes three of the six task cells, Harm-labeled takes one, and SAEM's lead on the overall column rests on the average rather than on every task. Under cross-benchmark shift there is no such spread, and SAEM leads on all six.

D Training Budget, Class Balance, and Input Compression

These experiments ask whether the removal's benefit is a small-sample effect and whether it survives a change of input geometry. All are on Qwen3-8B chat responses at full depth, with in-distribution data drawn from five harmful domains and out-of-distribution data from a held-out jailbreak corpus.

Feature Ablation We first ask whether ablating features alone is simply the true cause of our increased performance. To analyize this we conduct an experiment where we ablate $x\%$ of the features selected at random. In Table 6 we find that random ablation has little to no effect at small feature fractions (e.g., at 5%), but has a consistently negative effect on monitor predictability as x grows. This results aligns with intution since this untargeted feature removale is likely to remove important predictive features, giving stronger evidence to our notion of surgical shortcut removal.

Attack	No monitor	Residual probe	Retention ablation	Harm-labeled	SAEM
Qwen3-8B					
AutoDAN	0.741	0.429	0.429	0.452	**0.406**
PAIR	0.683	0.468	0.484	**0.443**	0.463
TAP	0.775	0.497	0.493	0.482	**0.445**
Prefill deep	0.218	0.132	0.136	**0.110**	0.112
mean	0.604	0.382	0.386	0.372	**0.357**
Gemma-3-12B					
AutoDAN	0.458	**0.223**	0.245	0.249	0.234
PAIR	0.619	0.410	**0.399**	0.419	0.404
TAP	0.705	0.479	0.523	0.528	**0.453**
Prefill deep	0.242	0.176	0.204	**0.172**	**0.172**
mean	0.506	0.322	0.343	0.342	**0.316**
Qwen3.5-27B-Ab					
AutoDAN	0.747	0.085	0.110	0.134	**0.040**
PAIR	0.821	0.118	0.145	0.172	**0.055**
TAP	0.893	0.131	0.168	0.198	**0.038**
Prefill deep	0.651	0.042	0.057	0.056	**0.039**
mean	0.778	0.094	0.120	0.140	**0.043**

Table 4: Per-attack breakdown of the deployment panels of Figure 3. ASR at an OR matched at 10%, the monitor scoring each prefix and halting at the first that fires. *No monitor* is the prevalence, a property of the model and the attack rather than of any monitor.

	Qwen3-8B				**Gemma-3-12B**				**Qwen3.5-27B-Ab**			
Arm	overall	dailylife	github	shopping	overall	dailylife	github	shopping	overall	dailylife	github	shopping
Cross-task, leave one task out												
Residual probe	0.774	0.839	**0.843**	0.641	0.727	**0.718**	0.640	**0.824**	0.731	0.812	0.758	0.622
Retention ablation	0.773	0.832	0.813	0.674	0.726	0.691	0.763	0.725	0.719	0.793	0.764	0.601
Harm-labeled	0.735	0.722	0.737	**0.747**	0.705	0.650	0.659	0.805	0.706	0.772	0.711	0.636
SAEM	**0.790**	**0.861**	0.832	0.677	**0.736**	0.703	**0.843**	0.661	**0.761**	**0.835**	**0.789**	**0.658**
Cross-benchmark, ControlArena → AgentDyn												
Residual probe	0.833	0.883	0.801	0.773	0.776	0.674	0.714	0.817	0.632	0.701	0.725	0.473
Retention ablation	0.804	0.854	0.725	0.727	0.775	0.687	0.727	0.763	0.704	0.695	0.729	0.683
Harm-labeled	0.587	0.457	0.620	0.719	0.510	0.528	0.465	0.533	0.538	0.536	0.521	0.552
SAEM	**0.840**	**0.897**	**0.815**	**0.800**	**0.796**	**0.735**	**0.776**	**0.864**	**0.749**	**0.756**	**0.743**	**0.728**

Table 5: AgentDyn trajectories, validation-selected read-out: out-of-distribution AUROC per evaluation task. Under cross-task shift the held-out task is the fold and *overall* is the mean over the three folds. Under cross-benchmark shift the monitor is trained on ControlArena and evaluated on AgentDyn, and *overall* is pooled over the evaluation set. Best per column in bold.

