EXSAL: EXPLAINING SPARSE AUTOENCODER LATENTS WITH DOMAIN-ADAPTED LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Autoencoders (SAEs) decompose language model activations into interpretable features. The standard approach to describing these features prompts a general-purpose LLM with the feature's top-activating tokens and asks it to produce a natural-language description. However, when the analyzed model has been domain-adapted, its SAE features encode specialized knowledge that a general-purpose explainer may fail to describe accurately. We argue that the domain-adapted model itself is better positioned to explain its own features. To this end, we introduce EXSAL (**Ex**plaining **S**parse **A**utoencoder **L**atents with Domain-Adapted LLMs), a mechanistic interpretability framework for analyzing features from domain-adapted LLMs. Given a domain-adapted model, EXSAL trains an SAE on its activations, and uses that same model to generate natural-language descriptions of the learned features. The framework is applied to two independently adapted models: an enterprise-software model and a finance model. Our experiments show that EXSAL outperforms the general-purpose LLM explainer and narrows the gap to a far larger frontier model, using only the domain-adapted model already at hand and requiring no external explainer. These observations indicate that EXSAL produces more accurate feature descriptions, providing a stronger foundation for analyzing SAE latents in domain-adapted LLMs without requiring a separate explainer model.

1 INTRODUCTION

Large language models (LLMs) have become general-purpose tools across a wide range of tasks (OpenAI et al., 2024; Brown et al., 2020), which has made understanding their internal computations increasingly important (Elhage et al., 2021). This has driven a growing body of work in mechanistic interpretability (Bereska & Gavves, 2024; Bricken et al., 2023; Templeton et al., 2024b; Nanda et al., 2023), pursued for two reasons. First, knowing which concepts a model represents internally helps understand what it has learned and whether that aligns with intended behavior (Templeton et al., 2024b). Second, identifying these concepts opens the door to intervening on them by amplifying or suppressing individual features to steer model behavior (Templeton et al., 2024b).

A natural starting point would be to inspect individual neurons, but neurons are typically polysemantic: a single neuron fires for many unrelated concepts, so it does not correspond to one interpretable unit (Elhage et al., 2022). A common response is to map the model's activations into a larger, sparser space whose dimensions align more closely with single concepts, using Sparse Autoencoders (Cunningham et al., 2023), and related methods such as transcoders (Templeton et al., 2024a; Marks et al., 2024) and crosscoders (Lindsey et al., 2024; Minder et al., 2026). These methods recover directions that are far more monosemantic than raw neurons and have been scaled to frontier models (Templeton et al., 2024b).

Recovering a large dictionary of features, however, only solves one part of the problem. A trained SAE yields tens of thousands of latents whose meaning is encoded implicitly and cannot be read off directly. This scale arises because SAEs apply an expansion factor to the model's hidden dimension with hidden sizes on the order of thousands and large expansion factors. Labeling them by hand is infeasible at this scale. The now-standard approach is therefore to delegate the description to an LLM explainer. An explainer model is shown a feature's top-activating contexts and asked to summarize, in natural language, the concept that unifies them (Bills et al., 2023; Paulo et al., 2025).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

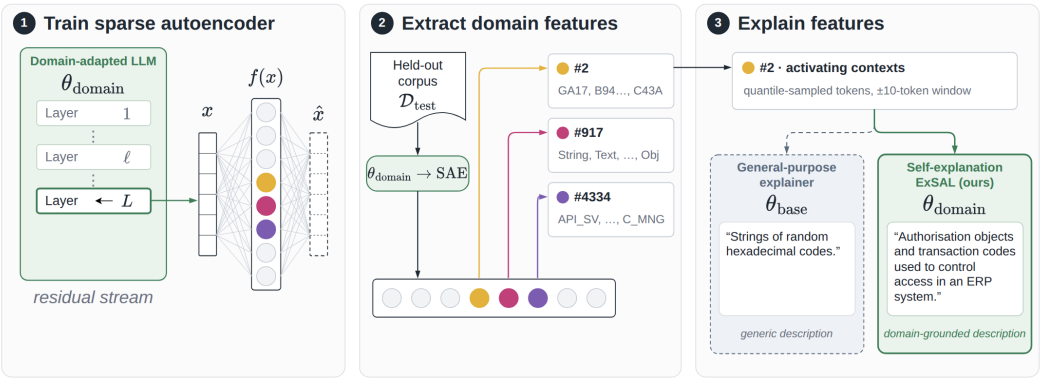


Figure 1: ExSAL trains a JumpReLU SAE on the activations of the residual stream in the last layer from a domain-adapted LLM (1). Then, a held-out set of tokens is passed to the SAE, and the activated latents are passed through a filter to retain domain-relevant features (2). Finally, the same domain-adapted LLM is used to generate natural-language descriptions of the learned features (3).

While early work focused mainly on general features in large state-of-the-art models (Templeton et al., 2024b; Bills et al., 2023; Bricken et al., 2023), inspecting fine-tuned models is becoming increasingly important, as it can reveal what knowledge a model has internalized and enable detection and steering of unwanted behavior (Kopf et al., 2024; Simon & Zou, 2024; Minder et al., 2026). In the domain-adapted setting, general-purpose explainer models have not been trained on the domain, so it is natural to question whether their descriptions capture the specialized concepts. This raises a fundamental question: is the domain-adapted model itself better suited to describe its own features than a general-purpose explainer? If a model has internalized specialized domain knowledge during training, that same knowledge may be what is needed to recognize and articulate the concepts its SAE features encode. We investigate this hypothesis with ExSAL (**Ex**plaining **S**parse **A**utoencoder **L**atents with Domain-Adapted LLMs), a framework that uses the domain-adapted model as its own explainer to produce better SAE feature descriptions.

ExSAL is designed around the following research questions:

- ***RQ1:*** Does using a domain-adapted model as the explainer produce better SAE feature descriptions than using the base model?
- ***RQ2:*** Can RAG explainers match domain-adapted explainers for SAE feature description, or does domain knowledge need to be encoded in the model's weights?
- ***RQ3:*** How does the quality of SAE feature descriptions from a domain-trained explainer compare to those from a frontier model?
- ***RQ4:*** Does using the domain-adapted model as both explainer and LLM judge introduce a bias?
- ***RQ5:*** Does the advantage of domain-adapted explainers for SAE feature description transfer to a different domain and model family?

2 Methods

We present our ExSAL framework below. To obtain domain-specific features, an SAE is first trained (Section 2.1) from scratch on activations produced by passing a generic corpus (Section 2.2) through θ_{base}, a pre-trained LLM. Training an SAE from scratch rather than fine-tuning a pre-existing SAE ensures ExSAL is universal even when no pre-trained SAEs are available for the model under study, which is the common case. The resulting SAE is then fine-tuned on activations produced by passing domain-specific data (Section 2.2) through θ_{domain}, a domain-adapted model. This lets the SAE specialize toward domain-specific latents while retaining the general features learned in the first stage. The held-out test split of the combined corpus (Section 2.2) is then passed through θ_{domain}. For each token, the residual-stream activations are captured and passed through

