Crosscoding Through Time: Sparse Feature Discovery Across Sequence Positions

Anonymous Author(s)
Affiliation
Address
email

Abstract

Dictionary learning methods - such as Sparse Autoencoders (SAEs) and crosscoders - decompose model activations into human-interpretable building blocks. We introduce *temporal crosscoders*, a simple and flexible framework for feature discovery in Large Language Models (LLMs). To properly evaluate temporal crosscoders we develop TempBench: a panel of synthetic and real-world tasks for evaluating temporal structures. Temporal crosscoders outperform both conventional and temporal architectures in both of our synthetic settings and on two out of four of the real world settings - ahead of other candidate architectures. Most strikingly, they can detect backtracking - a key reasoning behavior - at a 40% higher rate than conventional SAEs, and are 15% more effective in inducing it. Our results establish temporal crosscoders as a simple and flexible framework for feature discovery, both local and temporal. We provide full code at the following anonymous repository: `https://anonymous.4open.science/r/temp-bench-anon/`.

1 Introduction

Sparse autoencoders have become a foundational tool for interpreting large language models. By decomposing activations into sparse, reusable directions, SAEs provide a feature dictionary that supports monosemantic feature discovery, sparse probing, steering, model comparison, and large-scale interpretability workflows [Cunningham et al., 2023, Bricken et al., 2023a].

However, standard SAEs reconstruct each token independently of nearby positions. This assumes that the relevant feature structure is local in sequence position [Lubana et al., 2025]. Natural language does not satisfy this assumption. It has hierarchical syntactic structure [Chomsky, 1956]; tokens are constrained by expectations from prior context [Levy, 2008]; and neural responses to language integrate information over multiple temporal receptive windows, from short phrases to longer narrative contexts [Lerner et al., 2011].

Recent work has made this mismatch explicit, framing it as an architectural issue: standard SAE objectives factor over positions and so impose an independence prior over time, missing latent structure shared across tokens [Lubana et al., 2025, Bhalla et al., 2026]. A complementary observation is that correlations between sequence positions decay approximately as a power law with distance [Cagnetta and Wyart, 2024], so the relevant dependencies are not confined to a single fixed scale. Feature discovery therefore needs a windowed architecture that can capture correlations within a local context while allowing learned features to live at different scales inside that window.

In this work, we make four contributions to temporal feature discovery:

1. We introduce temporal crosscoders section 3, a simple and flexible architecture for capturing temporal structure in LLMs.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

2. We generalize existing synthetic benchmarks for ground-truth feature recovery to the temporal setting section 4.
3. We assemble four real-world evaluations with clear behavioral signals for comparing dictionary-learning architectures section 5.
4. We combine these synthetic and real-world evaluations into **TempBench**, an evaluation suite for conventional and temporal SAE architectures, and use it to benchmark the current architectural frontier under matched conditions.

Across *TempBench*, temporal aggregation helps when the target feature is supported by evidence distributed over nearby tokens, and hurts on tasks where per-token or length-correlated signals dominate.

2 Related Work

Sparse autoencoders and synthetic benchmarks. Sparse autoencoders (SAEs) trained on the residual stream have become the standard tool for decomposing language-model activations into a sparse basis of features [Cunningham et al., 2023, Bricken et al., 2023a, Templeton et al., 2024, Gao et al., 2024, Bussmann et al., 2024, Lieberum et al., 2024]. Because SAE features are derived rather than ground-truth, evaluating *which* dictionary recovers *what* requires either downstream proxies (loss recovery, sparse probing [Kantamneni et al., 2025, Bhalla et al., 2026]) or synthetic data with known features. Chanin [2025] establish the canonical synthetic setting: ground-truth feature directions $\{\mathbf{f}_i\}$ are fired according to a distribution that factorizes across sequence positions, so the recoverable ground truth is purely local. This setup has been used to study superposition and feature absorption [Makelov et al., 2024, Karvonen et al., 2024], but by construction it cannot test whether a dictionary recovers structure that is shared across positions, because no such structure exists in the data. Our section 4 generalizes this setting by promoting the firing distribution to a Hidden Markov Model, which (a) preserves the local synthetic setting as a special case and (b) admits two distinct ground truths — emission directions and hidden-state directions — that dissociate per-position from cross-position recovery.

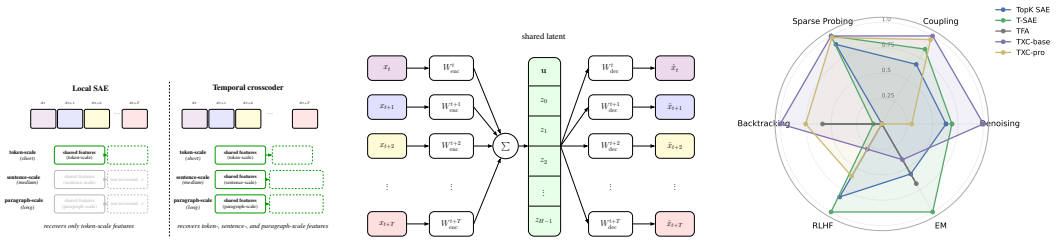
Crosscoders. Crosscoders [Lindsey et al., 2024] extend SAEs by encoding and decoding across an additional axis: a single sparse latent vector explains multiple inputs simultaneously, with axis-specific decoder weights. The original proposal targets the layer axis (a single feature reconstructs activations at several layers) and motivates the architecture by model-diffing applications — comparing base and finetuned models, or models from different training runs [Minder et al., 2026, Lindsey et al., 2024] — and cross-layer feature discovery. To our knowledge, the crosscoder framework has not been applied to the sequence axis. Our temporal crosscoder (section 3) is the temporal analogue.

Temporal sparse dictionaries. Two recent proposals share our motivation that per-token dictionaries fragment multi-token concepts, and we use both as baselines. The temporal SAE of Bhalla et al. [2025] introduces a T-position window with a per-position encoder/decoder and a multi-distance contrastive loss that aligns features describing the same temporal pattern across nearby offsets. Per-position weights mean that T-SAE features are still anchored to single positions; the contrastive loss is what couples them across the window. The temporal feature analyzer (TFA) of Lubana et al. [2025] takes a different route, introducing a temporal prior on the dictionary itself. The temporal crosscoder differs from both by tying every position of the window to a single shared latent and letting full-rank cross-position decoders distribute that latent's mass.

Systematic benchmarking. Each of the temporal architectures above was introduced and evaluated in isolation. There is no existing study that evaluates the candidate architectures side-by-side on a panel of comparable tasks. The closest prior efforts are SAEBench [Karvonen et al., 2025] and the sparse-probing panel of Kantamneni et al. [2025], both of which standardise downstream evaluation but cover only per-token dictionaries.

3 Temporal Crosscoders

Let $\mathbf{x}_t \in \mathbb{R}^d$ denote the residual-stream activation at position t, and let $\mathbf{X}_t = (\mathbf{x}_t, \ldots, \mathbf{x}_{t+T-1})$ be a temporal window. A temporal crosscoder learns one sparse latent vector $\mathbf{u}_t \in \mathbb{R}^H$ shared across the



(a) Local dictionaries recover token-scale features, while temporal dictionaries can recover longer-context features.

(b) TXC encodes a temporal window into one shared sparse latent and decodes back to each position.

(c) TempBench evaluates architectures across synthetic and real-model tasks.

Figure 1: Schematic motivation, architecture, and evaluation suite for temporal crosscoders.

whole window:

$$\mathbf{u}_t = \sigma\left(\sum_{\tau=0}^{T-1} W_{\text{enc}}^{(\tau)}\mathbf{x}_{t+\tau} + \mathbf{b}_{\text{enc}}\right), \qquad \hat{\mathbf{x}}_{t+\tau} = W_{\text{dec}}^{(\tau)}\mathbf{u}_t + \mathbf{b}_{\text{dec}}^{(\tau)}.$$

This is the temporal analogue of a layerwise acausal crosscoder [Lindsey et al., 2024]: the layer index is replaced by a relative position index. A feature may reconstruct primarily one token, recovering local-SAE behavior, or distribute decoder mass across several positions, representing a temporally extended pattern.

We train temporal crosscoders with a reconstruction loss over the full window, together with explicit sparsity from the latent activation function. In our default implementation, σ is BatchTopK [Bussmann et al., 2024], a variant of TopK [Gao et al., 2024], applied to the shared latent vector for each window, and we set decoder biases to zero so that reconstructive mass is assigned to features rather than biases. We define and use two TXC variants in this work; definitions and model decisions are explained in section A. Our base temporal crosscoder TXC-base uses only this acausal window-reconstruction objective. The larger TXC-pro variant adds two temporal inductive biases: a Matryoshka objective over nested latent groups, encouraging useful feature dictionaries at multiple effective widths, and a multi-distance contrastive loss over shifted windows, encouraging features that describe the same temporal pattern to align across nearby offsets.

3.1 Compared Architectures and Matching

Across TempBench, we compare TXC-base and TXC-pro against three baselines: TopK SAE [Gao et al., 2024], T-SAE [Bhalla et al., 2025], and TFA [Lubana et al., 2025]. These cover local dictionaries, temporal dictionaries with position-local structure, temporal-prior methods, and crosscoder-style sharing along a non-temporal axis.

Unless otherwise stated, all architectures are trained on the same activation cache and compared at fixed hookpoint, dictionary width, and expected sparsity. We write k_{pos} for the active-latent budget per sequence position: a per-token dictionary uses k_{pos} active latents at each token, while a windowed architecture of length T receives the matched window budget Tk_{pos}.

4 Benchmarking Temporal Crosscoders in a Synthetic Setting

Synthetic datasets make it possible to evaluate feature recovery against known ground-truth features [Chanin, 2025]. Existing synthetic benchmarks define feature activations independently at each sequence position, so the ground-truth feature structure is local. To benchmark temporal architectures, we generalize this setting by making the feature-firing pattern a stochastic process across sequence positions.

Consider a synthetic activation sequence

$$\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T), \qquad \mathbf{x}_t \in \mathbb{R}^d,$$

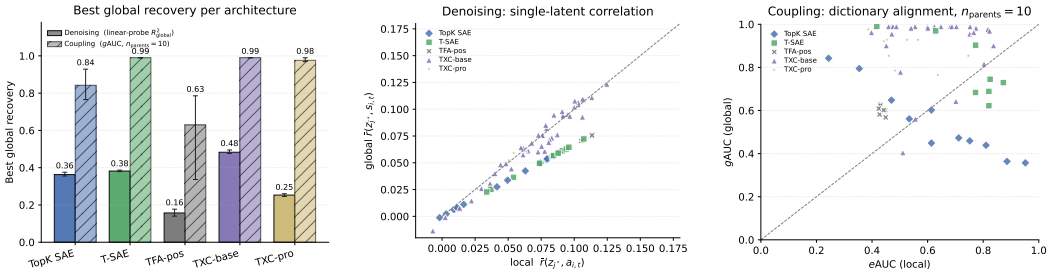


Figure 2: Synthetic global-feature recovery. *Left:* best global recovery per architecture, using linear-probe R^2_{global} for Denoising and gAUC for Coupling. *Middle:* Denoising single-latent correlations with noisy emissions $a_{i,t}$ versus clean hidden states $s_{i,t}$; the diagonal marks equal local and global correlation. *Right:* Coupling dictionary alignment with emission directions (eAUC) versus hidden-chain footprints (gAUC). Temporal architectures recover more global structure in Coupling, while TXC-base gives the strongest Denoising recovery.

defined over a temporal window of length T. A synthetic setting is specified by three pieces of data. First, a set of N ground-truth feature directions

$$\mathcal{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_N\}, \qquad \mathbf{f}_i \in \mathbb{R}^d.$$

Second, a probability distribution P_T over binary feature-firing masks

$$A = (\mathbf{a}_1, \ldots, \mathbf{a}_T) \in (\{0,1\}^N)^T,$$

where

$$\mathbf{a}_t = (a_{t,1}, \ldots, a_{t,N})$$

and $a_{t,i} = 1$ indicates that feature i fires at position t. Third, a distribution M_T over feature magnitudes

$$M = (\mathbf{m}_1, \ldots, \mathbf{m}_T), \qquad \mathbf{m}_t = (m_{t,1}, \ldots, m_{t,N}).$$

A synthetic activation is then constructed as

$$\mathbf{x}_t = \sum_{i=1}^{N} a_{t,i} m_{t,i} \mathbf{f}_i.$$

Thus, the feature directions specify what can be recovered, while P_T specifies the local and temporal structure of when those features appear. The conventional synthetic setting is the special case where P_T and M_T factorize across sequence positions. In our setting, local correlations are encoded by the one-position distribution over $\mathbf{a}_t$, while temporal correlations are encoded by the joint distribution over $\mathbf{a}_{1:T}$.

A natural way to specify the firing-mask distribution P_T is by an edge-emitting Hidden Markov Model (HMM), following the convention used in computational mechanics [Crutchfield and Young, 1989, Crutchfield, 2012, Shai et al., 2024, 2026]. An HMM $\mathcal{H}$ consists of

$$\mathcal{H} = (\mathcal{X}, \mathcal{S}, \boldsymbol{\eta}^{\emptyset}, \{T^{(x)}\}_{x \in \mathcal{X}}),$$

where the observed alphabet is the set of local firing masks,

$$\mathcal{X} = \{0,1\}^N.$$

Thus, each emitted symbol $x \in \mathcal{X}$ is a full binary vector indicating which features fire at a single sequence position. The set $\mathcal{S}$ is the latent state space, $\boldsymbol{\eta}^{\emptyset} \in \Delta(\mathcal{S})$ is the initial distribution over latent states, and $T^{(x)}$ is the symbol-labeled transition matrix with entries

$$T^{(x)}_{h,h'} = \Pr(h_{t+1} = h',\, \mathbf{a}_t = x \mid h_t = h).$$

The HMM generates the distribution over firing patterns by

$$P_T(\mathbf{a}_{1:T}) = \boldsymbol{\eta}^{\emptyset\top} T^{(\mathbf{a}_1)} T^{(\mathbf{a}_2)} \cdots T^{(\mathbf{a}_T)} \mathbf{1}.$$

A dictionary learning architecture can be evaluated at two levels. At the local level, we ask whether its decoder directions recover the emitted feature directions $\mathbf{f}_i$ and whether its activations track the local firing indicators $a_{t,i}$. At the global level, we ask whether its learned latents track the HMM state, or the corresponding hidden feature directions induced by those states. In non-synthetic settings, this corresponds to the distinction between learning local cues associated with a behavior and learning a latent feature for the behavior itself.