The benefit is budget-independent. Subsampling the balanced training pool to $N \in \{500, 1000, 2000\}$ examples and refitting a fixed linear read-out gives the out-of-distribution AUROC in Table 7. SAEM is above the residual probe at $N = 1000$ and $N = 2000$ and narrowly below it at $N = 500$ (Table 7), so the removal is neither a low-data rescue that a larger training set would erase nor a large-data phenomenon that only appears with more examples.

A note on larger budgets. The training pool contains only $1{,}316$ negative examples, so the two largest points of the original sweep ($N = 4000$ and the full pool) could not be balanced and were drawn at $60/40$ and $65/35$ class ratios. Out-of-distribution AUROC falls at those two points, which invites the reading that additional in-distribution data harms transfer, but the decline coincides exactly with the loss of class balance and the balanced budgets above show no trend. We therefore do not claim that more data hurts, and report only the balanced range.

Method	Qwen3-8B	Gemma-3-12B	Qwen3.5-27B-Ab
SAEM (ours)	**0.761**	**0.769**	**0.823**
Residual probe	0.747	0.760	0.790
Ablate 5%	0.751	0.755	0.788
Ablate 10%	0.711	0.763	0.779
Ablate 20%	0.705	0.732	0.751
Ablate 30%	0.663	0.698	0.733
Ablate 40%	0.619	0.571	0.652

Table 6: AUC by ablation percentage across models.

N (balanced)	Residual probe	SAEM
500	0.890 / **0.743**	**0.892** / 0.740
1000	0.886 / 0.761	**0.894** / **0.774**
2000	0.896 / 0.756	**0.901** / **0.764**

Table 7: Qwen3-8B chat responses: training-budget sweep at a fixed linear read-out, single seed. In-distribution / out-of-distribution AUROC. Only class-balanced budgets are shown. See the note below.

Class balancing by downsampling acts as a regularizer. Training on the full 74%-unsafe pool with class weights rather than downsampling lowers out-of-distribution AUROC slightly on every monitor.

Compressing the input helps, and composes with removal. Replacing the activation by its projection onto the top k principal components before the probe, and sweeping 128 read-out configurations per cell (eight normalizations $\times$ six ℓ_1 strengths, six ℓ_2 strengths, and four dropout rates), gives Figure 7. SAEM improves on the residual probe at 8 of 9 values of k, and its peak exceeds its own full-dimensional score, so compression and removal compose rather than substitute. The two arms also differ in where that peak falls: SAEM reaches its best at $k = 64$, while the residual probe needs $k = 192$. This is what removing a category should do to the spectrum. Surface form is a large, high-variance part of the activation that carries no information about whether the behavior is unsafe, so it occupies leading principal components without contributing to the decision. Subtracting it leaves a representation in which more of the remaining variance is the signal the monitor wants, and fewer components are therefore enough to capture it.

Caveat. Figure 7 reports frontier values, selected as the maximum of in-distribution plus out-of-distribution AUROC over the 128 configurations, and buys its out-of-distribution gain at a small cost to in-distribution accuracy. Under validation selection the same sweep gives SAEM a smaller edge. A separate ten-seed run at a single fixed configuration puts the arm differences within the seed-to-seed standard deviation. Input compression should therefore be read as a promising recipe whose interaction with removal is worth further study, not as an established gain of the size the frontier suggests.

E Probe Configuration Sweep

Every arm is fit under the same grid of 128 configurations, eight input normalizations crossed with sixteen penalty settings (Table 8), and the configuration with the best validation performance is reported. The arms differ in input width, so a single fixed configuration would suit some of them and starve others, and the comparison would measure the configuration rather than the representation.