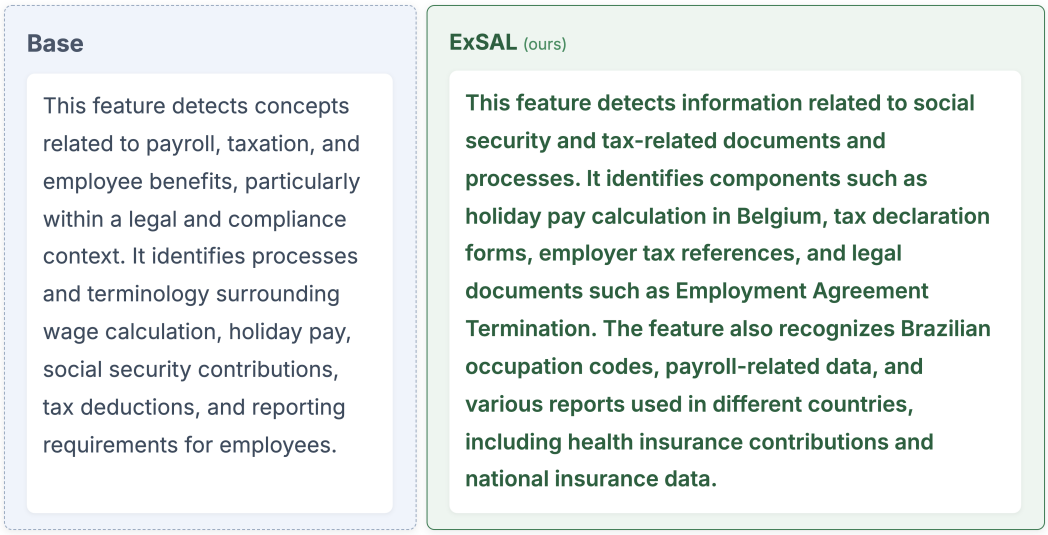


Figure 2: Example feature from the enterprise-software domain described independently by BASE and EXSAL (Ours) from the same input contexts sampled across activation quantiles, including tokens such as *STAT*, *TAX*, *employee*, and *Payroll* along with their surrounding context window. The domain-adapted explainer produces a more specific description, reflected in higher scores across all three metrics: Simulation Score 0.29 (BASE) vs. 0.39 (EXSAL), Detection AUROC 0.98 vs. 1.00, and Fuzzing Accuracy 0.60 vs. 0.67.

the SAE encoder, after which up to 150 active features are recorded together with their activation values and the token's surrounding context. The resulting per-token record of active features is used to identify domain-relevant features. A frontier model (Claude Sonnet 4.6) is prompted with each feature's tokens sampled across activation quantiles (Appendix A.5), and the 50 most domain-representative features are selected via a two-stage LLM-as-judge filter (Section 2.3). Finally, for each of the 50 selected features, tokens on which the feature activates are sampled by quantile (Appendix A.5) and passed to θ_{domain}, to return a concise natural-language description.

We use a Gemma-3 12B instruction-tuned model (Team et al., 2025) for θ_{base}, and fine-tune its base model on domain data via Continued Pre-Training (CPT) and Instruction Fine-Tuning (IFT) for θ_{domain} (see Appendix A.6 for details), but EXSAL is model-agnostic. To ensure θ_{domain} does not outperform BASE simply by being a stronger base model, we compare the two on MMLU (Hendrycks et al., 2021) and find θ_{domain} is in fact slightly worse; its advantage as an explainer must therefore come from domain knowledge rather than general capability (Section A.6.1).

We compare our EXSAL framework against a baseline framework: BASE, which uses θ_{base} as the explainer.

2.1 SAE TRAINING

An SAE takes a model's dense residual-stream activations, projects them into a much wider latent space, in which only a handful of dimensions are active at once, and then reconstructs the original activation from that sparse representation. Since few latents fire on any given token, each one can specialize to a single interpretable concept. We adopt the JumpReLU variant (Rajamanoharan et al., 2024b), which gates each latent with a learned per-latent threshold rather than an L_1 penalty (Section 2.1.1). We train the SAE on the activations of the last layer. As a first step we normalize and center the activations (Appendix A.2), then optimize a reconstruction loss with a sparsity regularizer that keeps the number of active latents close to a desired target (Section 2.1.2).

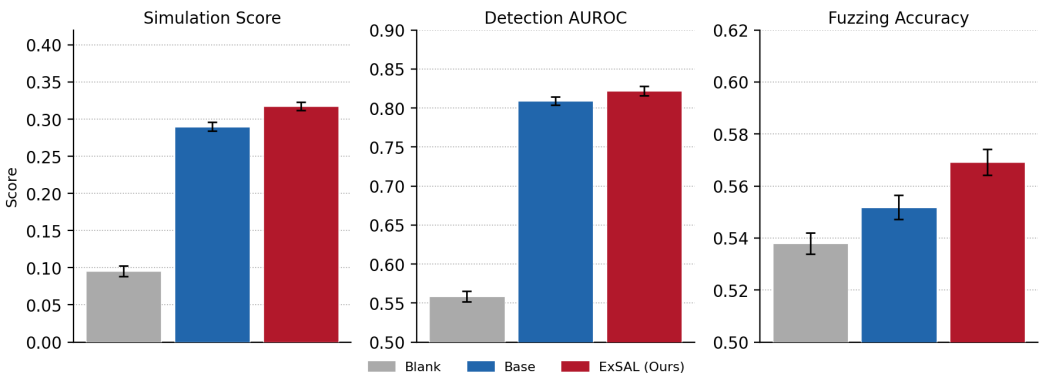


Figure 3: ExSAL outperforms Base on all three metrics confirming that the descriptions are more informative (enterprise-software domain, 50 features, θ_{domain} judge, 95% CIs). Both substantially exceed the Blank baseline.

2.1.1 Architecture

An SAE consists of an encoder and a decoder that reconstructs the original activation, trained jointly end-to-end. For a residual-stream activation $\boldsymbol{x} \in \mathbb{R}^{d_{\text{model}}}$, the encoder and decoder are defined as:

$$\boldsymbol{f}(\boldsymbol{x}) = \text{JumpReLU}_{\boldsymbol{\theta}}\big(\boldsymbol{W}_{\text{enc}}(\boldsymbol{x} - \boldsymbol{b}_{\text{pre}})\big), \qquad \hat{\boldsymbol{x}} = \boldsymbol{W}_{\text{dec}}\,\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{b}_{\text{pre}}. \tag{1}$$

Here $\boldsymbol{W}_{\text{enc}} \in \mathbb{R}^{n \times d_{\text{model}}}$ and $\boldsymbol{W}_{\text{dec}} \in \mathbb{R}^{d_{\text{model}} \times n}$ are trained weight matrices, and $\boldsymbol{b}_{\text{pre}}$ is a pre-encoder bias that is subtracted before encoding and added back after decoding. The sparsity itself comes from the JumpReLU activation, which applies a per-latent hard threshold θ_i that keeps a latent at its raw value once it crosses the threshold and sets it to zero otherwise:

$$\text{JumpReLU}_{\theta_i}(z_i) = z_i \cdot H(z_i - \theta_i), \tag{2}$$

with H the Heaviside step function. This decouples reconstruction strength from the sparsity decision. Because H has zero gradient almost everywhere, we optimize the thresholds through a straight-through estimator that replaces the derivative of H with a Gaussian kernel of bandwidth proportional to the local activation scale (Rajamanoharan et al., 2024b).

2.1.2 Training objective

The SAE is trained with a reconstruction term and a sparsity regularizer (McDougall et al., 2025), defined in terms of the encoder $\boldsymbol{f}$ and reconstruction $\hat{\boldsymbol{x}}$ from Equation 1:

$$\mathcal{L} = \underbrace{\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2}_{\text{reconstruction}} + \lambda \frac{2}{L_0^\star}\big(\|\boldsymbol{f}(\boldsymbol{x})\|_0 - L_0^\star\big)^2. \tag{3}$$

The first term is the mean squared reconstruction error and the second is a quadratic penalty. Rather than solely optimizing for sparsity, the loss function penalizes deviations of the realized sparsity ($\|\boldsymbol{f}(\boldsymbol{x})\|_0$) from the target number of active latents ($L_0^\star$) in either direction, which stabilizes training around a sparsity level. More details as well as a full list of hyperparameters are listed in Appendix A.1.