We benchmark dictionary learning architectures on two HMM special cases in which the observed local feature firings are only indirect evidence for a cleaner hidden temporal process. The two benches differ in where the local/global distinction appears. In the **Denoising** bench, each emitted feature has its own hidden Markov chain, so there is a one-to-one correspondence between hidden states $s_{i,t}$ and noisy local firings $a_{i,t}$. Since both variables are associated with the same residual-stream direction $\mathbf{f}_i$, local-vs-global recovery cannot be read off from decoder alignment; it must be read off from the learned activation traces. In the **Coupling** bench, a smaller set of hidden chains drives a larger emission alphabet through OR-gate coupling and per-token noise. There, local emission directions and hidden-chain footprints are geometrically distinct, so local-vs-global recovery can be measured directly from decoder alignment.

The Denoising bench. The Denoising bench is a panel of $N = 20$ independent two-state Markov chains, one per feature. Each chain has a sticky hidden state $s_{i,t}$ with self-correlation $\rho = 0.7$, and emits a noisy firing $a_{i,t}$ through a Bernoulli channel with hit rate $p_B = 0.625$; the activation reads out the noisy firings,

$$\mathbf{x}_t = \sum_{i=1}^{N} a_{i,t}\, m_{i,t}\, \mathbf{f}_i.$$

Thus the clean hidden state $s_{i,t}$ and noisy emission $a_{i,t}$ differ in their time traces, not in their per-position residual-stream direction. A learned feature whose decoder points along $\mathbf{f}_i$ may either track the noisy emission $a_{i,t}$ or use temporal context to recover the cleaner hidden state $s_{i,t}$.

We evaluate this distinction at the level of the matched feature's activation trace. For each ground-truth feature i, let $j(i)$ be the learned feature whose decoder slice has the highest cosine similarity with $\mathbf{f}_i$. We define

$$r_{\mathrm{local},i} = \mathrm{corr}(z_{j(i)}, a_i), \qquad r_{\mathrm{global},i} = \mathrm{corr}(z_{j(i)}, s_i),$$

where correlations are taken over held-out sequence positions. The single-latent scatter in section 4 plots $(r_{\mathrm{local},i}, r_{\mathrm{global},i})$. A per-token model that merely tracks the noisy emission should lie near the no-denoising floor

$$r_{\mathrm{global},i} \approx \mathrm{corr}(a_i, s_i)\, r_{\mathrm{local},i}.$$

The diagonal $y = x$ is a stronger reference line: points above it are more correlated with the clean hidden state than with the noisy emission. We also report a full-code linear probe, ridge-regressing $s_{i,t}$ on the learned latent code and reporting R^2_{global} in the headline bar chart. In section 4, per-token baselines remain near the no-denoising floor. TXC-base moves above this floor as the window length grows and achieves the best full-code denoising score ($R^2_{\mathrm{global}} = 0.48$). TXC-pro does not improve on this setting, suggesting that the additional temporal objectives are not uniformly helpful for this simple hidden-state recovery task.

The Coupling bench. The Coupling bench removes the one-to-one correspondence between hidden chains and emitted features. A set of $K = 10$ hidden Markov chains drives a larger alphabet of $M = 20$ local emissions through a binary coupling matrix $C \in \{0,1\}^{M \times K}$. At each position, emission j is active when at least one of its parent chains is active, followed by a Bernoulli noise channel:

$$a_{j,t} \sim \mathrm{Bernoulli}\left(p_B\, \mathbb{1}\left[\sum_{k=1}^{K} C_{jk} h_{k,t} \geq 1 \right] \right),$$

with $p_A = 0$ in the off state. The observed activation is

$$\mathbf{x}_t = \sum_{j=1}^{M} a_{j,t}\, m_{j,t}\, \mathbf{f}^{\mathrm{emit}}_j.$$

This setting has two spatial ground truths. The local ground truth is the set of emission directions

$$\{\mathbf{f}_j^{\text{emit}}\}_{j=1}^{M},$$

which are directly written into the activation. The global ground truth is the set of hidden-chain footprints

$$\mathbf{f}_k^{\text{hid}} = \text{normalize}\left(\sum_{j:C_{jk}=1} \mathbf{f}_j^{\text{emit}}\right), \qquad k = 1, \ldots, K.$$

Unlike in the Denoising bench, local and global recovery are now visible in the dictionary geometry itself: a locally aligned dictionary points at emission directions, while a globally aligned dictionary points at hidden-chain footprints. We therefore evaluate the Coupling bench with decoder-alignment AUCs. The local score $e\text{AUC}$ is the feature-recovery AUC against the M emission directions, and the global score $g\text{AUC}$ is the feature-recovery AUC against the hidden-chain footprints.

We study coupling-density regimes that vary how much the hidden-chain footprints overlap. The moderate-overlap regime takes $n_{\text{parents}} = 5$, so each emission is driven by a random half of the hidden chains while different hidden chains still have distinct child sets. We also include a dense-overlap stress test with $n_{\text{parents}} = 10$. Since this couples every emission to every hidden chain, the individual hidden chain footprints collapse to a shared aggregate direction; we use this regime as a local-vs-global stress test rather than as a disentanglement benchmark. Both regimes fix $K = 10$, $M = 20$, $\rho = 0.9$, $\pi = 0.05$, $p_B = 0.5$, $d = 256$, $d_{\text{sae}} = 40$, and sweep T and k_{pos} over three seeds. The architectures are the same set used on the Denoising bench.

Coupling bench results. In the dense-overlap stress test, temporal architectures recover the aggregate hidden footprint while per-token-only architectures are pulled toward local emission directions. T-SAE and TXC-base both reach $g\text{AUC} = 0.99$ at $k_{\text{pos}} = 1$, and TXC-pro reaches 0.98. The TopK SAE follows the opposite trend, dropping from $g\text{AUC} = 0.92$ at $k_{\text{pos}} = 1$ to 0.44 at $k_{\text{pos}} = 8$ as it reallocates dictionary capacity from the aggregate hidden footprint to the many local emission directions. The decoder-alignment scatter in section 4 makes this tradeoff concrete: temporal architectures remain high in global alignment across a wide range of local-alignment scores, while TopK SAE traces a down-right path as k_{pos} grows. Across the full $(p_B, n_{\text{parents}})$ sweep, the temporal advantage is largest at moderate-to-high coupling overlap, with a peak $g\text{AUC}$ gap of ≈ 0.47.

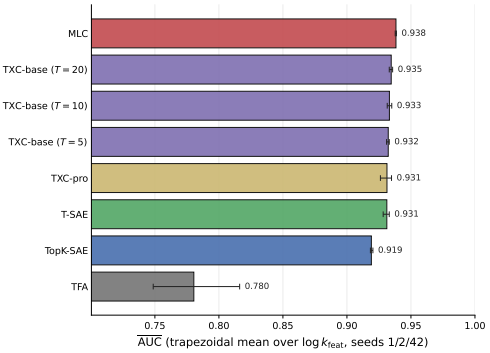
5 Real-Model Evaluations

The synthetic benchmarks above test whether a dictionary can recover known temporal ground truth. We now ask whether the same temporal inductive bias is useful in trained language models, where the relevant features are not known in advance and must be judged through downstream behavior. We evaluate four complementary settings: sparse probing for static concept readout, backtracking for causal control of a multi-token reasoning behavior, emergent misalignment for steering and detecting a finetuned behavioral shift, and HH-RLHF preference decomposition as a negative control where response length is a strong confounder. Across these settings, temporal aggregation helps when the target behavior is distributed across nearby tokens, but can hurt when the benchmark rewards length-correlated or per-token features. We include qualitative autointerpretation and per-token activation visualizations in section G.

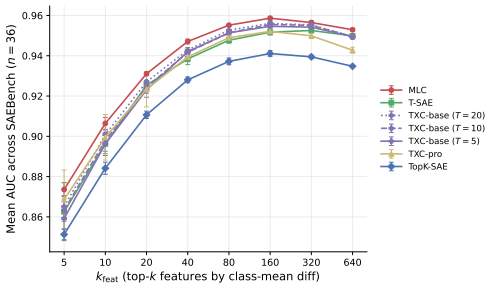
5.1 Sparse Probing

Sparse probing tests whether dictionary features support simple classifiers for human-interpretable text concepts. We evaluate `google/gemma-2-2b-it` layer 13 on the 36-task SAEBench panel, with all architectures trained at matched expected per-token $L_0 = 20$. For each task and feature budget $k_{\text{feats}} \in \{5, 10, 20, 40, 80, 160, 320, 640\}$, we select features by training-split class-mean difference and fit an ℓ_1-regularized logistic probe. Full architecture, pooling, task, and aggregation details are in section C.

Figure 3 shows that MLC leads at $\overline{\text{AUC}} = 0.907$, followed by TXC-base at 0.899–0.902, T-SAE/TXC-pro at 0.897–0.899, and TopK SAE at 0.886. The temporal-family gap over TopK SAE is small but larger than seed variation, appears by $k_{\text{feats}} \geq 20$, and saturates near 160. We read this as evidence that sparse probing benefits from temporal-window features, but that most of the gain comes from having temporal aggregation at all rather than from a specific TXC variant.

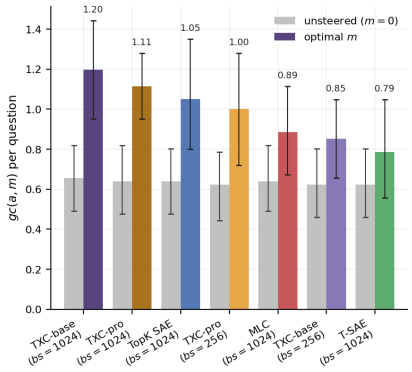


(a) $\overline{\text{AUC}}$ across the feature-budget sweep.

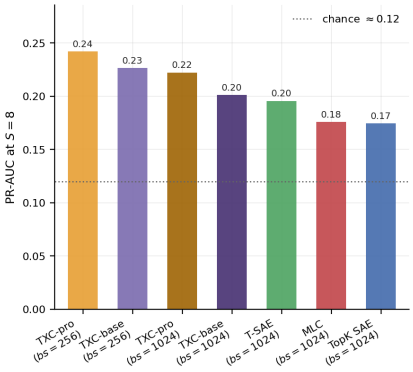


(b) AUC vs. number of probe features, with TFA omitted for scale.

Figure 3: Sparse probing on `google/gemma-2-2b-it` layer 13 over 36 SAEBench tasks. Temporal methods cluster near the top in $\overline{\text{AUC}}$ and outperform the per-token TopK SAE baseline; the advantage appears by $k_{\text{feats}} \geq 20$ and saturates near 160. Full curves including TFA are in fig. 7.



(a) Inducement: genuine backtracking at each architecture's optimal steering magnitude.



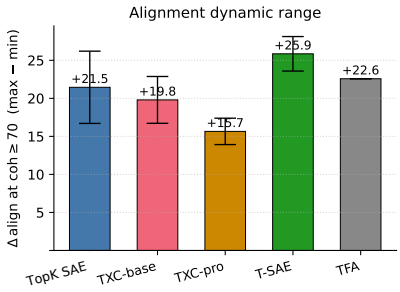
(b) Detection: sparse-probe PR-AUC at top-$S = 8$ features.

Figure 4: Backtracking inducement and detection on DeepSeek-R1-Distill-Llama-8B. TXC-base gives the strongest causal inducement, while TXC-pro gives the best sparse-probe detection above the class-prior baseline. A qualitative example is shown in fig. 12.

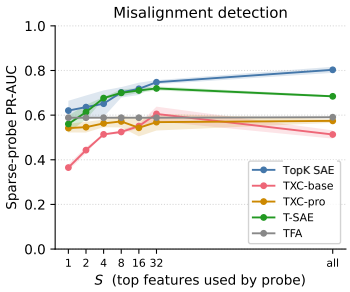
5.2 Inducing and Detecting Backtracking in a Reasoning Model

Reasoning models exhibit backtracking: the model commits to a partial argument, notices a problem, and reverses course. Because this signature spans multiple tokens, it is a natural testbed for temporal-window features. We follow the steering-vector setup of Ward et al. [2025], mining directions from `Llama-3.1-8B` and applying them at layer 10 of `DeepSeek-R1-Distill-Llama-8B`. The cohort contains 61 MATH-500 questions, and the cut25 protocol resamples continuations under steering across a grid of magnitudes.

We evaluate both causality and detection. Inducement counts genuine backtracking events under a Sonnet-4.6 judge, reported against the shared unsteered floor; detection trains sparse probes to distinguish backtracking from non-backtracking continuations under grouped cross-validation. Figure 4 shows that TXC-base gives the strongest causal inducement, while TXC-pro gives the best sparse-probe detection above the class-prior baseline. Full magnitude curves, judge rubric, and robustness checks are in section F.



(a) Steering dynamic range at coh ≥ 70.



(b) Sparse-probe PR-AUC at $S = 16$.

Figure 5: Steering and detection for emergent misalignment in `Qwen-2.5-7B-Instruct` with a bad-medical-advice LoRA. T-SAE performs best on both axes, while TXC variants underperform; full α-frontiers are in figs. 8 and 9.

5.3 Emergent Misalignment

Emergent misalignment (EM) occurs when a model finetuned on a narrow misaligned task generalizes to broader misaligned behavior. Since EM responses are coherent multi-token behaviors and have been controlled by steering on residual-stream directions, we compare architectures on two axes: steering control and sparse-probe detection.

We evaluate `Qwen-2.5-7B-Instruct` finetuned with a bad-medical-advice LoRA. For steering, we adapt the Wang-stage procedure to select candidate features and report the maximum alignment dynamic range achievable above a coherence threshold of 70. For detection, we train sparse probes to distinguish aligned from misaligned rollouts and report PR-AUC. Figure 5 shows the opposite pattern from backtracking: T-SAE performs best on both axes, while TXC variants underperform. Full steering frontiers are in section D.

5.4 Decomposing HH-RLHF Preference Data

HH-RLHF pairs each prompt with a preferred `chosen` and dispreferred `rejected` response Bai et al. [2022]. Since Bhalla et al. [2025] use this dataset to test whether temporal features recover multi-token preference signals, we rerun their §4.5 / Appendix B.1 protocol on our locked architectures. We rank features by $\mu_{\text{rejected}} - \mu_{\text{chosen}}$ over response tokens and classify top features by correlation with response-length difference.

The result is a negative case for TXC. T-SAE at the paper setting $k = 20$ recovers $14/20$ semantic top features, no length-spurious features, and 63% semantic mass. At the larger $k_{\text{win}} = 500$ budget, TopK SAE and T-SAE both fall to roughly $10/20$ semantic features and 50% semantic mass. TXC performs worst on this static decomposition metric, with $7/20$ semantic and 3 length-spurious features. We read this as evidence that temporal aggregation can help causal interventions spanning multiple tokens, as in section 5.2, while hurting rank-based interpretability when response length is a confounder. Full summary plots, scatter plots, and autointerpreted labels are in section E.