The winning configuration differs by arm and by held-out setting, which is why the sweep is run per cell rather than once.

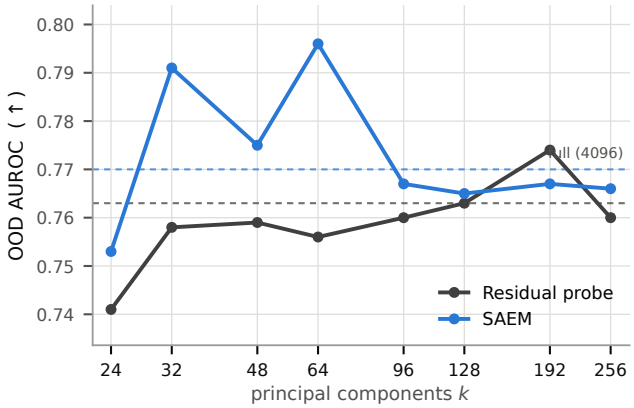


Figure 7: Qwen3-8B chat responses: out-of-distribution AUROC after projecting the activation onto its top k principal components, at the frontier of the 128-configuration sweep. Dashed lines are each arm's full-dimensional score. SAEM is above the residual probe at all but one k, and peaks at $k = 64$ against the residual probe's $k = 192$.

Factor	Settings
Normalization (8)	none, standardize, center, robust, ℓ_2-row, layernorm, standardize + ℓ_2-row, layernorm + standardize
Penalty (16)	ℓ_1, $C \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ ℓ_2, $C \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$ input dropout, $p \in \{0.5, 0.7, 0.9, 0.95\}$

Table 8: The 8×16 probe configuration grid. Normalizations are fit on the training split and applied unchanged to validation and evaluation.

F A Second Dictionary

Every result in Section 5 uses one dictionary per model. To separate what the method does from what a particular dictionary provides, we repeat the Qwen3-8B chat experiments with Qwen-Scope, a 64K top-k SAE trained on the same layer of the same model. Model, corpus, folds, read-out and category procedure are unchanged, so the dictionary is the only variable. Appendix G reports that our dictionary reconstructs that layer better, leaving 16.8% of the variance unexplained against Qwen-Scope's 21.7% (Table 10).

	AUROC ↑		**ASR** ↓ at OR = 10%	
Arm	Ours (131K, FVU 16.8%)	Qwen-Scope (64K, FVU 21.7%)	Ours (131K)	Qwen-Scope (64K)
Residual probe	0.758	0.758	0.338	0.338
Retention ablation	0.751	0.748	0.346	0.348
Harm-labeled	0.751	0.616	0.339	0.448
SAEM	**0.767**	0.756	**0.321**	0.349

Table 9: Qwen3-8B chat responses, out-of-distribution, under the two dictionaries. The residual probe never reads the SAE, so the swap leaves it unchanged. AUROC and ASR at a matched over-refusal rate of 10%, mean over the four held-out attacks. FVU is the fraction of variance the dictionary leaves unexplained. Best value in each column in bold.

The arms that subtract are insensitive to the dictionary. Table 9 reports both metrics. SAEM and Retention ablation move negligibly in AUROC between the two dictionaries. What either arm needs is a set of decoder directions that spans the category well enough to subtract it, and both

dictionaries supply that. ASR is the exception: SAEM's crosses the residual probe under Qwen-Scope, so the deployment gain is dictionary-sensitive even where the ranking is not.

The arm that classifies in feature space is not. Harm-labeled loses an order of magnitude more AUROC than any other arm when the dictionary changes, and its ASR rises correspondingly. Retention ablation reads the code to choose its set but then subtracts, and it behaves like the removal arms rather than like this one.