2.2 Data

The generic corpus is drawn from FineWeb (Penedo et al., 2024), a large open web dataset, and the domain corpus consists of publicly accessible data from an enterprise-software documentation website. In total, the combined corpus comprises roughly 2×10^7 tokens. We partition it into a $70\%/15\%/15\%$ train/validation/test split, where the training split is used for SAE training, the validation split for tuning SAE hyperparameters such as $L_0^\star$ and λ, and the test split for all feature analysis reported in this paper.

2.3 TWO-STAGE LLM-AS-JUDGE FEATURE FILTER

The goal of our experiment is to compare explainer models on domain-specific features, which requires a set of features that encode domain-specific content. Identifying which features to inspect is non-trivial: feature selection has been shown to meaningfully affect evaluation outcomes (Wang et al., 2025), and various strategies have been proposed (Templeton et al., 2024b; Gao et al., 2024; Goel et al., 2026). We therefore apply a two-stage LLM-as-judge filter rather than sampling randomly.

Since the dictionary contains tens of thousands of latents, we narrow the pool automatically in two stages. In the first stage, we prompt a frontier model (Claude Sonnet 4.6) with sampled activating token contexts (sampling described in Appendix A.5) and ask it to assign a judge score, which is the sum of three sub-scores: domain purity (0–4, how exclusively the feature fires on domain tokens), content richness (0–4, whether it captures meaningful content rather than structural noise), and specificity (0–4, how precisely the feature can be interpreted).

We rank all features by judge score and pass the top 100 candidates to a second LLM call. At this scale the candidate set is small enough that the model can compare features directly against each other rather than scoring them independently. Then the second call receives, for each candidate, its activation frequency, its first-stage judge score, and the natural-language description from the first stage, and selects the 50 most domain-representative features.

To validate the importance of this feature selection process, we run a separate evaluation on 50 randomly sampled features from the same SAE. Across this random set, scores collapse for every explainer configuration, because most randomly drawn features fall into one of three categories: polysemantic features that respond to unrelated concepts, syntactic features that fire on surface-level patterns rather than semantic content, or features with no connection to the target domain (Appendix A.8). Details of the scoring rubric and the prompts are given in Appendix A.11.

3 RESULTS

Experimental Setup To evaluate how well each explainer performs, we set aside held-out tokens: five tokens from each of the five activating quantiles and five tokens from the non-activating positions, resulting in 30 held-out tokens per feature. We evaluate description quality on three metrics, each using an LLM judge that receives the feature description as input. Simulation Score (Bills et al., 2023) is the Pearson correlation between a judge's predicted activation strengths and the ground-truth activations on held-out tokens. Detection AUROC (Paulo et al., 2025) operates at the sentence level, where the judge is given a complete sentence and must predict whether the feature activates anywhere in it, producing a binary yes/no decision. Fuzzing Accuracy (Paulo et al., 2025) operates at the token level, where individual tokens within a sentence are marked as active and the judge must decide for each whether the marking is correct. All three metrics are higher-is-better. We use θ_{domain} as the primary judge, since it has internalized domain knowledge through Continued Pre-Training. Whether this choice introduces bias is tested empirically in RQ4, which repeats the evaluation with an independent judge. Full metric definitions and judge prompts are provided in Appendix A.7.

Overview The analysis is organized around the five research questions introduced in Section 1. RQ1–3 compare EXSAL against a general baseline, RAG-augmented variants, and a frontier model. RQ4 checks for judge bias, and RQ5 tests generalizability on a second domain. The section closes with a qualitative analysis of individual feature descriptions, which helps to build intuition for what the quantitative differences look like in practice. All reported values are mean-aggregated over 50 features with 95% confidence intervals obtained by bootstrapping (each feature contributes 3 quantile bins × 3 explainer runs × 3 judge calls). Full results tables for all explainer and judge configurations are provided in Appendix A.1.2, as well as details on the bootstrapping procedure in Appendix A.3. All configurations are compared against a BLANK baseline whose descriptions are uninformative strings (an empty string, a newline, or a tab character), serving as a baseline for metric performance in the absence of any description.

RQ1: Does using a domain-adapted model as the explainer produce better SAE feature descriptions than using the base model? We begin with the most direct comparison: does the EXSAL pipeline produce more accurate feature descriptions than the BASE pipeline? Our experiments give a clear answer: EXSAL outperforms BASE on all three metrics (Figure 3). The increase in Simulation Score and Detection AUROC indicates that descriptions generated by a domain-trained explainer better capture the underlying concept of a feature: the judge can more accurately predict activation strength and more reliably distinguish activating from non-activating contexts. The gain on Fuzzing Accuracy is smaller, which is expected as it requires a per-token decision across multiple candidates rather than a single binary judgment per context. This suggests that domain knowledge primarily helps at the conceptual level rather than at identifying individual activating tokens. Nevertheless, EXSAL still outperforms BASE across all three metrics. Both BASE and EXSAL substantially exceed the BLANK baseline, confirming that the descriptions carry meaningful signal.

Table 1: EXSAL (Ours) outperforms BASE on all three metrics and is competitive with the FRONTIER model (enterprise-software domain, 50 features, θ_{domain} judge, 95% CIs). RAG hurts performance in nearly all cases. FRONTIER leads overall, which we attribute to its superior general ability to produce accurate and coherent natural-language descriptions. **Bold** marks the best configuration per category (open model, closed model).

	Simulation Score ↑	Detection AUROC ↑	Fuzzing Accuracy ↑
BASE	0.290 ± 0.006	0.809 ± 0.006	0.552 ± 0.005
BASE + RAG	0.302 ± 0.006	0.794 ± 0.006	0.547 ± 0.005
EXSAL (Ours)	$\mathbf{0.317 \pm 0.005}$	$\mathbf{0.822 \pm 0.006}$	$\mathbf{0.569 \pm 0.005}$
EXSAL (Ours) + RAG	0.298 ± 0.007	0.800 ± 0.006	0.521 ± 0.005
FRONTIER	$\mathbf{0.351 \pm 0.005}$	$\mathbf{0.863 \pm 0.005}$	0.578 ± 0.005
FRONTIER + RAG	0.349 ± 0.006	0.857 ± 0.005	$\mathbf{0.579 \pm 0.005}$

RQ2: Can RAG explainers match domain-adapted explainers for SAE feature description, or does domain knowledge need to be encoded in the model's weights? RQ1 showed that using a domain-adapted model as the explainer produces better feature descriptions. A natural follow-up is whether the same effect can be achieved without training: instead of encoding domain knowledge in model parameters, one could retrieve relevant context at inference time via RAG. To test this, we evaluate RAG variants of both BASE and EXSAL. For retrieval, we use BGE-M3 (Chen et al., 2025), which combines dense semantic similarity with sparse lexical matching. The hybrid approach is important for domain corpora, where abbreviations and product identifiers require exact lexical overlap rather than semantic proximity alone. The vector database is built from the same corpus used to train θ_{domain}, so both approaches have access to the same domain information. At inference time, the top-activating tokens are used as the retrieval query, and the top-5 retrieved chunks are attached to the explainer's prompt alongside the tokens. Full setup and hyperparameter details are given in Appendix A.4.

Table 1 shows that RAG hurts performance in nearly all cases in this domain; as we show in RQ5, this effect does not hold universally. For BASE, RAG yields a small gain on Simulation Score, but Detection AUROC and Fuzzing Accuracy both drop (Table 1), suggesting that retrieved context provides some signal for activation prediction but makes context-level and token-level discrimination harder. For EXSAL, the effect is negative across all three metrics. We hypothesize that the retrieved passages distract the explainer from the quantile-sampled tokens that should anchor the description: instead of focusing on what the presented tokens have in common, the model attends to the retrieved context and produces descriptions that are broader or less grounded in the actual activation pattern.