6 Limitations

Our work has two key limitations. Although we believe TempBench provides useful guidance to the dictionary-learning community, it is not a full map of how each architecture behaves. Our results, summarized in fig. 1c, tentatively suggest that different temporal architectures, namely TXCs and T-SAEs, may be complements rather than substitutes: TXCs perform well on synthetic temporal recovery and backtracking, while T-SAEs perform better on emergent misalignment and HH-RLHF decomposition. A more systematic temporal benchmark, across both synthetic processes and real-world tasks, would help clarify the capability profile of each architecture and is an important direction for further work.

Second, we focus on mapping the capability profile of temporal crosscoders and not a theoretical account of when and why they outperform. This leads to three limitations. First, we cannot yet predict in advance which real-world tasks will favor TXCs, T-SAEs, standard SAEs, or crosscoders. Second, we do not derive optimal hyperparameters for TXCs, and we believe it likely that the flexibility of the TXC framework will allow significant improvement on the results reported here. Third, an intriguing question left unanswered by our work is whether the TSAE architectures can be effectively incorporated into the TXC framework. This is particularly important since these two architectures had complementary capability profiles on TempBench and combining the strengths of both into a single architecture is an exciting possibility for future work.

Taken together, these gaps mean that significant work remains both to fully understand the TXC framework and to fully exploit it. Mapping the regimes in which a temporal inductive bias pays off, and combining the architectural strengths of TXCs and T-SAEs into a single dictionary, are in our view the most exciting directions this paper points toward.

7 Discussion

Temporal crosscoders are currently best understood as a flexible addition to the sparse-dictionary toolkit. Across TempBench, TXCs help when the target feature is supported by evidence spread over nearby tokens: they recover hidden temporal variables in the synthetic benches and perform strongly on sparse probing and backtracking detection/inducement. The negative cases are equally informative: on emergent misalignment and HH-RLHF preference decomposition, temporal aggregation can diffuse representations or amplify length-correlated artifacts. Thus, the right unit of explanation is task-dependent: some features are token-local, while others are temporally extended.

Several directions follow naturally. First, TXCs currently use short fixed windows. Scaling to longer contexts is a confirmed priority, and will likely require additional structure such as hierarchical windows, low-rank methods, state-space compression, or tensor-network factorizations [Orús, 2014, 2019]. Second, transformer residual streams contain skip connections across layers, so temporal feature discovery should eventually account for both sequence structure and residual-stream composition [Elhage et al., 2021]. Combining TXCs with cross-layer crosscoders is one direct way to test this. Third, the architecture leaves substantial room for improvement: adaptive window lengths, stronger contrastive losses, predictive objectives, and multi-scale training could separate genuine temporal features from confounds such as response length and feature density. The central question is therefore not whether temporal dictionaries are always better, but when model features are best represented as local directions, temporal directions, or interactions between the two.

References

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.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023a. https://transformer-circuits.pub/2023/monosemantic-features.

Ekdeep Singh Lubana, Can Rager, Sai Sumedh R. Hindupur, Valérie Costa, Greta Tuckute, Oam Patel, Sonia Krishna Murthy, Thomas Fel, Daniel Wurgaft, Eric J. Bigelow, Johnny Lin, Demba Ba, Martin Wattenberg, Fernanda Viegas, Melanie Weber, and Aaron Mueller. Priors in time: Missing inductive biases for language model interpretability. *arXiv preprint arXiv:2511.01836*, 2025. URL `https://arxiv.org/abs/2511.01836`.

N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956. doi: 10.1109/TIT.1956.1056813.

Roger Levy. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177, 2008.

Yulia Lerner, Christopher J Honey, Lauren J Silbert, and Uri Hasson. Topographic mapping of a hierarchy of temporal receptive windows using a narrated story. *Journal of Neuroscience*, 31(8): 2906–2915, 2011. doi: 10.1523/JNEUROSCI.3684-10.2011.

Usha Bhalla, Alex Oesterling, Claudio Mayrink Verdun, Himabindu Lakkaraju, and Flavio P. Calmon. Temporal sparse autoencoders: Leveraging the sequential nature of language for interpretability, 2026. URL https://arxiv.org/abs/2511.05541.

Francesco Cagnetta and Matthieu Wyart. Towards a theory of how the structure of language is acquired by deep neural networks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 83119–83163. Curran Associates, Inc., 2024. doi: 10.52202/079017-2645. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/9740da1c07c7b451af14e11523f95271-Paper-Conference.pdf.

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

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. *arXiv preprint arXiv:2406.04093*, 2024. URL https://arxiv.org/abs/2406.04093.

Bart Bussmann, Patrick Leask, and Neel Nanda. BatchTopK sparse autoencoders, 2024. URL https://arxiv.org/abs/2412.06410.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In *Proceedings of the 7th BlackboxNLP Workshop*, pages 278–300, 2024. URL https://arxiv.org/abs/2408.05147.

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.

David Chanin. Can saes recover true features from token embeddings? https://colab.research.google.com/drive/1lLTvn6ksjjVZOJtF4va_mkDGFf8TsZ7B?usp=sharing, 2025. Colab notebook.

Aleksandar Makelov, George Lange, and Neel Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control, 2024. URL https://arxiv.org/abs/2405.08366.

Adam Karvonen, Benjamin Wright, Can Rager, Rico Angell, Jannik Brinkmann, Logan Smith, Claudio Mayrink Verdun, David Bau, and Samuel Marks. Measuring progress in dictionary learning for language model interpretability with board game models. *Advances in Neural Information Processing Systems*, 37:83091–83118, 2024.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Tom Conerly, Joshua Baston, and Chris Olah. Sparse crosscoders for cross-layer features and model diffing. Transformer Circuits Thread, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Julian Minder, Clément Dumas, Caden Juang, Bilal Chugtai, and Neel Nanda. Overcoming sparsity artifacts in crosscoders to interpret chat-tuning, 2026. URL https://arxiv.org/abs/2504.02922.

Usha Bhalla, Alex Oesterling, Claudio Mayrink Verdun, Himabindu Lakkaraju, and Flavio P. Calmon. Temporal sparse autoencoders: Leveraging the sequential nature of language for interpretability. *arXiv preprint arXiv:2511.05541*, 2025. URL https://arxiv.org/abs/2511.05541.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, Demian Till, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability, 2025. URL https://arxiv.org/abs/2503.09532.

James P. Crutchfield and Karl Young. Inferring statistical complexity. *Physical Review Letters*, 63(2): 105–108, 1989.

James P. Crutchfield. Between order and chaos. *Nature Physics*, 8(1):17–24, 2012.

Adam S. Shai, Lucas Teixeira, Alexander Gilbert Oldenziel, Sarah Marzen, and Paul M. Riechers. Transformers represent belief state geometry in their residual stream. In *Advances in Neural Information Processing Systems*, 2024.

Adam S. Shai, Loren Amdahl-Culleton, Casper L. Christensen, Henry R. Bigelow, Fernando E. Rosas, Alexander B. Boyd, Eric A. Alt, Kyle J. Ray, and Paul M. Riechers. Transformers learn factored representations, 2026. URL https://arxiv.org/abs/2602.02385.

Francis Rhys Ward et al. Reasoning-finetuning repurposes latent representations in base models. In *ICML 2025 Workshop on Actionable Interpretability*, 2025. arXiv:2507.12638.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014. doi: 10.1016/j.aop.2014.06.013. URL https://doi.org/10.1016/j.aop.2014.06.013.

Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1: 538–550, 2019. doi: 10.1038/s42254-019-0086-7. URL https://doi.org/10.1038/s42254-019-0086-7.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. Transformer Circuits Thread, 2021. URL https://transformer-circuits.pub/2021/framework/index.html. Published December 2021.

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

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors. In *Reasoning and Planning for LLMs @ ICLR2025*, 2025. URL https://arxiv.org/abs/2506.18167.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2103.03874.

Anthropic. Introducing Claude Haiku 4.5, Oct 2025. URL https://www.anthropic.com/news/claude-haiku-4-5. Accessed: 2026-05-06.

A Model Parameters for each architecture

This appendix specifies architecture and training choices held fixed across components. The only per-component free knobs are dictionary width d_{SAE} and sparsity budget k_{pos}, the expected number of active latents per sequence position. Toy data uses $d_{\text{SAE}} = 40$; real-model widths are set by subject-model dimension. All locked choices are in `configs/locked_archs.yaml` and `configs/datasources.yaml`. Component-specific datasource, width, step count, k_{pos} grid, and seed tables appear in the component appendices.

Common training pipeline. All cells use the same trainer: Adam with $\text{lr} = 3{\times}10^{-4}$, batch size $B = 1024$, 1,000 warmup steps, and mixed precision, using `bf16` on H100/H200 and `fp16` with grad-scaler on A40. Toy-data cells run $n_{\text{steps}} \in \{8{,}000, 30{,}000\}$; real-LM cells run $n_{\text{steps}} \in \{10{,}000, 20{,}000\}$. Runs with $n_{\text{steps}} < 1000$ are discarded as smoke runs. Across the seven components, the paper uses $\approx 5{,}700$ trained checkpoints and $\approx 6{,}200$ leaderboard eval rows. Per-component counts and the full (arch, datasource, seeds) breakdown are given below. Subject-model activations are cached once per datasource and reused by all architectures at matched hookpoint. Sparsity is matched by k_{pos}: per-token dictionaries use k_{pos} active latents per token, while a windowed architecture of length T uses

$$k_{\text{win}} = T k_{\text{pos}}.$$

For TXC-pro, matryoshka groups use

$$k_{\text{train}} = 5k_{\text{pos}}, \qquad k_{\text{inference}} = 10k_{\text{pos}}.$$

Anti-dead stack. TopK SAE, T-SAE, TXC-base, and TXC-pro share the same anti-dead stack: (i) per-feature `num_tokens_since_fired` counters with a 10M-token dead threshold; (ii) AuxK loss with $\alpha_{\text{auxk}} = 1/32$, where the top-$\text{aux\_k}$ dead features re-reconstruct the residual; (iii) decoder unit-norm constraint per latent over (T, d_{in}); (iv) decoder-parallel gradient removal on W_{dec}; and (v) geometric-median b_{dec} initialization on the first batch. This ports the Bhalla et al. [2025] anti-dead recipe into the TXC family. Bricken-style hard resampling [Bricken et al., 2023b] is off by default and used only in C6 to match the Qwen-7B medical setup; see section D.

TXC-base. TXC-base is the acausal temporal crosscoder from section 3 with TopK sparsity and the full anti-dead stack. The locked window length is $T = 5$. For each $\tau \in [0, T)$, encoder slabs

$$W_{\text{enc}}^{(\tau)} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{SAE}}}$$

sum into one shared latent vector $\mathbf{u}_t \in \mathbb{R}^{d_{\text{SAE}}}$ per window. BatchTopK is applied over the latent axis with $k_{\text{win}} = T k_{\text{pos}}$. Decoder slabs

$$W_{\text{dec}}^{(\tau)} \in \mathbb{R}^{d_{\text{SAE}} \times d_{\text{in}}}, \qquad b_{\text{dec}}^{(\tau)} \in \mathbb{R}^{d_{\text{in}}},$$

reconstruct the full T-window. TXC-base uses no matryoshka objective, no contrastive loss, and no Bricken resampling. The free knobs are k_{pos} and d_{SAE}. C3 also trains TXC-base at $T \in \{10, 20\}$ to probe temporal-context length; see section C.1.

TXC-pro. TXC-pro adds three choices to TXC-base. First, the subseq encoder trains with $T_{\max} = 10$ position slabs while sampling $t_{\text{sample}} = 5$ positions per step; all 10 positions are used at evaluation. Second, Matryoshka training uses $H = 8$ nested feature groups, each reconstructing from the first

$$G d_{\text{SAE}}/8, \qquad G = 1, \ldots, 8$$

features. Third, a multi-distance contrastive loss applies inverse-distance-weighted InfoNCE at shifts $\Delta \in \{1, 2\}$. The matryoshka head is disabled at toy scale in C1 and C2 by setting $h_{\text{size}} = d_{\text{SAE}} = 40$, since 5 prefix levels of 8 features do not form a useful hierarchy and the smallest prefix would fall below $k_{\text{train}} = 5k_{\text{pos}}$ at high k_{pos}. The free knobs are again k_{pos} and d_{SAE}.

Baselines. **TopK SAE** [Gao et al., 2024] is a per-token dictionary with TopK at k_{pos} and the anti-dead stack. **Stacked SAE**, used in C1, C2, and C7, is a bank of T independent TopK SAEs, one per position in a T-window, with total window budget $k_{\text{win}} = T k_{\text{pos}}$. It isolates temporal aggregation from cross-position weight sharing. **T-SAE** [Bhalla et al., 2025] is a faithful port of the referenced work's implementation: Matryoshka BatchTopK with $h_{\text{frac}} = 0.20$ high-level features, AuxK loss,

geometric-median b_{dec} initialization, decoder unit-norm projection, raw-dot temporal contrastiveA loss at $\alpha = 1.0$, and threshold-based inference. Its locked $d_{\text{SAE}} = 16{,}384$ matches the original Gemma-2-2B setting. **TFA** [Lubana et al., 2025] is the temporal feature analyzer with 4 attention heads; the no-position-embedding variant is the default, and TFA-pos is reported on synthetic data. **MLC** [Lindsey et al., 2024] is a 5-layer multi-layer crosscoder over the residual-stream layer axis, centered at the component hookpoint. It uses layers 11–15 for Gemma-2-2B-IT C3 and layers 8–12 for Llama-3.1-8B C7. **SAE-arditi** is used only as the C6 baseline to match the Qwen-medical setup, with $d_{\text{SAE}} = 32{,}768$ and $k_{\text{pos}} = 128$; see section D.

Locked versus free choices. The locked TXC-base and TXC-pro definitions above are the only architectural commitments. Across all seven components, only k_{pos} and d_{SAE} vary by component. We do not hill-climb architecture hyperparameters such as T, matryoshka group count, contrastive shifts, AuxK weight, or dead threshold. Training-time augmentations are allowed only when documented in the component appendix; the only such opt-in is Bricken resampling for C6.

B Synthetic Benchmarks: Ground-Truth Correspondences and Probe Setup

The denoising metrics in section 4 (single-latent correlation and linear-probe R^2) rely on two distinct correspondences between ground-truth objects and the architecture's dictionary: one fixed by construction in the data generator, one inferred post-hoc by decoder geometry. This appendix makes both explicit so that the local-vs-global scatters in section 4 can be interpreted unambiguously.