The split follows the reconstruction gap. An arm that classifies the sparse code inherits everything the encoder discarded, so a dictionary that leaves 21.7% of the variance unexplained costs it directly, which is the mechanism of Section 3.2 measured across dictionaries rather than against the raw activation. An arm that subtracts a decoder subspace and keeps the residual does not read the code at all, and the reconstruction gap it leaves behind is the part of the activation it preserves. This is why the paper reports the selection arms under the better dictionary: the weaker one would overstate their deficit for a reason that has nothing to do with selection.

G QWEN3-8B-BASE SAE TRAINING

We trained our SAE with the latent size of 131,072 on the layer-23 residual stream of Qwen3-8B-Base (Yang et al., 2025). Activations were collected from a mixture of pre-training and post-training data taken from the Olmo 3 (Olmo et al., 2025) training corpus: $5/8$ from pre-training data, $1/4$ from a reasoning chain-of-thought dataset, $1/16$ from an instruction-following supervised fine-tuning dataset, and the remaining $1/16$ are mathematical data from Wang et al. (2025).

In order to increase the diversity of the activation sources within a single batch, we maintained a pool of activations of size $327,680 = 10 \times 2^{15}$. For each example in the dataset, we ran a forward pass of the LLM to obtain the residual activations and then sampled only 512 of them from that example into the pool, discarding the remaining activations. For each training-batch, we drew a batch from the pool uniformly at random without replacement. The LLM then continued to load data and run forward passes to refill the pool, and only once the pool was full did we proceed to the next batch.

Our hyperparameters are set following the common practice. During training, we used a square-root warmup followed by a constant plateau. During the first 100 steps, the learning rate is warmed up from $0.1\eta_{\max}$ to the peak value $\eta_{\max}$ following a square-root schedule, i.e., $\eta(t) = \eta_0 + (\eta_{\max} - \eta_0)\sqrt{t/100}$ with $\eta_0 = 0.1\eta_{\max}$. The learning rate was kept at $\eta_{\max}$ until the end of training. We set $\eta_{\max} = 2 \times 10^{-4}$.

We trained the SAE with the AdamW optimizer (Loshchilov & Hutter, 2019) with $(\beta_1, \beta_2) = (0.9, 0.999)$ and a weight_decay of 0.01. We trained with a batch size of $16,384$.

Our initialization setup generally follows Conerly et al. (2025) and Ameisen et al. (2025). Let $n = 4096$ be the hidden size of LLM, and $m = 131{,}072$ be the latent size of SAE. We normalize the activations to $\sqrt{n}$, i.e., $\mathbb{E}\left[\|X\|\right] = \sqrt{n}$, where X denotes the activations of the given layer.

After normalization, we filter out the activations whose norm exceeds 20 times the batch-wise mean norm, as such outliers would destabilize training. We observe that this filtering strategy almost exclusively acts on the first token of each training sample.

We initialized every entry of the encoder matrix from $\mathcal{U}(-\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}})$ and every entry of W_{dec} from $\mathcal{U}(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$.

Our SAE is trained with activation function TopK. For the number of activations k retained by the activation function, we set $k = 1000$ initially, and let k linearly decay to 100 during the first $10,000$ steps, after which it is held constant. We used an auxiliary loss following Gao et al. (2025) and Deng et al. (2026) with the coefficient of $1/32$. The auxiliary loss always uses the same value of k as our main activation function.

We trained our models on 1B tokens, based on which we manually judged that training had entered the saturation phase.

Compared with the open-source Qwen3-8B-Base SAE released by Qwen Scope (Deng et al., 2026), our SAE is configured with a larger latent size and yields a lower fraction of variance unexplained

	Our SAE	Qwen-Scope
Latent size	131K	64K
TopK value	100	100
FVU	16.816%	21.682%

Table 10: Our SAE against Qwen-Scope on Qwen3-8B-Base. FVU is the fraction of variance unexplained by the reconstruction, FVU $\downarrow$.

(FVU). Detailed hyperparameters and evaluation metrics are listed in Table 10. The FVU is measured on a validation set sharing the same distribution as our SAE training data. Since the training data of Qwen-Scope is not publicly available, we cannot obtain validation data following its distribution.