RQ3: How does the quality of SAE feature descriptions from a domain-trained explainer compare to those from a frontier model? The current standard practice for SAE feature explanation is to use a large frontier model as the explainer. We compare EXSAL against FRONTIER, for which we use Claude Opus 4.7 as the explainer model. We also include FRONTIER + RAG to investigate whether providing a frontier model with retrieved domain context further improves its descriptions.

Table 1 shows that FRONTIER configurations achieve the highest scores overall, yet EXSAL remains competitive despite being a much smaller model. We attribute FRONTIER's advantage to its greater training data and model size, which translates to a more robust general ability to generate accurate and coherent natural-language descriptions. Consistent with the pattern from RQ2, adding RAG to FRONTIER offers no consistent benefit (Table 1). Following the pattern observed in RQ1, one would expect that a frontier model additionally continued-pre-trained on domain data could combine both advantages and push performance beyond what either approach achieves in isolation. We leave this as an open question for future work as the frontier model used in our experiments is closed-source and does not permit continued pre-training, and even open-weight frontier models of that size would require substantial compute resources to train at that scale.

RQ4: Does using the domain-adapted model as both explainer and LLM judge introduce a bias? It is well established that LLM judges can exhibit self-preference bias, assigning higher scores to outputs generated by the same model (Panickssery et al., 2024). This raises the question of whether the results are partially an artifact of the θ_{domain} model serving as both explainer and judge. The potential bias here is stylistic; take Simulation Score as an example: the judge is asked to predict, on a scale of 0–9, how strongly a held-out token activates a feature given its description. If the judge and the explainer are the same model, the judge may find it easier to reason about descriptions written in its own style, inflating scores independently of whether those descriptions are actually more accurate. The same reasoning applies to Detection AUROC and Fuzzing Accuracy. To rule this out, we use θ_{base} + RAG as an independent judge, applying the same RAG setup described in RQ2. This gives the judge access to domain information through retrieval, without sharing any domain-specific training with the EXSAL explainer.

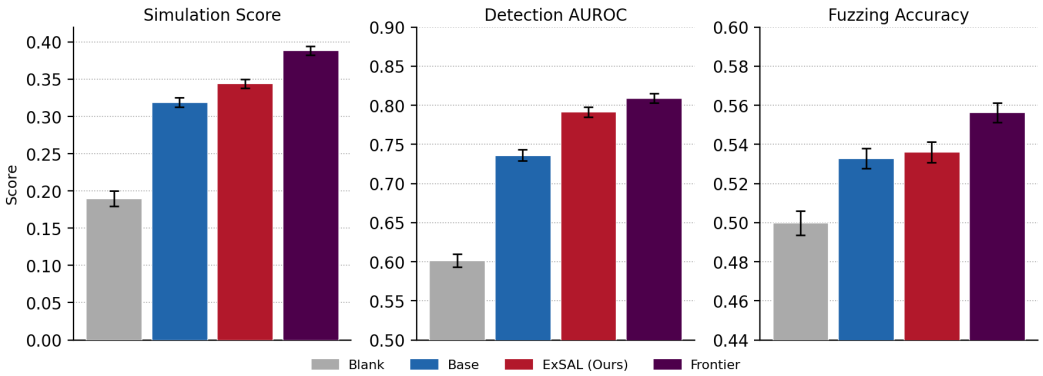


Figure 4: The ranking of explainer configurations is preserved under the independent θ_{base} + RAG judge: EXSAL outperforms BASE and FRONTIER leads overall, confirming that the results are not an artifact of EXSAL-judge bias (enterprise-software domain, 50 features, 95% CIs).

Figure 4 shows that the ranking is preserved under the independent judge. EXSAL outperforms BASE on all three metrics while remaining competitive with FRONTIER (Figure 4). The consistent ordering across both judges confirms that our results reflect genuine differences in description quality rather than a bias introduced by the choice of judge.

RQ5: Does the advantage of domain-adapted explainers for SAE feature description transfer to a different domain and model family? So far, all experiments have used a single domain. To assess generalizability, we replicate the full pipeline on a second, unrelated domain. We use Llama-Fin-8B (Ke et al., 2025) as θ_{domain}, a version of Meta-Llama-3-8B-Instruct (Grattafiori et al., 2024) adapted on the FinTrain financial corpus (Ke et al., 2025). Analogously to the primary domain, Llama-Fin-8B serves as the model used by EXSAL, Meta-Llama-3-8B-Instruct serves as θ_{base} used by BASE, and the RAG vector database is built from the same corpus the Llama-Fin-8B model was trained on. The FRONTIER model and all other pipeline settings are identical to the primary domain.

Table 2 shows that the finance domain results replicate the core finding: EXSAL outperforms BASE on Simulation Score and Detection AUROC (Table 2). FRONTIER again leads on both metrics. On

Table 2: EXSAL outperforms BASE on Simulation Score and Detection AUROC, replicating the primary finding on a different model family (finance domain, 50 features, θ_{domain} judge, 95% CIs). **Bold** marks the best configuration per category (open model, closed model).

	Simulation Score ↑	Detection AUROC ↑	Fuzzing Accuracy ↑
BASE	0.426 ± 0.007	0.923 ± 0.004	0.747 ± 0.005
BASE + RAG	0.428 ± 0.007	0.930 ± 0.003	$\mathbf{0.758 \pm 0.005}$
EXSAL (Ours)	$\mathbf{0.456 \pm 0.007}$	0.935 ± 0.003	0.734 ± 0.004
EXSAL (Ours) + RAG	0.451 ± 0.007	$\mathbf{0.937 \pm 0.003}$	0.736 ± 0.005
FRONTIER	0.471 ± 0.007	$\mathbf{0.954 \pm 0.002}$	$\mathbf{0.749 \pm 0.004}$
FRONTIER + RAG	$\mathbf{0.473 \pm 0.007}$	0.953 ± 0.003	0.748 ± 0.004

Fuzzing Accuracy, however, the picture is different: BASE slightly outperforms EXSAL, and BASE + RAG achieves the highest Fuzzing Accuracy of all configurations, surpassing even FRONTIER. This suggests that in the finance domain, token-level precision benefits from retrieval in a way not observed in the enterprise-software domain. More broadly, RAG does not exhibit the consistently negative effect seen there: BASE + RAG improves over BASE on all three metrics, and FRONTIER + RAG matches FRONTIER closely. One possible explanation is that the FinTrain corpus is better suited for retrieval since it is $\sim 4\times$ larger, making the retrieved passages more useful to the explainer. The finance domain results are also higher in absolute terms than the enterprise-software domain, which we attribute to differences in either the model family or the training corpus. These results confirm that the EXSAL advantage over BASE observed in the enterprise-software domain is not specific to a single corpus or model family, supporting the generalizability of our main finding.

Qualitative Analysis To complement the quantitative results, we conduct a qualitative analysis by examining one example feature in Figure 2. As shown in the figure, EXSAL produces more precise descriptions, for instance referencing Belgian holiday pay rules, Brazilian occupation codes, and severance premium reserve funds, while BASE remains at a generic level. This tendency of BASE to produce broader descriptions is a pattern observed across multiple features. In some cases it takes a more extreme form, where BASE enumerates input tokens rather than explaining the underlying concept. In one case, BASE produces "[...] it activates strongly for terms like 'Data,' 'Wage,' and 'Employee' [...]" listing the description tokens rather than explaining what the feature detects.