B.1 Construction-time correspondence (feature index ↔ chain)

The Denoising bench generator (`temp_bench.data.toy.markov:markov_chain_support`) draws N orthogonal feature directions $\mathbf{f}_i \in \mathbb{R}^d$ as the rows of a QR-orthogonalised Gaussian matrix, then runs N independent two-state Markov chains $s_{i,t} \in \{0, 1\}$ in parallel and emits noisy firings $a_{i,t} \mid s_{i,t} \sim \text{Bernoulli}\big(p_A\,[s_{i,t}{=}0] + p_B\,[s_{i,t}{=}1]\big)$. The feature index $i \in \{1, \ldots, N\}$ simultaneously labels three objects: the orthogonal direction $\mathbf{f}_i$, the hidden state $s_{i,t}$, and the noisy emission $a_{i,t}$. The activation

$$\mathbf{x}_t \;=\; \sum_{i=1}^{N} a_{i,t}\, m_{i,t}\, \mathbf{f}_i$$

threads this index through the residual-stream construction, so the feature ↔ emission ↔ hidden-state mapping is determined entirely by the generator. This is the ground-truth correspondence the metrics evaluate against.

The Coupling bench (`temp_bench.data.toy.coupled_noisy:coupled_noisy_hmm`) keeps the same construction-time logic for the $M = 20$ emissions $a_{j,t}$ and adds a separate index $k \in \{1, \ldots, K\}$ with $K = 10$ for the hidden chain directions $\mathbf{f}^{\text{hid}}_k$. A binary coupling matrix $C \in \{0, 1\}^{M \times K}$ specifies which subset of hidden chains drives each emission via OR-gate firing, so the construction-time correspondence has two layers (chain index k, emission index j) instead of one.

B.2 Post-hoc correspondence (feature direction ↔ dictionary atom)

For an autoencoder with decoder columns $\mathbf{w}_j \in \mathbb{R}^d$, $j \in \{1, \ldots, d_{\text{sae}}\}$, the matched latent for ground-truth feature i is the column with the largest absolute cosine,

$$j^*(i) \;=\; \arg\max_j \left| \frac{\langle \mathbf{w}_j, \mathbf{f}_i \rangle}{\|\mathbf{w}_j\|\, \|\mathbf{f}_i\|} \right|.$$

The matching is computed independently for each feature, so two ground-truth features may in principle pick the same latent. We do not enforce uniqueness; this matches the wasteland-source convention in `denoising_probes.py`.

B.3 Probe definitions

Given the eval set of activations $\mathbf{x}_t$, every architecture produces a per-token latent code $\mathbf{u}(t) \in \mathbb{R}^{d_{\text{sae}}}$. For per-token architectures (TopK SAE, T-SAE, TFA-pos) the encoder runs directly on each token.

For window-level architectures (TXC, Stacked) the encoder produces one code per T-window; we broadcast that code to all T positions in the window and average over the T overlapping windows that touch each position, so every (seq, t) pair ends up with one d_{sae}-dim vector.

Single-latent correlation. For each ground-truth feature i, take the matched latent $j^*(i)$ defined above and compute Pearson correlations over all (seq, t) pairs,

$$\bar{r}_{\text{local}}(i) \;=\; \text{Corr}\big(u_{j^*(i)}(t),\; a_{i,t}\big),$$
$$\bar{r}_{\text{global}}(i) \;=\; \text{Corr}\big(u_{j^*(i)}(t),\; s_{i,t}\big),$$

then average across the N features. This is a *strict* test: the dictionary direction must align with $\mathbf{f}_i$ *and* the code dimension along that direction must track the hidden state.

Linear probe. For each feature i fit two ridge regressions ($\alpha = 1$, 80/20 train/test split) using the full code $\mathbf{u}(t)$,

$$\hat{a}_{i,t} \;=\; \mathbf{w}_{\text{loc}}^{(i)\top}\,\mathbf{u}(t) + b_{\text{loc}}^{(i)},$$
$$\hat{s}_{i,t} \;=\; \mathbf{w}_{\text{glob}}^{(i)\top}\,\mathbf{u}(t) + b_{\text{glob}}^{(i)},$$

and report $R^2_{\text{local}}(i)$ and $R^2_{\text{global}}(i)$ on the held-out 20%, averaged across the N features. This is a *looser* test: information may be diffused across many latents and the probe still recovers it.

The local-vs-global scatters in section 4 use both metrics on the same population of cells. Points above the $y = x$ diagonal denoise: the latent code tracks the hidden state better than the noisy emission.

C Sparse Probing: Full Details

Sparse probing measures whether a dictionary exposes features useful for simple classifiers of human-interpretable text concepts. Standard probing panels include both token-local properties, such as punctuation and part of speech, and broader passage-level properties, such as topic, sentiment, and named entities. This makes sparse probing a useful static test of whether temporal-window dictionaries expose concepts that per-token dictionaries fragment.

C.1 Architectures and training configuration

Six base architectures, each trained on the same `Gemma-2-2B-IT` layer-13 residual-stream activation cache (FineWeb 24K sequences $\times$ 128 tokens, fp16): a per-token TopK SAE Gao et al. [2024], the temporal SAE of Bhalla et al. [2025], the temporal feature analyzer (TFA) of Lubana et al. [2025], a multi-layer crosscoder (MLC) on layers 11–15 Lindsey et al. [2024], TXC-base ($T = 5$, $k_{\text{pos}} = 20$), and TXC-pro ($T_{\max} = 10$, $t_{\text{sample}} = 5$, $k_{\text{pos}} = 20$, eight Matryoshka groups, contrastive shifts $\Delta \in \{1, 2\}$). All cells train for $n_{\text{steps}} = 20{,}000$ at $d_{\text{SAE}} = 18{,}432$. The matched-sparsity invariant fixes the expected per-token L_0 at 20 across architectures; each arch's internal TopK budget scales with the unit it operates on (TopK SAE: $k_{\text{win}} = 20$; T-SAE BatchTopK: $20 \cdot B \cdot T$; TXC-base (T): $k_{\text{win}} = 20 \cdot T$; TXC-pro: $k_{\text{train}} = 100$, $k_{\text{inference}} = 200$; MLC ($L = 5$): $k_{\text{win}} = 100$; TFA: $20 \cdot T_{\text{tfa}}$). Within-component fairness is enforced by this matched-sparsity convention rather than by uniform per-step token throughput. We additionally train TXC-base at $T \in \{10, 20\}$ as a within-arch probe of temporal-context length.

C.2 Probing protocol

For every probe task we cache the last $S = 32$ token activations of the subject model on each prompt. Tokenisation is right-padded then per-example resliced into a 32-frame so the real tokens occupy positions $[\text{first\_real}_i,\, S - 1]$ and zero padding sits at the start; the per-example `first_real` index is stored so downstream pooling can mask the padding region. Per-token architectures encode the full (N, S, d_{in}) tensor and mean-pool over the token axis with positions $k < \text{first\_real}_i$ masked. Window architectures slide a width-T window across the S-frame at stride 1 ($n_{\text{windows}} = S - T + 1$ per prompt; with $S = 32$: $T = 5 \to 28$, $T = 10 \to 23$, $T = 20 \to 13$), encode each window to one (d_{SAE}) latent, then mean-pool over the window axis with windows whose left edge lies in

padding masked. The output is a single (d_{SAE}) feature vector per prompt for every architecture. Top-S features are selected on the train split only by $|\mu_{y=1} - \mu_{y=0}|$ (no CV / no peeking at the test set during selection), and the probe is scikit-learn `LogisticRegression(penalty='l1', solver='liblinear', C=1.0, max_iter=1000, random_state=0)` fit on the selected k_{feats} features. The per-task readout is ROC-AUC on the task's held-out test split; the headline is the unweighted mean across the SAEBench panel, averaged over 3 training seeds.

C.3 SAEBench task suite

The probing panel is the upstream SAEBench composition fixed by `chosen_classes_per_dataset` in `sae_bench/sae_bench_utils/dataset_info.py`, totalling 36 binary one-vs-rest tasks across 8 datasets:

dataset	classes	n
`LabHC/bias_in_bios_class_set1`	professions 0, 1, 2, 6, 9	5
`LabHC/bias_in_bios_class_set2`	professions 11, 13, 14, 18, 19	5
`LabHC/bias_in_bios_class_set3`	professions 20, 21, 22, 25, 26	5
`canrager/amazon_reviews_mcauley_1and5`	categories 1, 2, 3, 5, 6	5
`canrager/amazon_reviews_mcauley_1and5_sentiment`	1.0-vs-rest, 5.0-vs-rest	2
`codeparrot/github-code`	C, Python, HTML, Java, PHP	5
`fancyzhx/ag_news`	world, sports, business, scitech	4
`Helsinki-NLP/europarl`	en, fr, de, es, nl	5
SAEBench total		**36**

Two cross-token tasks (`winogrande_correct_completion` and `wsc_coreference`) round out a 38-task panel; the main-text $\overline{\text{AUC}}$ summary averages over all 38 tasks. Per-task AUCs are persisted in the leaderboard for downstream diagnostic queries.

C.4 Sweep aggregation: $\overline{\text{AUC}}$

The per-arch summary metric reported in fig. 3a compresses the k_{feats} sweep into a single number. Let $\text{AUC}_a(k)$ be the mean SAEBench AUC of arch a at $k_{\text{feats}} = k$ averaged over seeds. We define

$$\overline{\text{AUC}}(a) \;=\; \frac{1}{\log_2(k_{\max}/k_{\min})} \sum_{i=0}^{n-2} \frac{\text{AUC}_a(k_i) + \text{AUC}_a(k_{i+1})}{2} \big(\log_2 k_{i+1} - \log_2 k_i\big), \qquad (1)$$

where $\{k_i\}_{i=0}^{n-1} = \{5, 10, 20, 40, 80, 160, 320, 640\}$. The grid is geometric so $\log_2 k_{i+1} - \log_2 k_i = 1$ for every i, and eq. (1) reduces to the unweighted trapezoidal mean over $\log_2 k_{\text{feats}}$. Seed-σ on the summary metric is computed by propagating per-cell seed-σ through the trapezoidal weights.

C.5 Per-task breakdown at $k_{\text{feats}} = 160$

Figure 6 shows the architecture $\times$ task ROC-AUC matrix at the peak-AUC sparsity $k_{\text{feats}} = 160$, averaged over the 3 training seeds. Most cells in the temporal family saturate above 0.90 on the in-distribution datasets (Bias in Bios, Amazon, GitHub, AG News, Europarl), with TopK SAE consistently $\sim$0.01–0.04 behind. Two patterns are visible: TFA underperforms by $\sim$0.05–0.15 AUC on every in-distribution group, confirming that the gap reported in fig. 3a is broad rather than driven by a small subset of tasks; and the two cross-token tasks (`winogrande`, `wsc`) sit at 0.13–0.40 for every architecture, reflecting that a per-token mean-pool encoder cannot represent the long-range coreference contrast these tasks require regardless of the SAE inductive bias (section C.7).

C.6 Full AUC curve with TFA

Figure 7 shows the same per-architecture probe sweep as the right panel of fig. 3 but with TFA included on the same axes. We separate TFA into the appendix because its AUC range ($\sim$ 0.65 at $k_{\text{feats}} = 5$, climbing to $\sim$ 0.85 at $k_{\text{feats}} = 640$) compresses the y-axis on the main-text panel and obscures the $\sim$0.01–0.02 AUC ordering among the temporal family and the per-token TopK SAE baseline. TFA does not finish saturating within the $k_{\text{feats}} \leq 640$ sweep; the gap between TFA and the

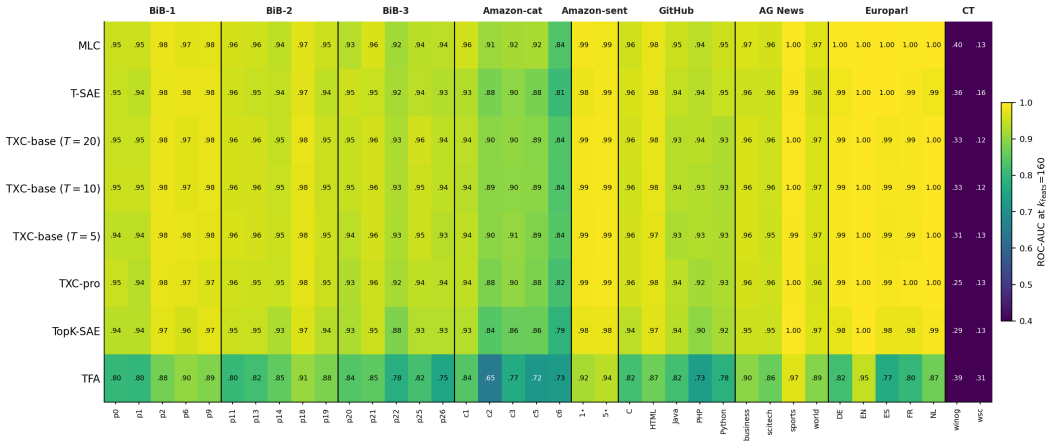


Figure 6: Per-architecture, per-task ROC-AUC at $k_{\text{feats}} = 160$ on the 38-task SAEBench panel, averaged over 3 training seeds. Rows ordered top-down by overall ranking; columns grouped by source dataset (BiB-1/2/3 = `bias_in_bios` sets, Amazon-cat = 5-class category, Amazon-sent = 1.0-vs-rest / 5.0-vs-rest, CT = the two cross-token coreference tasks). Cell labels are the AUC values with the leading "0." dropped.

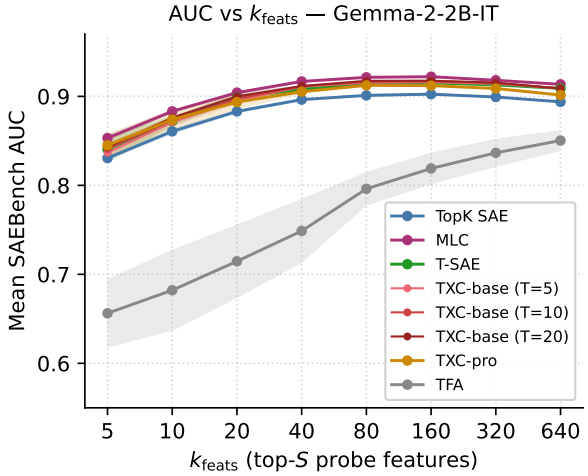


Figure 7: Full mean SAEBench AUC vs k_{feats} on `Gemma-2-2B-IT` for every architecture, including TFA. Lines are mean over 3 seeds; shaded band is $\pm 1\,\sigma$ across seeds. The temporal family + TopK SAE cluster sits at the top (saturating near $k_{\text{feats}} = 160$ at AUC ≈ 0.90–0.92); TFA underperforms by 0.05–0.18 AUC across the sweep and is still climbing at the largest k_{feats} tested.

temporal family at the largest k_{feats} is well outside the per-cell seed-σ (≤ 0.02 for TFA, ≤ 0.002 for every other architecture).