4 RELATED WORK

4.1 SPARSE AUTOENCODERS

A key obstacle to interpreting neural networks is superposition (Elhage et al., 2022; Arora et al., 2018): neurons are polysemantic and human-interpretable concepts correspond to linear directions in activation space rather than individual neurons (Park et al., 2024). Disentangling such features is a central goal of mechanistic interpretability (Bereska & Gavves, 2024). Sparse Autoencoders address this by mapping the polysemantic activations of a layer to a larger, sparse hidden space whose dimensions ideally align with single interpretable concepts (Templeton et al., 2024b; Cunningham et al., 2023), an approach since scaled to frontier models (Gao et al., 2024; Templeton et al., 2024b) and refined through architectures such as TopK (Gao et al., 2024), JumpReLU (Rajamanoharan et al., 2024b), and Gated SAEs (Rajamanoharan et al., 2024a). Once an SAE is trained, its features can be used to inspect or steer specific model behaviors (Templeton et al., 2024b), but a trained SAE only yields a record of which tokens activate which features. To inspect or steer a particular semantic concept, one must first understand what each latent represents.

4.2 AUTOMATIC SAE FEATURE INTERPRETATION

Since the number of features is far too large to be labeled manually, researchers have proposed several techniques for determining what a feature represents. Some approaches bypass external explainers by patching feature embeddings directly into the model (Ghandeharioun et al., 2024; Chen et al., 2024; Kharlapenko et al., 2024), but none consider how domain adaptation affects explanation

quality. In practice, the most common approach delegates labeling to an external language model, prompting it with a feature's top-activating tokens to obtain a natural-language description (Bills et al., 2023; Paulo et al., 2025). Subsequent work improves description quality, for example through agentic explainer–critic loops (Han et al., 2026), interpretability agents that iteratively run their own experiments on a unit (Shaham et al., 2025), or explainers trained directly on activation-based reward signals (He et al., 2026).

4.3 SAEs in Domain-Adapted Contexts

As SAEs move beyond general-purpose settings, domain-aware interpretation becomes important. Large off-the-shelf SAE suites such as Gemma Scope (Lieberum et al., 2024; McDougall et al., 2025) make it increasingly practical to obtain domain-adapted latents by fine-tuning existing SAEs rather than training them from scratch, shifting the main challenge from obtaining features to describing them well. Existing work tackles this differently: Simon & Zou (2024) validate SAE features of a protein language model against curated biological databases, while Minder et al. (2026) label latents with an LLM explainer aided by manual inspection. These studies show that domain-specific features differ qualitatively from general-purpose ones.

5 Conclusion

In this work, we introduced ExSAL, a framework that uses a domain-adapted model as its own SAE feature explainer. In the enterprise-software domain, our primary evaluation setting, ExSAL outperforms the base pipeline (Base) across all three evaluation metrics. This pattern largely replicates in a second, independent experiment on a finance domain using a different model family. We also find that RAG-augmented explainers are not consistently able to match domain-trained ones, suggesting that encoding domain knowledge in the model's weights is more effective than injecting it at inference time via retrieval. ExSAL narrows the gap to Frontier despite θ_{domain} being considerably smaller and using no external explainer, and the observed pattern suggests that a frontier model additionally trained on domain data would likely push performance further. Taken together, these results show that a domain-adapted model can serve as an effective explainer of its own SAE latents, offering a practical path to interpreting domain-adapted LLMs without relying on a separate explainer model.

6 Limitations

Our experiments use the residual stream activations of the model's last layer for the JumpReLU SAE, rather than extensions such as transcoders (Marks et al., 2024) or crosscoders (Lindsey et al., 2024), which allowed us to run a controlled comparison across six explainer configurations and two domains within a feasible compute budget. Notably, even within this focused scope, the ExSAL advantage is consistent and replicates across settings. We did not evaluate a frontier model additionally continued-pre-trained on domain data: the frontier model used in our experiments (Claude Opus 4.7) is closed-source and does not permit continued pre-training, and even open-weight frontier models (Team et al., 2026) would require substantial compute resources to train at that scale. We expect our findings to transfer in this extension.

7 Future Work

Several directions follow naturally from this work. An immediate next step would be to combine a frontier model with continued pre-training on domain data, which could unite the benefits of scale and domain knowledge and push performance beyond what either achieves alone. Beyond the primary setting, it would be valuable to apply the same framework to transcoders (Marks et al., 2024) and crosscoders (Lindsey et al., 2024), to test whether the advantage extends to these alternative SAE variants. Finally, evaluating ExSAL on additional specialized domains such as medicine, law, or code would establish how broadly the findings generalize.

AI USAGE STATEMENT

In this work, we used generative AI tools (Claude Sonnet 4.6, Anthropic) for aiding and polishing writing, literature retrieval and discovery, research ideation and execution, and drafting portions of this paper. We have not used generative AI tools for generating synthetic datasets, proving mathematical claims. We have reviewed all AI-assisted work: AI-suggested research directions were evaluated against existing literature, AI-assisted writing was revised and verified for correctness by the authors, and AI-drafted text was edited to ensure all claims are grounded in our experimental results. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, sensitive data collection, or applications with foreseeable direct harm potential. The enterprise-software corpus and the domain-adapted model trained on it are proprietary and cannot be released; all other models and corpora used are publicly available. We believe improved interpretability tooling is broadly beneficial, as it supports efforts to audit language models for unwanted or biased behavior.

REPRODUCIBILITY STATEMENT

SAE training hyperparameters and all prompts used for feature selection and description generation are provided in Appendix A. Both base models are publicly available: Gemma-3-12B-IT (Team et al., 2025) for the enterprise-software domain and Meta-Llama-3-8B-Instruct (Grattafiori et al., 2024) for the finance domain. The finance domain-adapted model Llama-Fin-8B is likewise publicly available (Ke et al., 2025), while the enterprise-software domain-adapted model and its training corpus are proprietary and cannot be released. All details required to reproduce the evaluation (metric computation, judge prompts, vector database setup, retrieval configuration, and quantile sampling procedure) are documented in the main paper and Appendix A.

REFERENCES

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy, 2018. URL https://arxiv.org/abs/1601.03764.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety – a review, 2024. URL https://arxiv.org/abs/2404.14082.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html, 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*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Haozhe Chen, Carl Vondrick, and Chengzhi Mao. Selfie: Self-interpretation of large language model embeddings, 2024. URL https://arxiv.org/abs/2403.10949.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2025. URL https://arxiv.org/abs/2402.03216.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pp. 758–759, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584836. doi: 10.1145/1571941.1572114. URL https://doi.org/10.1145/1571941.1572114.

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

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. https://transformer-circuits.pub/2021/framework/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy$_m odel/index.html$.

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

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models, 2024. URL https://arxiv.org/abs/2401.06102.

Purvi Goel, Isaac Kauvar, Nicholas L Turner, and Harish Kamath. Circuits updates – may 2026. *Transformer Circuits Thread*, 2026. https://www.transformer-circuits.pub/2026/may-update/index.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas

Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy

Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Jiaojiao Han, Wujiang Xu, Mingyu Jin, and Mengnan Du. Sage: An agentic explainer framework for interpreting sae features in language models, 2026. URL https://arxiv.org/abs/2511.20820.

Jingyi He, Haiyan Zhao, Ruxue Shi, Yanguang Liu, Xin Wang, Fei Sun, and Mengnan Du. Sae-explainer: Interpreting sae features with activation-guided preference optimization, 2026. URL https://arxiv.org/abs/2606.08496.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Zixuan Ke, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. Demystifying domain-adaptive post-training for financial llms, 2025. URL https://arxiv.org/abs/2501.04961.

Dmitrii Kharlapenko, neverix, Neel Nanda, and Arthur Conmy. Self-explaining sae features. LessWrong, 8 2024. URL https://www.lesswrong.com/posts/8ev6coxChSWcxCDy8/self-explaining-sae-features.

Laura Kopf, Philine Lou Bommer, Anna Hedström, Sebastian Lapuschkin, Marina M. C. Höhne, and Kirill Bykov. Cosy: Evaluating textual explanations of neurons, 2024. URL https://arxiv.org/abs/2405.20331.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

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, 2024. URL https://arxiv.org/abs/2408.05147.

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

Samuel Marks, Adam Karvonen, and Aaron Mueller. dictionary$_{l}earning.$, 2024.

Callum Stuart McDougall, Arthur Conmy, János Kramár, Tom Lieberum, Senthooran Rajamanoharan, Neel Nanda, and Google. Gemma scope 2 - technical paper. 2025. URL https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/gemma-scope-2-helping-the-ai-safety-community-deepen-understanding-of-complex-language-model-behavior/Gemma_Scope_2_Technical_Paper.pdf.

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.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023. URL https://arxiv.org/abs/2301.05217.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. Llm evaluators recognize and favor their own generations, 2024. URL https://arxiv.org/abs/2404.13076.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models, 2024. URL https://arxiv.org/abs/2311.03658.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models, 2025. URL https://arxiv.org/abs/2410.13928.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

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

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

Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. A multimodal automated interpretability agent, 2025. URL https://arxiv.org/abs/2404.14394.

Elana Simon and James Zou. Interplm: Discovering interpretable features in protein language models via sparse autoencoders, 2024. URL https://arxiv.org/abs/2412.12101.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an open corpus of three trillion tokens for language model pretraining research, 2024. URL https://arxiv.org/abs/2402.00159.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica

Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Adly Templeton, Joshua Batson, Adam Jermyn, and Chris Olah. Circuits updates - january 2024: Predicting future activations. Transformer Circuits Thread, 2024a. URL https://transformer-circuits.pub/2024/jan-update/index.html#predict-future.

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

Xu Wang, Yan Hu, Benyou Wang, and Difan Zou. Does higher interpretability imply better utility? a pairwise analysis on sparse autoencoders, 2025. URL https://arxiv.org/abs/2510.03659.

A APPENDIX

A.1 HYPERPARAMETERS

A.1.1 SAE TRAINING

Table 3 lists all hyperparameters used to train the JumpReLU SAE for both domains.

Table 3: SAE training hyperparameters for both domains.

Hyperparameter	Enterprise-Software	Finance
Base model	Gemma-3-12B-IT	Meta-Llama-3-8B-Instruct
CPT model	Ours	Llama-Fin-8B
Hook point	Residual stream	Residual stream
Hook layer	48	32
Input dimension d_{model}	3840	4096
Expansion factor	16	16
Number of latents n	61,440	65,536
L_0 target $L_0^\star$	150	150
Sparsity λ	1×10^{-4}	1×10^{-4}
Training epochs	10	10
Batch size	4096 tokens	4096 tokens
Adam optimizer β_1	0.0	0.0
Adam optimizer β_2	0.999	0.999
Learning rate (weights)	1×10^{-3}	1×10^{-3}
Learning rate (θ)	1×10^{-3}	1×10^{-3}
KDE bandwidth ε	0.001	0.001
θ initialisation	0.001	0.001
λ warmup type	Linear	Linear
λ warmup duration	10% of steps	10% of steps
LR schedule	Cosine annealing	Cosine annealing
Activation normalisation	unit mean-squared-norm	unit mean-squared-norm

Table 4 lists the hyperparameters governing the description generation and evaluation pipeline.

Table 4: Explainer and evaluation run hyperparameters, identical across both domains.

Hyperparameter	Value
Quantile samples per feature (q)	3
Description samples per quantile (d)	3
Judge calls per description (j)	3
Number of quantiles	5
Explainer tokens per quantile	20
Held-out tokens per quantile	5
Generated sentences per judge call	20
LLM sampling temperature	0.7
Context window (tokens per side)	10
RAG top-k	5
RAG prefetch per query (dense branch)	15
RAG prefetch per query (sparse branch)	15

A.1.2 FULL RESULTS

Table 5: Full results for the enterprise-software domain, θ_{domain} judge (50 features, 95% CIs).

	Simulation Score ↑	Detection AUROC ↑	Fuzzing Accuracy ↑
BLANK	0.095 ± 0.007	0.559 ± 0.007	0.538 ± 0.004
BASE	0.290 ± 0.006	0.809 ± 0.006	0.552 ± 0.005
BASE + RAG	0.302 ± 0.006	0.794 ± 0.006	0.547 ± 0.005
EXSAL (Ours)	0.317 ± 0.005	0.822 ± 0.006	0.569 ± 0.005
EXSAL (Ours) + RAG	0.298 ± 0.007	0.800 ± 0.006	0.521 ± 0.005
FRONTIER	0.351 ± 0.005	0.863 ± 0.005	0.578 ± 0.005
FRONTIER + RAG	0.349 ± 0.006	0.857 ± 0.005	0.579 ± 0.005

Table 6: Full results for the enterprise-software domain, θ_{base} + RAG judge (50 features, 95% CIs).

	Simulation Score ↑	Detection AUROC ↑	Fuzzing Accuracy ↑
BLANK	0.190 ± 0.010	0.602 ± 0.008	0.500 ± 0.006
BASE	0.319 ± 0.006	0.736 ± 0.007	0.533 ± 0.005
BASE + RAG	0.310 ± 0.006	0.707 ± 0.007	0.508 ± 0.005
EXSAL (Ours)	0.344 ± 0.006	0.791 ± 0.007	0.536 ± 0.005
EXSAL (Ours) + RAG	0.305 ± 0.006	0.740 ± 0.007	0.502 ± 0.005
FRONTIER	0.388 ± 0.006	0.809 ± 0.006	0.556 ± 0.005
FRONTIER + RAG	0.383 ± 0.006	0.795 ± 0.006	0.554 ± 0.005

Table 7: Full results for the finance domain, θ_{domain} judge (50 features, 95% CIs).

	Simulation Score ↑	Detection AUROC ↑	Fuzzing Accuracy ↑
BLANK	0.111 ± 0.008	0.496 ± 0.008	0.697 ± 0.004
BASE	0.426 ± 0.007	0.923 ± 0.004	0.747 ± 0.005
BASE + RAG	0.428 ± 0.007	0.930 ± 0.003	0.758 ± 0.005
EXSAL (Ours)	0.456 ± 0.007	0.935 ± 0.003	0.734 ± 0.004
EXSAL (Ours) + RAG	0.451 ± 0.007	0.937 ± 0.003	0.736 ± 0.005
FRONTIER	0.471 ± 0.007	0.954 ± 0.002	0.749 ± 0.004
FRONTIER + RAG	0.473 ± 0.007	0.953 ± 0.003	0.748 ± 0.004

Table 8: Full results for the finance domain, θ_{base} + RAG judge (50 features, 95% CIs).