C.7 Caveats

- **Cross-token tasks.** `winogrande_correct_completion` and `wsc_coreference` require resolving a referent that lives many tokens from the variant word; a per-token mean-pool encoder cannot represent the relevant contrast, and every locked-arch cell scored 0.40–0.50 AUC on them irrespective of architecture (fig. 6). Including them shifts every $\overline{\text{AUC}}$ down by ~ 0.027 uniformly with no rank flips except TFA, which is already low.
- **TFA at paper-faithful $B = 32$.** TFA's $\overline{\text{AUC}} = 0.764$ underperforms the locked set by 0.07–0.18 AUC across the sweep; we attribute this to the small batch size required by the

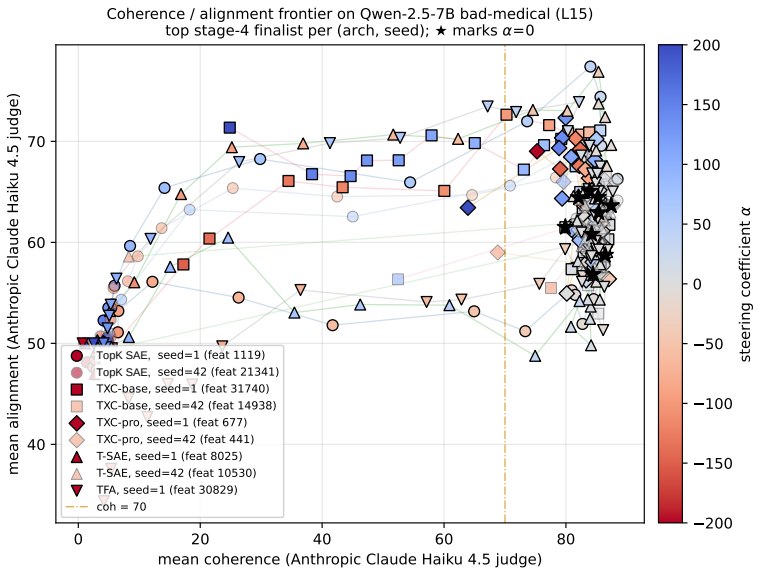


Figure 8: Coherence/alignment Pareto frontier for the `Qwen-7B` medical-misalignment cells. Each curve follows one architecture/seed across steering magnitudes; the vertical extent to the right of the $\mathrm{coh} = 70$ threshold defines the dynamic range used in fig. 5a.

(B, T, d_{SAE})-sized attention tensor at $d_{\mathrm{SAE}} = 18{,}432$. A fairer comparison at $B = 1024$ would require re-architecting the attention head and is left for follow-up.

- **Single subject model.** The headline is on `Gemma-2-2B-IT` only; replication on the BASE side and on a different subject model is left for follow-up.

D Emergent Misalignment Case Study: Full Steering Grid

The two summary subfigures in fig. 5 collapse, for each architecture, the full Wang stage-4 α frontier into a single coherence-floored alignment dynamic range (fig. 5a) and a single sparse-probe PR-AUC (fig. 5b). Figure 8 replots the same data as a coherence/alignment Pareto scatter (one curve per (architecture, seed), each curve following the top stage-4 finalist's full α frontier), making the dynamic range visible as the vertical extent of points to the right of the $\mathrm{coh} = 70$ threshold. Figure 9 shows the same data in unaggregated form: each panel is one (architecture, seed) cell on `Qwen-2.5-7B-Instruct` + bad-medical LoRA, with the mean alignment of all three Wang stage-4 finalists plotted against α, the top-finalist mean coherence overlaid as a dashed line, and the $\alpha = 0$ unsteered baseline marked in green. The shaded green band marks the $\mathrm{coh} \geq 70$ region; red and blue triangles mark the extrema used in eq. (2). The dense per-finalist α grid combines the 27-point Wang stage-4 grid (densely sampled in $|\alpha| \in [1, 10]$ with $\alpha = \pm 100$ at the tails) with the extended $|\alpha| \in \{110, 120, 130, 140, 150, 200\}$ sweep run on the top finalist of each cell. The dense extension is what reveals coherence-preserving alignment peaks at $|\alpha| > 10$ that Han's canonical grid missed (e.g. SAE-arditi seed 1, top finalist `feat 1119`, peaks at $\alpha = +40$ with align 77.4, coh > 70). All judge transcripts (per-rollout completions and Anthropic Claude Haiku 4.5 align/coh scores) are persisted in `judge_outputs.jsonl` (canonical) and `judge_outputs_extended.jsonl` (extension) for post-deadline κ validation.

Alignment dynamic range. For each architecture/seed cell c, let $\mathcal{A}_c$ be the set of steering magnitudes whose judged coherence is at least 70. We summarize steering control by the coherence-floored alignment range

$$\Delta\mathrm{align}(c) = \max_{\alpha \in \mathcal{A}_c} \mathrm{align}(c, \alpha) - \min_{\alpha \in \mathcal{A}_c} \mathrm{align}(c, \alpha), \qquad \mathcal{A}_c = \{\alpha : \mathrm{coh}(c, \alpha) \geq 70\}. \tag{2}$$

This is the statistic plotted in fig. 5a.

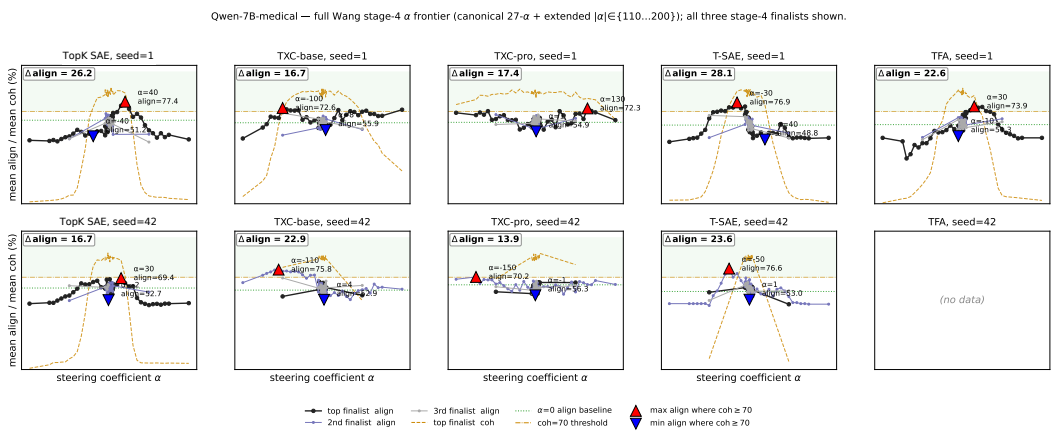


Figure 9: Full steering sweep for the `Qwen-7B` medical-misalignment cells. Panels show alignment across steering magnitudes, with coherence overlaid for the top finalist; extrema inside the coh ≥ 70 region determine the summarized dynamic range.

D.1 Decoder-norm calibration ($1/\sqrt{T}$)

When a TXC reconstructs T positions from a single shared latent, each position-specific decoder row carries only a fraction of the per-position reconstructive mass that a per-token SAE decoder row carries; under the BatchTopK normalisation convention used in section 3 this leaves each TXC decoder row at ℓ_2 norm $\approx 1/\sqrt{T}$ relative to the unit-norm convention of a per-token SAE decoder row. A magnitude α applied to a TXC mined direction therefore produces an effective residual-stream perturbation that is a factor of $1/\sqrt{T}$ smaller than the same α applied to a per-token SAE direction. To make the cross-architecture α-grid comparable, we rescale the TXC's grid by $\sqrt{T}$, so that the same α corresponds to the same ℓ_2-norm intervention across architectures.

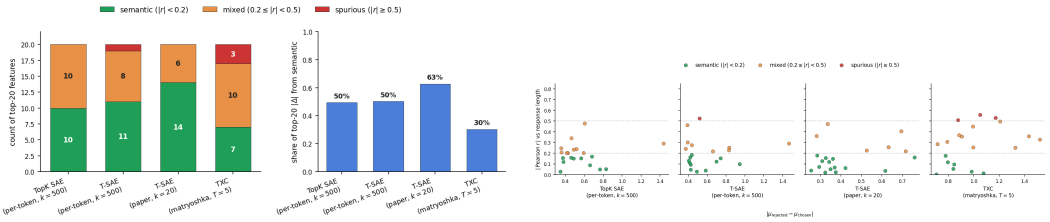
We confirm this empirically across the canonical c6 cells: the mean ℓ_2-norm of mined TXC-base ($T = 5$) decoder rows is 0.443, against the predicted $1/\sqrt{5} = 0.4472$; for TXC-pro ($T = 10$) the mean is 0.310, against $1/\sqrt{10} = 0.3162$. Both match the prediction within $\sim 1.5\%$, justifying the $\sqrt{T}$ rescaling without per-cell calibration.

E HH-RLHF Preference Decomposition: Full Details

Figure 10 collapses the per-feature decomposition for each architecture into a top-20 tier-count panel and a semantic-mass-share panel. This appendix supplies the methodology behind that summary and lists the top-5 autointerpreted features per architecture (table 1). The qualitative reading of the architecture comparison is concrete in that table: T-SAE at the paper-faithful $k = 20$ surfaces alignment-relevant labels (*clarification questions and requests for explanation*, *expressions of limitation or inability in responses*, *AI capability limitations or disclaimers*), while TXC at $T = 5$, $k_{\text{win}} = 500$ surfaces sentence-level discourse markers (*conversational filler or discourse markers*, *conversational responses and speech patterns*) whose length correlations are exactly the spurious channel the paper-faithful contrastive-loss training is tuned to suppress.

E.1 Response masking and ranking metric

For each `(chosen, rejected)` pair, we tokenise the two strings independently with `max_length`$=$ 256 and identify the differing assistant response by a character-level longest common prefix between the two raw strings; we then map character offsets back to token indices via the tokenizer's offset map to produce a per-side `response_mask` that selects only the differing-response tokens. Layer-12 residual-stream activations are captured under this mask, and per-feature means are taken across all valid response-token positions in the 1000 pairs. Features are ranked by the signed metric $\mu_{\text{rejected}} - \mu_{\text{chosen}}$, matching Bhalla et al. [2025, §4.5]; negative values flag features that activate more strongly on *chosen* responses, positive values on rejected. Tier classification thresholds $|r| < 0.2$



(a) Top-20 feature classification and semantic-mass share.

(b) Preference-separation magnitude versus response-length correlation.

Figure 10: HH-RLHF preference decomposition for `Gemma-2-2B` layer 12. Features are ranked by $\mu_{\text{rejected}} - \mu_{\text{chosen}}$ and classified by correlation with response-length difference.

(semantic) and $|r| \geq 0.5$ (length-spurious) follow the paper. Our absolute response-length means (rejected 36.23, chosen 28.57 tokens) are lower than the paper's reported 49.24 and 37.84 because of tokenizer and truncation differences across HH-RLHF dataset versions; the paired-t-test signal direction and significance ($p = 9.76 \times 10^{-10}$) match the paper's $p \approx 9 \times 10^{-10}$ to leading digit.

For each feature j, define its preference separation

$$\Delta\mu_j = \mu_{\text{rejected},j} - \mu_{\text{chosen},j},$$

where means are taken over response-token positions only. We also compute the length correlation

$$r_j = \text{Corr}_{\text{pairs}}\big(\text{act}_{\text{rejected},j} - \text{act}_{\text{chosen},j}, \ell_{\text{rejected}} - \ell_{\text{chosen}}\big).$$

Features are classified as semantic if $|r_j| < 0.2$, mixed if $0.2 \leq |r_j| < 0.5$, and length-spurious if $|r_j| \geq 0.5$. For a top-K set $\mathcal{T}_K$, the semantic-mass share is

$$\text{SemMass}(\mathcal{T}_K) = \frac{\sum_{j \in \mathcal{T}_K : |r_j| < 0.2} |\Delta\mu_j|}{\sum_{j \in \mathcal{T}_K} |\Delta\mu_j|}.$$

E.2 Autointerpretation setup

For each top-K feature per architecture, we identify the 5 max-activating positions across the cohort (restricted to response tokens, so the label captures the chosen-vs-rejected differential signal rather than generic prompt activations), decode a symmetric token window around each firing position with the firing token marked by `**double asterisks**`, and send the 5 contexts in a single Bills-et-al-style **?** prompt to Claude Haiku 4.5 (`claude-haiku-4-5-20251001`, max output 80 tokens, default sampling). The prompt asks for a single short concept label (3–12 words), no preamble, no markdown. We use Haiku rather than Sonnet because the task is descriptive labeling rather than judgment; the lower cost lets us label the full top-50 per architecture for $\sim$\$0.25 total. All call records, contexts, and labels are persisted to the per-architecture `top_features.json` files referenced above.

F Backtracking Case Study: Full Details

This appendix gives implementation details for section 5.2.

F.1 Architectures and training configuration

We compare six architectures trained on the same Llama-3.1-8B residual-stream activation cache at layer 10, with $d_{\text{SAE}} = 32{,}768$, $n_{\text{steps}} = 300{,}000$, and batch size 1024: a per-token TopK SAE Gao et al. [2024], the temporal SAE of Bhalla et al. [2025], the temporal feature analyzer (TFA) of Lubana et al. [2025], a multi-layer crosscoder (MLC) Lindsey et al. [2024], TXC-base ($T = 5$, $k_{\text{pos}} = 20$), and TXC-pro ($T_{\max} = 10$, $t_{\text{sample}} = 5$, $k_{\text{pos}} = 20$, eight Matryoshka groups, and contrastive shifts $\Delta \in \{1, 2\}$). We also train TXC-base and TXC-pro at batch size 256; these cells are reported in section F.13.

rank	r	$\mu_{\text{rejected}} - \mu_{\text{chosen}}$	autointerp label
TopK SAE (per-token, $k = 500$)			
0	$+0.29$	$+1.439$	objects or entities being directly affected by actions
1	-0.05	$+0.830$	physical objects or locations used in scenarios
2	-0.05	-0.763	information or data items that are absent or unavailable
3	-0.17	$+0.680$	illegal drugs and controlled substances
4	$+0.09$	$+0.660$	direct objects of transitive verbs or noun phrases
T-SAE (per-token, $k = 500$)			
0	$+0.29$	$+1.468$	pronouns and discourse connectives in explanatory contexts
1	-0.10	-0.944	negation or negative polarity markers
2	-0.25	-0.833	seeking confirmation or agreement on statements
3	-0.23	-0.828	hedging language and softening expressions
4	-0.15	-0.743	uncertainty markers in clarification requests
T-SAE (paper-faithful, $k = 20$)			
0	-0.16	-0.759	clarification questions and requests for explanation
1	-0.22	-0.716	clarification requests and confused responses
2	-0.41	-0.695	expressions of limitation or inability in responses
3	$+0.26$	$+0.633$	verb forms in instructional or procedural contexts
4	-0.04	-0.617	AI capability limitations or disclaimers
TXC (matryoshka, $T = 5$, $k_{\text{win}} = 500$)			
0	$+0.33$	$+1.519$	nouns denoting objects or entities being modified
1	-0.36	-1.433	discourse markers indicating uncertainty or topic shifts
2	-0.25	-1.327	clarification requests or seeking more information
3	-0.50	-1.207	conversational filler or discourse markers
4	-0.53	-1.172	conversational responses and speech patterns

Table 1: Top-5 features per architecture, ranked by $|\mu_{\text{rejected}} - \mu_{\text{chosen}}|$ on the 1000-pair `harmless-base` cohort. Sign of the diff indicates the side the feature activates more strongly on (negative $\rightarrow$ chosen, positive $\rightarrow$ rejected). Length-Pearson r classifies each feature as semantic ($|r| < 0.2$), mixed, or length-spurious ($|r| \geq 0.5$). Autointerp labels are produced by Claude Haiku 4.5 (section E.2).

F.2 Reproduction setup

We follow the dataset recipe of Venhoff et al. [2025], as reproduced in Ward et al. [2025, Appendix C]. The procedure has three steps: generate 300 math prompts with Claude Sonnet 3.7, sample reasoning traces from DeepSeek-R1-Distill-Llama-8B, and label each sentence with an LLM judge. The prompts cover ten categories, with 30 prompts per category: `basic_logic`, `geometry`, `probability`, `arithmetic`, `counting`, `number_theory`, `set_theory`, `sequences`, `inequalities`, and `algebra_word_problems`. We use 280 prompts for the `dom` split and hold out 20 for `eval`. Following our inducement-evaluation pipeline, we use Claude Sonnet 4.6 rather than GPT-4o for sentence labels. Labels are binary, with field `is_backtracking`.

The steering vector is the difference-of-means direction for `is_backtracking`, computed from base-model residual activations at layer 10. Activations are taken from token offsets -13 through -8 relative to the first token of each labeled backtracking sentence. We use the same layer, offset window, and base-model activation convention as Ward et al. [2025, Sec. 3.1, Appendix B.1].

F.3 Cohort construction

The evaluation cohort contains 61 MATH-500 questions Hendrycks et al. [2021]. It has two strata: 31 questions where the unsteered model gives a parsable but incorrect boxed answer, and 30 questions where it gives a parsable correct boxed answer. We drop questions with no boxed answer within the token budget, since both the inducement and answer-change metrics require a parsable continuation. Questions are drawn deterministically from a locked `flip_matrix.parquet` file, so all architectures are evaluated on the same questions in the same order. Question identifiers use the MATH-500 `unique_id` format, for example `test/algebra/1184.json`.

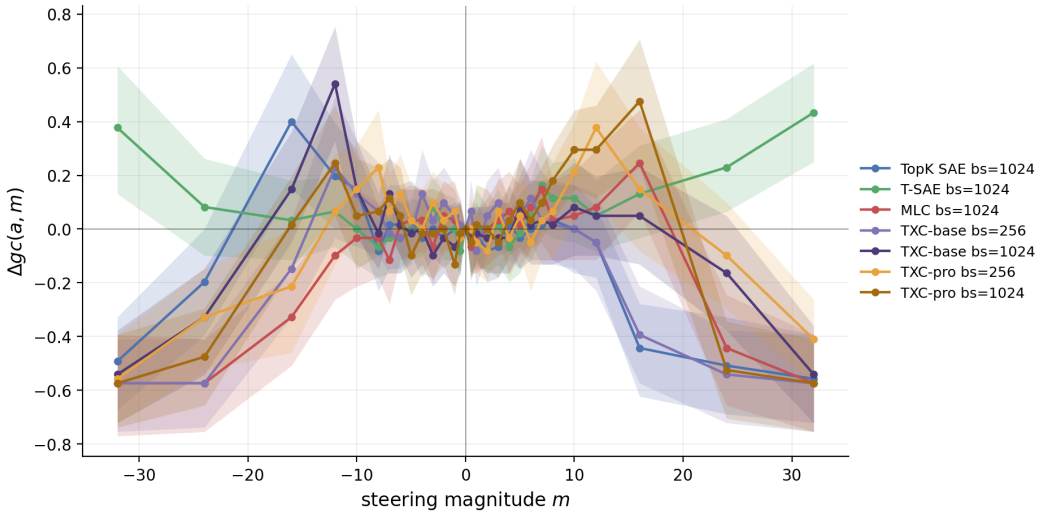


Figure 11: Per-architecture inducement curves $\Delta gc(a, m)$. Curves are per-question baseline-corrected at $m = 0$; shaded bands are bootstrap 95% CIs over the 61 cohort questions.

F.4 Magnitude grid

We evaluate the 25-point grid

$$\mathcal{M} = \{0, \pm 0.5, \pm 1, \pm 2, \pm 3, \pm 4, \pm 5, \pm 6, \pm 7, \pm 8, \pm 10, \pm 12, \pm 16\}.$$

The grid is denser between ± 0.5 and ± 8, where the inducement curves change most quickly, and coarser at larger magnitudes. The $m = 0$ point is the no-intervention baseline used in section F.5. Figure 11 plots the per-architecture $\Delta gc(a, m)$ curves on this grid plus the extended follow-up at $m \in \{\pm 24, \pm 32\}$.

F.5 Inducement metric

For each architecture a, question q, and steering magnitude m, let $gc(a, q, m)$ be the Sonnet judge count of genuine backtracking events in the continuation. The main inducement statistic subtracts the shared unsteered cut-and-continue floor questionwise:

$$\Delta gc(a, m) = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \big(gc(a, q, m) - gc(a, q, 0) \big),$$

where $\mathcal{Q}$ is the locked 61-question MATH-500 cohort. Positive Δgc means steering induces more genuine backtracking than the unsteered continuation protocol on the same questions.

F.6 cut25 continuation protocol

For each question q, let t_q^{full} be the token sequence of the unsteered reproduction. The prefix is the first $\lfloor 0.25 |t_q^{\text{full}}| \rfloor$ tokens. Conditioned on this prefix, the reasoning model samples a new continuation at temperature 1.0 with a 1024-token budget. During this continuation, the steering hook adds $m v_{\text{feat}}$ to the layer-10 residual stream at each generated position. Here v_{feat} is the architecture-specific mined direction, rescaled to match the ℓ_2 norm of the dom-base-union vector, and $m \in \mathcal{M}$.

We use cut25 because a fixed proportional cut gives comparable continuations across questions of different lengths. Full-trace continuation gives little separation between architectures, since early tokens often restate the problem. Cutting at an LLM-judged sentence boundary adds a second judge-dependent step.

F.7 Sonnet judge prompt and reliability

For each $(\text{architecture}, \text{question}, \text{magnitude})$ tuple, the inducement metric uses a judge count $gc(a, q, m)$ of genuine backtracking events in the sampled continuation. The rubric counts cases

Unsteered (m=0) **no boxed ×**

…*so* $2.4/0.2 = 12$. ***Wait, let me double-check my steps.*** $2.5^2 = 6.25$, $0.7^2 = 0.49$, $6.25 - 0.49 = 5.76$… ***Let me just verify each step once more****… [continues looping]*

TXC-pro (bs=256), $m = +12$ **ans = 12 ✓**

… the entire fraction is $2.4/0.2 = 12$. ***Wait, let me double-check that.*** $0.2 \times 12 = 2.4$. ***So, yes, that's correct.*** … $\boxed{12}$

Figure 12: Qualitative backtracking example on `test/prealgebra/951.json`. The unsteered continuation derives the right value but loops without committing; TXC-pro steering produces one decisive verification step and emits the correct boxed answer.

where the model identifies a problem in its reasoning and changes course, such as correcting arithmetic, noticing a missing constraint, abandoning an approach, or revising a wrong assumption. It does not count filler uses of "wait," restatements, repeated text, loops, or degeneration.

The judge is Claude Sonnet 4.6 (`claude-sonnet-4-6`, max output 200 tokens, default sampling temperature). The prompt is:

```
You are evaluating a math problem-solving generation for genuine
backtracking behavior.
Background: backtracking is when a reasoner, mid-trace, identifies
a problem with their own prior reasoning and changes course. A
reasoner who emits the word ‘‘wait’’ or ‘‘hmm’’ as conversational filler
is NOT backtracking. A reasoner who says ‘‘wait, no, actually...’’
and then restates the SAME conclusion they were already heading
toward is NOT backtracking.
Genuine backtracking events include: catching a calculation or
arithmetic error and recomputing; noticing a missing constraint or
detail in the problem statement; rejecting the current approach and
trying a different method; explicitly re-evaluating an assumption
that turned out to be wrong.
NOT genuine (do NOT count these): conversational filler (‘‘Hmm, let
me think,’’ ‘‘Hmm, okay’’); restating the problem without finding
an error; re-stating the same conclusion with different wording;
pseudo-backtracking where ‘‘wait’’ is followed by repeating the same
content; looped or repetitive emissions (e.g., ‘‘Wait, I’m not.
Wait, I’m not.’’); gibberish, single-token loops, or non-English
degeneration.
Problem prompt the model was solving: {prompt_text}
Model’s generation: ‘‘{generation}’’
Count the number of GENUINE backtracking events in this generation.
Reply with EXACTLY this format on two lines:
COUNT: <integer>
NOTES: <one short sentence explaining your count>
```

Each continuation produces one judge call. We persist all calls to a per-cell `judge_outputs.jsonl` file with `transcript_id`, `magnitude`, `architecture`, `seed`, `judge_id`, `prompt_hash`, `label`, `raw_response`, and `timestamp`. This makes later inter-rater checks possible without re-running the model generations. Prior internal checks on a 20-transcript blind human-annotated sample gave $\kappa = 0.749$ for sentence-level coherence, $\kappa = 0.773$ for genuine-backtracking detection, and $\kappa = 1.000$ for looping detection, with raw agreements 0.85, 0.95, and 1.00.

F.8 Token-count budget and convergence probe

Each main cell trains for 300,000 steps at batch size 1024. With window length $T = 5$, this is about 1.5B token-positions. This is within the training-token range used by TFA Lubana et al. [2025], T-SAE Bhalla et al. [2025], and GemmaScope Lieberum et al. [2024], and is about 15× larger than the 20,000-step sprint runs used during protocol development.

We log held-out NMSE, ℓ_0 density, and dead-feature count every 100 steps on a separate seed stream. These probe batches do not overlap with training minibatches. Figure 13 shows the resulting curves.

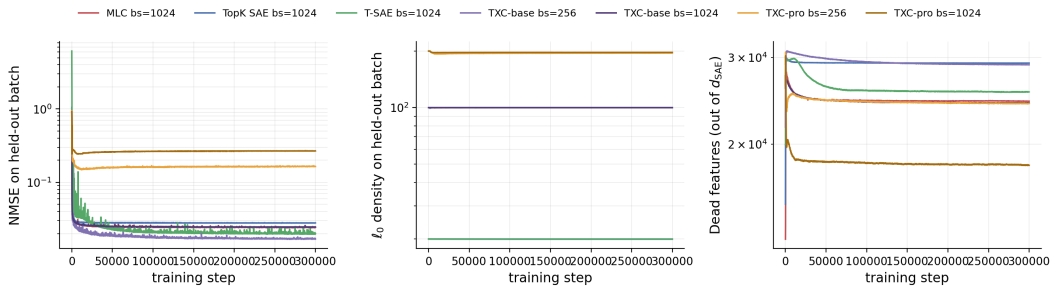


Figure 13: Held-out probe metrics over training for each cell at seed 42: NMSE, ℓ_0 density, and dead-feature count out of $d_{\text{SAE}} = 32{,}768$. The y-axes are log-scaled.

	PR-AUC						ROC-AUC					
Architecture	$S=1$	$S=2$	$S=4$	$S=8$	$S=16$	$S=32$	$S=1$	$S=2$	$S=4$	$S=8$	$S=16$	$S=32$
TopK SAE (bs=1024)	0.130	0.132	0.137	0.175	0.196	0.229	0.507	0.508	0.518	0.566	0.589	0.626
T-SAE (bs=1024)	0.158	0.164	0.169	0.196	0.213	0.245	0.591	0.603	0.617	0.640	0.655	0.683
MLC (bs=1024)	0.128	0.147	0.165	0.176	0.213	0.242	0.509	0.561	0.582	0.599	0.648	0.682
TXC-base (bs=256)	0.166	0.167	0.182	0.226	0.254	0.269	0.604	0.604	0.625	0.666	0.685	0.696
TXC-base (bs=1024)	0.143	0.168	0.177	0.201	0.217	0.250	0.536	0.593	0.609	0.628	0.647	0.679
TXC-pro (bs=256)	0.161	0.181	0.231	0.242	0.258	0.266	0.601	0.635	0.681	0.688	0.697	0.708
TXC-pro (bs=1024)	0.162	0.168	0.176	0.222	0.242	0.267	0.595	0.604	0.618	0.665	0.681	0.707

Table 2: Sparse-probe PR-AUC and ROC-AUC for backtracking-sentence detection. We use five GroupKFold splits by question and ℓ_1-regularized logistic regression with $C = 1$. Entries are fold means. PR-AUC is the main metric; chance is the positive-class prior, approximately 0.12.

By step 200,000, NMSE has plateaued for all cells. The relative change from step 200K to step 300K is below 2.5% in every cell and below 0.5% in five of seven cells. The ℓ_0 density reaches its target within the first 1K steps and remains fixed thereafter. Dead-feature counts continue to decrease slowly, with typical changes over the final 100K steps below 1% of d_{SAE}. Thus the reconstruction metric is not sensitive to extending the run past 300K steps, although longer runs may revive additional low-use features.