	Simulation Score ↑	Detection AUROC ↑	Fuzzing Accuracy ↑
BLANK	0.050 ± 0.009	0.486 ± 0.008	0.669 ± 0.005
BASE	0.385 ± 0.006	0.905 ± 0.004	0.734 ± 0.005
BASE + RAG	0.386 ± 0.006	0.909 ± 0.004	0.744 ± 0.005
EXSAL (Ours)	0.432 ± 0.006	0.913 ± 0.004	0.713 ± 0.005
EXSAL (Ours) + RAG	0.420 ± 0.006	0.915 ± 0.004	0.721 ± 0.005
FRONTIER	0.439 ± 0.006	0.933 ± 0.003	0.749 ± 0.005
FRONTIER + RAG	0.432 ± 0.006	0.929 ± 0.003	0.750 ± 0.005

A.2 ACTIVATION NORMALISATION

Raw residual-stream norms vary substantially across tokens, which would let a few high-norm tokens dominate the reconstruction loss. Following the Gemma Scope recipe (Lieberum et al., 2024), we therefore rescale activations by a single global scalar estimated as the root of the mean squared

norm over a sample of tokens, and we initialise $\boldsymbol{b}_{\text{pre}}$ to the dataset mean so that the encoder input is approximately zero-centerd. The thresholds $\boldsymbol{\theta}$ are initialised data-dependently to a per-latent percentile such that roughly $L_0^\star$ latents fire on average, giving the sparsity penalty a well-conditioned starting point.

A.3 Score Aggregation and Confidence Intervals

Each table cell in the paper reports, for a fixed combination of explainer configuration (e.g. BASE), judge (e.g. θ_{domain}), and metric (e.g. Simulation Score), a mean and a 95% confidence interval. This section explains how both are computed.

Per-feature records. For a given feature f, the pipeline runs a three-stage branching process with $q = 3$ independent quantile samples, $d = 3$ descriptions per sample, and $j = 3$ judge calls per description, producing $r = q \cdot d \cdot j = 27$ raw values of that metric, $m_{f,1}, \ldots, m_{f,r}$.

Bootstrap procedure. We repeat the following for $B = 10{,}000$ iterations. In iteration b, we resample r values with replacement from $m_{f,1}, \ldots, m_{f,r}$ for each of the $n = 50$ features and compute the per-feature mean:

$$s_f^{(b)} \;=\; \frac{1}{r}\sum_{i=1}^{r} m_{f,i}^{(b)}. \tag{4}$$

We then average across features to obtain the cross-feature mean for that iteration:

$$\bar{s}^{(b)} \;=\; \frac{1}{n}\sum_{f=1}^{n} s_f^{(b)}. \tag{5}$$

After B iterations we have a distribution $\{\bar{s}^{(b)}\}_{b=1}^{B}$. The reported mean is the average of this distribution and the 95% confidence interval is its 2.5th and 97.5th percentiles:

$$\mathrm{CI}_{95} \;=\; \big[\hat{q}_{0.025},\ \hat{q}_{0.975}\big]. \tag{6}$$

A.4 Vector Database Setup

The RAG variants (BASE + RAG, EXSAL + RAG, FRONTIER + RAG) retrieve domain-relevant context at inference time from a vector database built over the same corpus used for continued pretraining. We use Qdrant as the vector store.

Indexing. The enterprise-software corpus is chunked into passages of at most 512 tokens with a 64-token overlap between adjacent chunks. Each passage is embedded using `BAAI/bge-m3` (Chen et al., 2025), a multilingual retrieval model that jointly produces a dense 1024-dimensional embedding and sparse lexical weights over a 250,002-dimensional vocabulary. Indexing is performed on the CPU in full precision, using a batch size of 8,192. We maintain two separate Qdrant collections: the enterprise-software collection contains 3,876,430 chunks and the finance collection contains 12,459,572 chunks.

Retrieval. BGE-M3 produces both a dense 1024-dimensional embedding and sparse lexical weights for the query and each indexed chunk. We retrieve the top $k = 5$ chunks using hybrid retrieval: the 15 highest-scoring candidates from the dense branch and the 15 highest-scoring from the sparse branch are merged via Reciprocal Rank Fusion (Cormack et al., 2009), and the top 5 of the merged list are prepended to the explainer prompt as additional domain context.

A.5 Quantile Sampling and Explanation

For each selected feature, we stratify the activating tokens into quantiles and sample from each, following Paulo et al. (2025), who show that sampling only from the top activations performs worse than quantile sampling.

Rank-based normalisation. We stratify by rank rather than by activation value. The activation distribution of an SAE latent is heavy-tailed: a handful of top tokens often activate at values many times larger than the bulk, so value-based binning would compress almost all tokens into a narrow low range and place nearly everything in a single bin. We therefore sort a feature's activating tokens by activation and cut the ranking into the strata 0–5%, 5–15%, 15–30%, 30–60%, and 60–100% (strongest to weakest), plus a "zero" stratum of non-activating positions used later for evaluation. Rank-based cuts are invariant to the scale and shape of the activation distribution and guarantee that each stratum is populated, so the explainer sees the full range of the feature's behavior rather than only its outliers.

Explainer prompting. From each of the five activating strata we sample 20 tokens (100 in total) and present them to the explainer, each labeled by its quantile and shown with a context window of ± 10 tokens around it. These tokens, together with their quantile labels and context windows, are then passed to the explainer, which is instructed to summarize the semantic or domain concept unifying the tokens. The explainer returns a single concise natural-language description per feature. The exact prompt is reproduced in Appendix A.

A.6 Domain-Adaptation

A.6.1 General-Capability Control

To confirm that ExSAL's advantage does not stem from θ_{domain} being a stronger general model, we compare both explainers on MMLU. θ_{domain} scores slightly lower than θ_{base} (Table 9). Since general capability favors θ_{base}, ExSAL's advantage cannot be attributed to a stronger general model and instead reflects the domain knowledge acquired during CPT.

Model	MMLU (0-shot)
θ_{base} (Gemma 3 12B IT)	0.715
θ_{domain} (Gemma 3 12B CPT + IFT)	0.685

Table 9: θ_{domain} scores 3.0 points below θ_{base} on MMLU: ExSAL's explainer is the weaker general model yet still outperforms Base, isolating domain knowledge as the source of its advantage.

A.6.2 Enterprise Software

We perform continued pre-training (CPT) then instruction fine-tuning (IFT) on the Gemma-3-12B base model (Team et al., 2025). We use an internal corpus of documentation, product guides, tables, API metadata, and knowledge-graph verbalizations. We use the Dolma tool for data cleaning (Soldaini et al., 2024), resulting in $\sim$1.3B tokens for CPT. After the CPT stage, we perform IFT where the model is trained to follow instructions and respond to queries. We use the English-filtered Tulu-3 dataset(Lambert et al., 2025), resulting in $\sim$650K instruction examples.

We train using 8 NVIDIA H100s and report the hyperparameters for each stage of the training process below. We use mixed-precision training and select the epoch with the lowest evaluation loss. In CPT, we train on contiguous chunks the size of the block size (8192 tokens), while in IFT we pack samples up to the same block size.

A.6.3 Finance

We do not perform the domain-adaptation ourselves for the finance domain; we use the off-the-shelf domain-adapted model from Ke et al. (2025). This training setup performs domain adaptation through joint training on domain-specific pretraining data and instruction-tuning data, rather than sequentially applying CPT and IFT. Datasets used, token counts, hyperparameters, and other training details are reported in Ke et al. (2025).

Table 10: Training hyperparameters for the continued pre-training (CPT) and instruction fine-tuning (IFT) stages for Enterprise Software Domain Adaptation.

Hyperparameter	**CPT**	**IFT**
Block size (sequence length)	8192	8192
Epochs	3	3
Optimizer	AdamW	AdamW
Learning rate	3×10^{-5}	3×10^{-5}
LR scheduler	linear	linear
Warmup ratio	0.01	0.20
Weight decay	0.1	0.1
Max gradient norm	1.0	1.0
Per-device train batch size	2	1
Gradient accumulation steps	128	8
Effective batch size	2048	64

A.7 METRICS

All three metrics require an LLM judge model. In our main experiments we use θ_{domain}. To test whether this choice introduces a bias in favour of ExSAL descriptions, we additionally evaluate with θ_{base} + RAG as an independent judge in RQ4 (Section 3).