F.9 PR-AUC methodology and full table

Following Kantamneni et al. [2025], we test whether each architecture supports a sparse linear probe for the sentence-level label "does this sentence begin a backtracking event?" Labels are taken from the original reproduction traces. We capture base-model residual activations at layer 10 using a six-token window with offsets -13 through -8 relative to the first token of each sentence. Architectures with smaller windows use the trailing T_{arch} positions of this window. Architectures with $T_{\text{arch}} > 6$ are excluded from this protocol. We encode each $(\text{sentence}, T \times d_{\text{in}})$ tensor and max-pool absolute feature activations over positions to obtain one d_{SAE} vector per sentence.

Feature selection is done inside each fold. On the train split only, we compute $|\mu_{\text{pos}} - \mu_{\text{neg}}|$ for every feature and keep the top S. The probe is ℓ_1-regularized logistic regression on those selected features, with $C = 1$, scikit-learn's `liblinear` solver, `max_iter`=2000, and random seed 42. We report the mean over five GroupKFold splits by question.

We use PR-AUC as the main detection metric because the positive class rate is about 12%. ROC-AUC is also reported for comparison with earlier exploratory runs.

F.10 Steering variants

The main inducement results in fig. 4a use a uniform per-position write: at every generated position, we add mv_{feat} to the layer-10 residual stream, where v_{feat} is the mined feature's mean decoder direction

Architecture	bs	m_a^*	rescues@m^*	regr.@m^*	rescues@0	regr.@0	Δ_{net}@m^*	Δ_{net}@0	$\mathbf{\Delta}_{\text{net}}^{\text{corr}}$
TXC-base	256	-12	1	16	0	7	-15	-7	$\mathbf{-8}$
TXC-base	1024	-12	1	15	0	7	-14	-7	$\mathbf{-7}$
TXC-pro	256	$+12$	6	7	0	7	-1	-7	$\mathbf{+6}$
TXC-pro	1024	$+16$	0	18	0	7	-18	-7	$\mathbf{-11}$
TopK SAE	1024	-16	1	14	0	7	-13	-7	$\mathbf{-6}$
T-SAE	1024	$+7$	3	7	0	7	-4	-7	$\mathbf{+3}$
MLC	1024	$+16$	0	13	0	7	-13	-7	$\mathbf{-6}$

Table 3: Rescue and regression counts at each cell's optimal magnitude m_a^* and at $m = 0$. A rescue is a question where the unsteered cut-and-continue baseline is wrong but the steered continuation is correct. A regression is the reverse. The corrected statistic subtracts the $m = 0$ cut-and-continue noise floor. Cohort size $n = 61$.

across the window. We call this V0. Pilot variants included position-cycled writes over decoder slabs (V1), trailing-window broadcast writes (V2), and an encoder-pre-image least-squares write (V4). We use V0 here to avoid coupling the evaluation protocol to a specific temporal architecture.

F.11 Magnitude-axis convention

The sign of a mined feature direction is arbitrary. We therefore report curves using the raw mined sign for each architecture rather than flipping signs so that all peaks occur at $m > 0$.

The raw-sign peak magnitudes are: TopK SAE at $m = -16$, T-SAE at $m = +7$, MLC at $m = +16$, TXC-base at $m = -12$ for both batch sizes, TXC-pro at $m = +12$ for $bs = 256$, and TXC-pro at $m = +16$ for $bs = 1024$. The two TXC-base cells match in both sign and magnitude. The two cells that peak at the grid edge, MLC and TXC-pro $bs = 1024$, were also evaluated at $m \in \{\pm 24, \pm 32\}$; in both cases Δgc drops beyond $|m| = 16$. Thus the observed peaks are not artifacts of truncating the grid at 16.

F.12 Multi-seed replication

All main numbers in fig. 4 and table 2 use training seed 42. Multi-seed replication is left for camera-ready or follow-up work. Since all judge outputs are persisted, new seeds require new model generations and judge calls only for those new transcripts.

F.13 TXC at batch size 256

TXC-base and TXC-pro are also trained at batch size 256 for 300,000 steps. This isolates the batch-size change from the architecture change.

Peak Δgc values are 0.230 for TXC-base $bs = 256$, 0.541 for TXC-base $bs = 1024$, 0.377 for TXC-pro $bs = 256$, and 0.475 for TXC-pro $bs = 1024$. The corresponding raw-sign peak magnitudes are -12, -12, $+12$, and $+16$. At $S = 8$, PR-AUC values are 0.226, 0.201, 0.242, and 0.222.

The TXC-base peak magnitude is unchanged across batch sizes, while peak Δgc increases from 0.230 to 0.541. For TXC-pro, the peak moves from $+12$ to $+16$ and the lift increases from 0.377 to 0.475. Detection behaves differently: the smaller batch gives higher $S = 8$ PR-AUC for both TXC-base and TXC-pro. These results do not show a single monotone effect of batch size across metrics.

F.14 Optimal-magnitude rescue analysis

For each architecture a, we evaluate two magnitudes: $m = 0$ and $m = m_a^*$, where m_a^* is the peak-Δgc magnitude from the canonical grid. At each magnitude, we save generations and run two Sonnet 4.6 judge passes: one for genuine-backtracking count and one for coherence on a 0 to 3 scale. We define `coherent` as coherence grade at least 2, and `backtracking` as judge count at least 1.

Architecture	bs	m_a^*	coh+bt	coh+no-bt	inc+bt	inc+no-bt	missing	n
TXC-base	256	-12	22	17	17	5	0	61
TXC-base	1024	-12	30	17	11	3	0	61
TXC-pro	256	$+12$	28	28	3	2	0	61
TXC-pro	1024	$+16$	10	1	44	6	0	61
TopK SAE	1024	-16	26	20	11	3	1	61
T-SAE	1024	$+7$	29	31	0	1	0	61
MLC	1024	$+16$	28	24	6	3	0	61

Table 4: Coherence and backtracking contingency at each cell's optimal magnitude m_a^*. Coherent means Sonnet 4.6 coherence grade at least 2 on a 0 to 3 rubric. Backtracking means judge count at least 1.

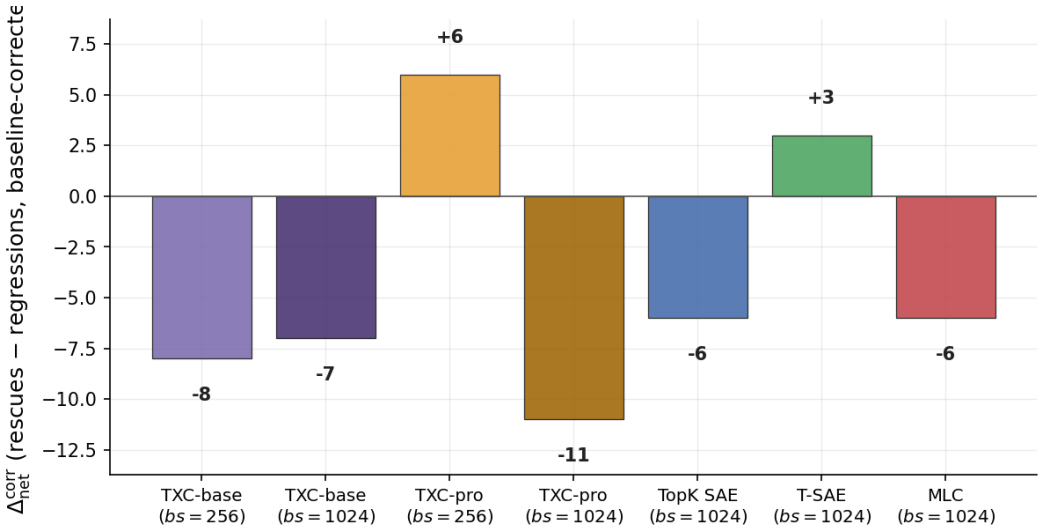


Figure 14: Baseline-corrected net rescue lift $\Delta_{\text{net}}^{\text{corr}}$ from table 3. Positive values indicate more rescues than regressions after subtracting the $m = 0$ cut-and-continue baseline.

F.15 Reference numbers from prior architectures

For context, we also record two prior exploratory TXC variants that are not part of the locked architecture set. A TXC variant with $k = 16$ active features per position and $T = 6$ reaches peak $\Delta gc = +1.574$ at $m = -12$ on the same cohort and protocol. A Matryoshka-8 TXC variant with $k = 16$ per position reaches peak $\Delta gc = +0.492$ at $m = +12$. These variants differ in sparsity and position resolution, so we treat them as context rather than direct comparisons.

G Qualitative Analysis of Large Model Features

This appendix gives a qualitative view of what TXC features look like in a trained language model. We analyze a $T = 5$ TXC trained on the `gemma-2-2b-it` block 13 post-residual stream. Because TXC uses full-rank cross-position encoder and decoder weights, a single feature can aggregate information from all T positions and contribute reconstructive mass to any position in the window. We therefore ask whether the learned features look token-local, as in a conventional SAE, or whether they often track multi-token spans.

We autointerpret active TXC features by running the trained dictionary over 1,500 32-token chains from FineWeb, retaining every feature with at least three top-window examples. This yields 5,033 retained features. For each feature, we elicit a single-sentence concept explanation with Claude Haiku 4.5, embed explanations with `sentence-transformers/all-MiniLM-L6-v2`, project them

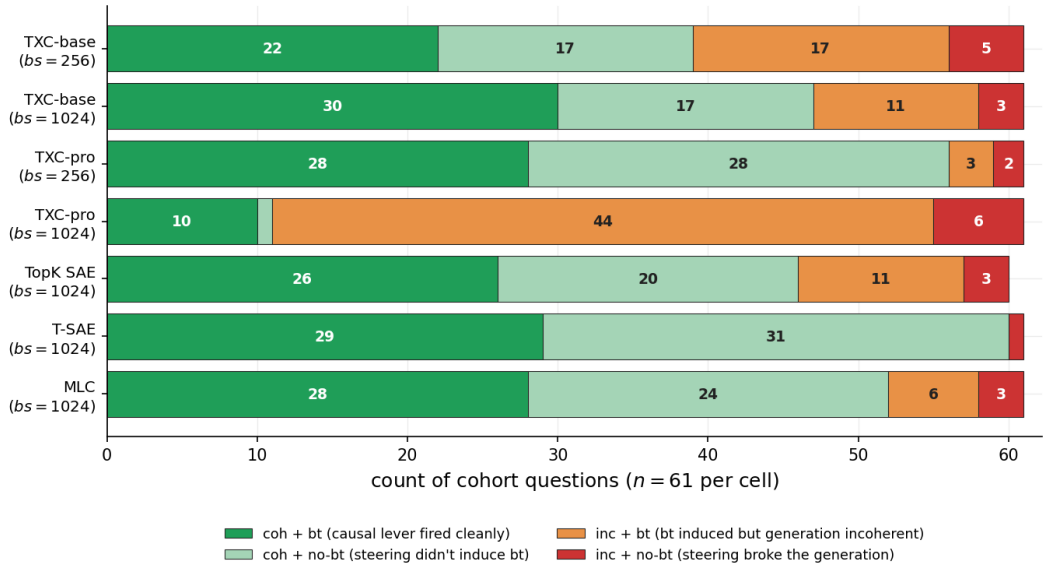


Figure 15: Coherence and backtracking contingency from table 4. The `coh+bt` segment counts coherent generations with induced backtracking. The `inc+bt` segment counts generations where backtracking is detected but the surrounding continuation is incoherent.

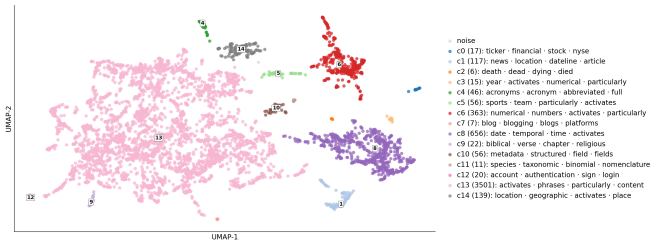


Figure 16: **UMAP of TXC feature explanations** for a `gemma-2-2b-it` block 13 TXC. Autointerpreted features form weakly separated clusters spanning both entity-like and discourse-level concepts.

to 2D with UMAP, and cluster them with HDBSCAN. We also visualize one 32-token chain by selecting one feature per position using an exclusive score that rewards features whose activation mass is concentrated at that token.

Figure 16 shows that the explanation embedding contains weakly separated clusters spanning both entity-like and discourse-level concepts. The cluster inventory includes concrete entity types such as tickers, acronyms, dates, and sports headlines, as well as discourse-level abstractions such as news article datelines, blogging platforms, biblical chapter/verse references, and taxonomic binomial nomenclature. The global silhouette score is low ($+0.01$), so we treat these clusters as a qualitative inventory rather than sharply separated semantic islands.

Figure 17 shows that selected TXC features often appear as $\geq T$-token diagonal bands, tracking short concept spans rather than isolated tokens. Together, the UMAP inventory and per-sentence activation map suggest that the window-shared parameterization surfaces features whose support is distributed across multiple positions.

The rest of this appendix gives the precise definitions behind these plots: the top-window surfacing procedure, the explanation prompt, the MiniLM $\rightarrow$ UMAP $\rightarrow$ HDBSCAN pipeline, the cluster-quality metrics, and the exclusive-score selection rule.

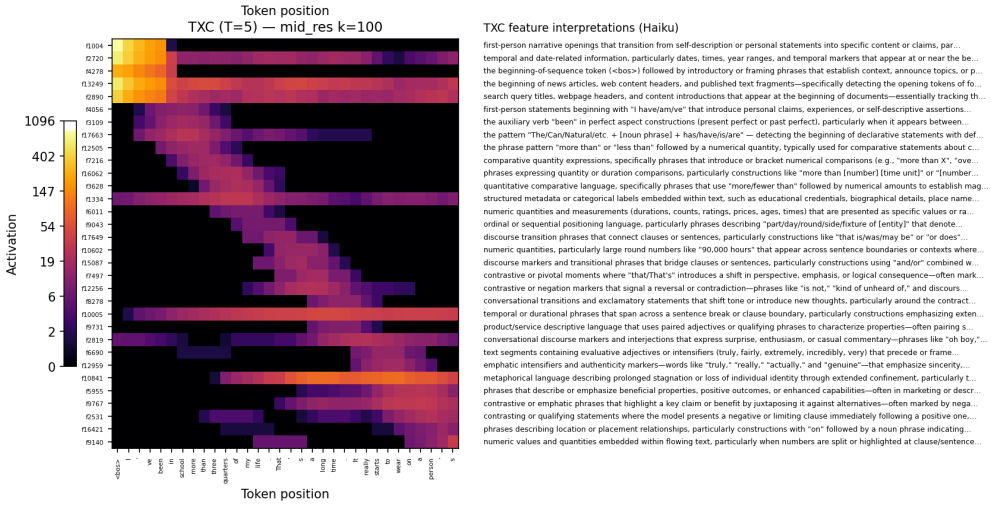


Figure 17: **Per-token TXC activations on a single 32-token chain.** Each row is the feature that activates most exclusively at one position. Diagonal bands show that TXC features often claim coherent multi-token spans rather than isolated tokens.

G.1 Feature surfacing: top-k activating windows

The TXC of section 3 is trained for 3,000 steps at batch 1,024 with Adam (η=3×10^{-4}) on a cached `gemma-2-2b-it` block-13 post-residual stream, and reaches a final FVU of 0.085. We then run the trained dictionary over a uniformly sampled N_{chains}=1,500 subset of the activation cache (each chain is a fixed-length token sequence) and, for every feature index $j \in \{1, \ldots, d_{\text{sae}}\}$, retain a min-heap of the N_{ex}=12 window starts with the largest pre-TopK activation $u_j(c, s)$, where c is a chain index and $s \in \{0, \ldots, L-T\}$ a window start within that chain. After the scan,

$$\mathcal{W}_j \;=\; \text{argtop}_{N_{\text{ex}}} \left\{ u_j(c,s) \;\middle|\; c \in \text{Sample}(N_{\text{chains}}),\; s = 0, \ldots, L-T \right\}. \tag{3}$$

A feature j is retained for autointerp iff $|\mathcal{W}_j| \geq N_{\min} = 3$, yielding $|\mathcal{F}| = 5{,}033$ retained features. Each retained window is decoded back to text via the Gemma tokenizer (special tokens such as `<bos>`, `<start_of_turn>`, `<end_of_turn>` rendered literally) and the activating span is wrapped in `[FOCUS]` ... `[/FOCUS]` markers with C=10 tokens of context on either side.

G.2 Single-feature explanation prompt

For each surfaced feature j, the $N_{\text{ex}} = 12$ activating windows are formatted as a single user message and sent to `claude-haiku-4-5-20251001` [Anthropic, 2025] with concurrency 1 and SDK retry-after backoff. The system prompt (verbatim):

```
You are a meticulous AI researcher analyzing patterns in neural
network features.  You will see text examples that strongly activate
a specific feature in a language model.  Identify the concept,
pattern, or topic this feature represents.
The activating window in each example is wrapped in
[FOCUS]...[/FOCUS] tags.  These tags are inserted by us to mark the
span -- they are NOT part of the underlying text.  Do NOT describe
the tags themselves, the brackets, or the word 'focus'; describe
only the actual tokens between the tags and how they relate to the
surrounding context.
Guidelines:  - Focus on the COMMON pattern across examples, not
one-off details.  - 1-2 short sentences only.  - Describe in
linguistic / semantic terms, not numerics.  - Never mention [FOCUS],
[/FOCUS], '>>', '<<', or 'markers' in your output.
After your explanation, on a new line emit a SAFETY tag:  [SAFETY]:
REFUSAL | DECEPTION | HARMFUL_CONTENT | BIAS | NONE ...
Format strictly:  [EXPLANATION]: <one sentence> [SAFETY]: <tag>
```

The user message lists the N_{ex} examples ordered by activation, each prefixed with `Example` i `(act=`a_i`):` where $a_i = u_j(c_i, s_i)/\max_{i'} u_j(c_{i'}, s_{i'})$, truncated at 280 characters. We parse the `[EXPLANATION]` line into explanation$_j$; empty parses fall back to the first 240 characters of the raw output.

G.3 Cluster discovery: MiniLM → UMAP → HDBSCAN

Let $e_j \in \mathbb{R}^{384}$ be the ℓ_2-normalized embedding of explanation$_j$ under `sentence-transformers/all-MiniLM-L6-v2`.

UMAP. 2D projection $\phi_j = \text{UMAP}(e_j) \in \mathbb{R}^2$ with hyperparameters $n_{\text{neighbors}}$=10, $\min_{\text{dist}}$ =0.1, metric = cosine, seed = 0.

HDBSCAN. Cluster labels $\ell_j = \text{HDBSCAN}(\{\phi_j\}_{j\in\mathcal{F}})$ with $\min_{\text{cluster\_size}}$ =4, $\min_{\text{samples}}$ =2, cluster-selection $\epsilon = 0.4$. Points with $\ell_j = -1$ are reported as noise. The pipeline yields k=15 clusters and a noise fraction below 2%, summarised in fig. 16.

G.4 Lexical cluster labeling

Cluster labels in the legend of fig. 16 are produced by a cheap TF-IDF-style score over content tokens. Let $\text{toks}(s)$ extract lower-cased alphabetic tokens of length ≥ 4 from explanation s after removing a small stop-list (determiners, function words, autointerp boilerplate such as `feature`, `features`, `describes`, `represent`, `relates`). For each cluster c and token w:

$$\text{tf}_c(w) = \sum_{j:\ell_j=c} \mathbb{1}\big[w \in \text{toks}(\text{explanation}_j)\big], \qquad \text{df}(w) = \sum_c \mathbb{1}\big[\text{tf}_c(w) > 0\big], \tag{4}$$

$$\text{score}_c(w) = \text{tf}_c(w)\left(1 + \log \tfrac{n_{\text{cl}}}{\max(\text{df}(w),\,1)}\right), \tag{5}$$

where n_{cl} is the number of clusters. The label for cluster c is the top-4 tokens by $\text{score}_c(\cdot)$.

G.5 Cluster-quality metrics

We report three metrics. Let $E_c = \{e_j : \ell_j = c\}$ be the set of MiniLM embeddings in cluster c, with $n_c = |E_c|$.

Silhouette. Standard sklearn silhouette in the 2D UMAP space, restricted to non-noise points: $+0.01$. A near-zero silhouette indicates clusters that are well-separated on the embedding plane but not internally tight, consistent with the wide-but-distinct clusters of fig. 16.

Mean intra-cluster cohesion. Mean intra-cluster cosine similarity in MiniLM space, averaged over non-singleton clusters:

$$\rho_c = \frac{1}{n_c(n_c-1)} \sum_{j,j'\in E_c,\, j\neq j'} \langle e_j,\, e_{j'}\rangle, \qquad \bar{\rho} = \frac{1}{|\{c : n_c > 1\}|} \sum_{c:\, n_c>1} \rho_c = 0.63. \tag{6}$$

Temporal-vocabulary fraction. We define a temporal-vocabulary heuristic as the fraction of individual feature explanations matching a fixed case-insensitive regex for sequence/position cues:

$$R = \texttt{\textbackslash b(sequenc|consecut|context|preced|follow|}$$
$$\texttt{temporal|window|position|previous|next|order)\textbackslash w*}.$$

For the multiset S of feature explanations,

$$\text{temporal} = \frac{1}{|S|} \sum_{s\in S} \mathbb{1}[R \text{ matches } s] = 0.68.$$

Rescaling to $[0, 10]$ gives the 6.80 score reported in the body. This is a regex heuristic over explanations, not a separate LLM judge.

G.6 Per-sentence selection: the exclusive score

For the per-sentence activation map of fig. 17, we fix a 32-token chain and a window length $T{=}5$, then compute a $32 \times d_{\text{sae}}$ activation matrix $A \in \mathbb{R}_{\geq 0}^{L \times d_{\text{sae}}}$ ($L{=}32$) by sliding the T-window across the chain and recording the per-position decoder contribution of each feature.

We then assign *one* feature to each token position via the **exclusive score**, which rewards features whose total activation mass is concentrated at p:

$$\text{score}[p, j] \;=\; \frac{A_{p,j}^2}{\varepsilon + \sum_{p'=1}^{L} A_{p',j}}. \tag{7}$$

The numerator's square forces a feature to actually fire strongly at p (not just rarely elsewhere); the denominator penalises features that fire broadly across the chain. With $\varepsilon = 10^{-8}$, assignment is greedy (algorithm 1): positions are processed in descending order of their best per-position score, and each feature can be claimed by at most one position, so the strongest position-feature pairs win first. The result is exactly $L{=}32$ features, each fingerprinted to a single token position. The alternative *magnitude* selection (top-L features by $\sum_p A_{p,j}$, no exclusivity constraint) lets a single high-firing feature dominate multiple positions and obscures the diagonal-band geometry; we use it only for sanity checks.

Algorithm 1 EXCLUSIVESELECT: per-position feature assignment.

Require: activation matrix $A \in \mathbb{R}_{\geq 0}^{L \times d_{\text{sae}}}$
Ensure: selection $\text{sel} \in \{1, \ldots, d_{\text{sae}}\}^L$ with all entries distinct
$\text{score}[p, j] \leftarrow A_{p,j}^2 \,/\, \big(\varepsilon + \sum_{p'} A_{p',j}\big)$
$\text{best}[p] \leftarrow \max_j \text{score}[p, j]$ for each p
$\pi \leftarrow \text{argsort}_{\,\text{best},\,\downarrow}\big(\{1, \ldots, L\}\big)$
$\text{used} \leftarrow \emptyset$; $\text{sel}[p] \leftarrow -1$ for all p
for $p \in \pi$ **do**
    for j in features sorted by $\text{score}[p, j]$ descending **do**
        if $j \notin \text{used}$ **then** $\text{sel}[p] \leftarrow j$; $\text{used} \leftarrow \text{used} \cup \{j\}$; **break**
        end if
    end for
end for
return sel

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction accurately summarize the paper's contributions and scope. We present temporal crosscoders as a complementary dictionary-learning architecture for temporally distributed features, introduce TempBench as a synthetic and real-world evaluation suite, and report both positive and negative results. The claims are scoped to the models, datasets, and benchmarks evaluated, and the paper explicitly notes that temporal crosscoders do not uniformly dominate other architectures.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Section 6 discusses the two key scope limits explicitly. First, TempBench is a deliberately compact benchmark; a more systematic temporal evaluation suite, across both synthetic and real-world tasks, is needed to fully map each architecture's capability profile. Second, we focus on the TXC capability profile rather than a theoretical account of when they should outperform alternatives. This leaves three concrete open directions: a-priori prediction of task winners, systematic hyperparameter optimisation for TXCs, and a deeper understanding of the relationship between TXCs and TSAEs.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper does not present formal theoretical results, theorems, or lemmas requiring proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.

- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: Code, data, configuration files, and reproduction instructions are provided in the supplementary material. The paper and appendices specify the architectures, datasets, hyperparameters, seeds, and evaluation protocols used for the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: Code, data-generation scripts, configuration files, and instructions for reproducing the main experimental results are provided in the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The main text and appendices specify the training and evaluation details needed to interpret the results. Synthetic benchmark construction and probe definitions are given in Appendix B; sparse-probing training and evaluation details are given in Appendix C; emergent-misalignment and HH-RLHF details are given in Appendices D–E; and backtracking details are given in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: All multi-seed experiments report error bars in the figures and accompanying tables. Each architecture is trained over 3 seeds on the synthetic and sparse-probing experiments (sections 4 and 5.1) and over 2 seeds on the case-study experiments (sections 5.2 to 5.4); whiskers show the seed min–max range on the headline bar charts, std-across-seeds envelopes on the $\overline{\mathrm{AUC}}$ vs. k curves, and bootstrap confidence intervals on the per-question Δgc in the backtracking case study. The factor of variability captured is the random seed used for dictionary initialization and training data shuffling. Computation methods (trapezoidal-mean propagation for $\overline{\mathrm{AUC}}$, min/max for the bar charts, bootstrap for Δgc) are documented in the per-component appendices.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper-bound experiments (the seven components reported in this paper) totalled approximately 330 H100-equivalent GPU-hours, and the full research project — including failed hypotheses, dropped architecture variants, and exploration on side branches — used approximately 820 additional H100-equivalent hours, for a grand total of approximately 1,150 H100-equivalent GPU-hours. Compute ran on a heterogeneous mix of H100 80GB, H200 141GB, A40 48GB, RTX PRO 6000 Blackwell 96GB, and RTX 5090 32GB GPUs, with conversion to H100-equivalent applied per device (H100 $\approx$ 1.0, H200 $\approx$ 1.3, A40 $\approx$ 0.4, RTX 5090 $\approx$ 0.6, RTX PRO 6000 $\approx$ 0.8). Per-component subtotals — C1 toy Markov sweep $\approx$ 1 hr, C2 coupled-features sweep $\approx$ 135 hr, C3 sparse probing $\approx$ 95 hr, C4 qualitative latents $\approx$ 10 hr (cache-hit on C3), C5 RLHF steering $\approx$ 30 hr, C6 emergent misalignment $\approx$ 50 hr, C7 backtracking $\approx$ 10 hr — are tabulated alongside per-cell timing and the supplementary reproduction guide.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed and confirmed the research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No].

Justification: The manuscript does not include a dedicated broader-impact discussion. The work is methodological research on dictionary-learning architectures and evaluation for language-model representations, and we do not identify direct application-specific societal impacts beyond those generally associated with progress in model interpretability and evaluation.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A].

Justification: This work does not release pretrained language models, image generators, scraped datasets, or other high-risk assets requiring special release safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes].

Justification: We cite the creators of the existing models, datasets, benchmarks, and codebases used in this work, and provide license and access information for these assets in the supplementary material. These assets include the pretrained language models and LoRAs, public benchmark datasets, existing SAE/dictionary-learning baselines, and evaluation suites used in the experiments. All assets are used according to their stated licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes].

Justification: We provide the new code, synthetic data generators, configuration files, and evaluation scripts in the supplementary material, together with documentation and instructions for reproducing the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: The paper does not involve crowdsourcing or newly conducted research with human participants. All evaluations use existing datasets, synthetic data generators, automated probes, or LLM-based judges. No new human annotations were collected for this work, and no workers or study participants were recruited or compensated.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve newly conducted human-subjects research. We do not recruit participants, collect data from individuals, or intervene on human subjects. Existing datasets used in the experiments are handled as research assets rather than newly collected human-subject data, and LLM-based judges are used for some behavioral evaluations instead of human annotators.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes].

    Justification: We use LLMs as automated evaluators in a subset of the real-model experiments. In the backtracking experiment, an LLM judge is used to classify whether generated continuations contain genuine backtracking events. In the emergent-misalignment experiment, an LLM judge is used to score coherence and alignment for generated responses. These uses are part of the evaluation pipeline and are described in the corresponding experimental sections and appendices. LLMs were not used as trainable components of the proposed architecture or as sources of ground-truth labels for the synthetic benchmarks.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.