Simulation Score Introduced by Bills et al. (2023), Simulation Score tests how well a description lets a judge predict how strongly a token activates the feature. For each held-out token, the judge is prompted to rate expected activation strength on a scale of 0 to 9. Rather than parsing the digit output directly, we read the log-probabilities of the ten digit tokens, apply a softmax, and compute the expected value $\sum_{i=0}^{9} i \cdot P(i)$. Simulation Score is the Pearson correlation between these soft predictions and the true normalized activation values, and ranges from -1 to $+1$.

Detection AUROC Proposed by Paulo et al. (2025), detection asks the judge to classify each held-out token as activating or non-activating given the description, answering Y or N. We extract the log-probabilities of Y and N, apply a two-way softmax to get $p_{\text{yes}} \in [0, 1]$, and compute the AUROC over activating and non-activating tokens, which ranges from 0 to 1. Because detection operates at the sequence level, the judge does not need to localise the feature to a specific token, it only needs to recognize that the context is the kind of context where the feature fires.

Fuzzing Accuracy Also from Paulo et al. (2025), fuzzing operates at the token level. Held-out tokens are presented to the judge in batches of five for efficiency, each highlighted as `<<token>>` within its surrounding context window of ± 10 tokens, and the judge outputs a binary decision for each token indicating whether it activates the feature. Unlike the other two metrics, fuzzing parses the judge's text output directly rather than reading logits. Fuzzing Accuracy is the fraction of correct binary predictions and ranges from 0 to 1.

A.8 RESULTS ON RANDOMLY SAMPLED FEATURES

The main experiments evaluate the 50 features selected by the two-stage LLM-as-judge filter described in Section 2.3. To quantify the effect of this selection, we repeat the evaluation on a set of 50 randomly sampled features from the same enterprise-software SAE. The only requirement for inclusion is a minimum of 130 activating tokens, matching the exact token budget of the main pipeline to ensure a fair, cost-consistent comparison.

Tables 11–13 show filtered and random results side by side. In some cases the score collapses to values that are 8 times lower. This collapse is consistent across every explainer configuration and both judges. Inspecting the features makes clear why, as most randomly drawn features fall into one of three categories.

Polysemantic features have learned to respond to multiple unrelated concepts simultaneously. One example from the random set activates across human anatomy, sports commentary, software integration, and taxation.

Out-of-domain features activate primarily on general English text rather than domain-specific content. A feature from the random set, for instance, fires on sports commentary, medical research, and mixed general tokens with no connection to the enterprise-software domain.

Syntactic features respond to surface-level patterns such as punctuation or common function words rather than semantic content. Periods, forward slashes, and equal signs can be enough to trigger activation, with no coherent concept underlying the pattern. While such tokens naturally appear in domain-adapted text, the feature captures nothing domain-specific and is of no interest for interpretability.

None of these are the features one would want to inspect when analyzing a domain-adapted LLM. Including them blurs the comparison between explainer configurations and obscures the signal that domain knowledge in the explainer actually provides.

Table 11: Simulation Score (↑): filtered vs. randomly sampled features, enterprise-software domain, both judges (95% CIs).

Simulation Score	θ_{domain} judge		θ_{base} + RAG Judge	
	Filtered	Random	Filtered	Random
BASE	0.290 ± 0.006	0.084 ± 0.008	0.319 ± 0.006	0.063 ± 0.009
BASE + RAG	0.302 ± 0.006	0.036 ± 0.009	0.310 ± 0.006	0.015 ± 0.009
EXSAL (Ours)	0.317 ± 0.005	0.095 ± 0.008	0.344 ± 0.006	0.069 ± 0.009
EXSAL (Ours) + RAG	0.298 ± 0.007	0.019 ± 0.008	0.305 ± 0.006	-0.014 ± 0.009
FRONTIER	0.351 ± 0.005	0.078 ± 0.008	0.388 ± 0.006	0.038 ± 0.009
FRONTIER + RAG	0.349 ± 0.006	0.091 ± 0.008	0.383 ± 0.006	0.060 ± 0.009

Table 12: Detection AUROC (↑): filtered vs. randomly sampled features, enterprise-software domain, both judges (95% CIs).

Detection AUROC	θ_{domain} judge		θ_{base} + RAG Judge	
	Filtered	Random	Filtered	Random
BASE	0.809 ± 0.006	0.524 ± 0.007	0.736 ± 0.007	0.518 ± 0.008
BASE + RAG	0.794 ± 0.006	0.472 ± 0.007	0.707 ± 0.007	0.474 ± 0.008
EXSAL (Ours)	0.822 ± 0.006	0.508 ± 0.007	0.791 ± 0.007	0.512 ± 0.008
EXSAL (Ours) + RAG	0.800 ± 0.006	0.457 ± 0.007	0.740 ± 0.007	0.466 ± 0.008
FRONTIER	0.863 ± 0.005	0.529 ± 0.008	0.809 ± 0.006	0.508 ± 0.008
FRONTIER + RAG	0.857 ± 0.005	0.552 ± 0.007	0.795 ± 0.006	0.531 ± 0.007

Table 13: Fuzzing Accuracy (↑): filtered vs. randomly sampled features, enterprise-software domain, both judges (95% CIs).

Fuzzing Accuracy	θ_{domain} judge		θ_{base} + RAG	
	Filtered	Random	Filtered	Random
BASE	0.552 ± 0.005	0.414 ± 0.005	0.533 ± 0.005	0.375 ± 0.006
BASE + RAG	0.547 ± 0.005	0.354 ± 0.004	0.508 ± 0.005	0.357 ± 0.005
EXSAL (Ours)	0.569 ± 0.005	0.417 ± 0.006	0.536 ± 0.005	0.368 ± 0.006
EXSAL (Ours) + RAG	0.521 ± 0.005	0.349 ± 0.004	0.502 ± 0.005	0.352 ± 0.005
FRONTIER	0.578 ± 0.005	0.459 ± 0.005	0.556 ± 0.005	0.388 ± 0.006
FRONTIER + RAG	0.579 ± 0.005	0.453 ± 0.005	0.554 ± 0.005	0.387 ± 0.006

A.9 LICENSES

Models. Gemma 3 is released under the Gemma Terms of Use (https://ai.google.dev/gemma/terms).
Llama 3 is released under the Meta Llama 3 Community License (https://huggingface.co/meta-llama/Meta-Llama-3-8B/blob/main/LICENSE).
Llama-Fin-8b (Salesforce FinDAP) is released under the Creative Commons Attribution 4.0 International License (https://huggingface.co/Salesforce/Llama-Fin-8b).
BGE-M3 is released under the MIT License (https://huggingface.co/BAAI/bge-m3).

Libraries. Qdrant is licensed under the Apache License, Version 2.0 (https://github.com/qdrant/qdrant/blob/master/LICENSE).

Datasets. FineWeb is released under the Open Data Commons Attribution License (ODC-By) v1.0. Use of this dataset is additionally subject to CommonCrawl's Terms of Use.
FinTrain (Salesforce FinDAP) is licensed under the Apache License, Version 2.0 (https://github.com/SalesforceAIResearch/FinDap/blob/main/LICENSE.txt).

A.10 COMPUTE

All experiments across both the enterprise-software and finance domains were conducted on a single NVIDIA A10G GPU (24 GB VRAM) for SAE training and 4 NVIDIA A10G GPUs (96 GB VRAM total) per explainer model for evaluation.

A.11 PROMPTS

This section lists all prompts used in the pipeline. The placeholder [domain] refers to the target domain (e.g. enterprise-software or finance) and is substituted at runtime. In the feature scoring prompts, the label [EN] refers to content from FineWeb (Penedo et al., 2024), the general English web corpus used in SAE training.

A.11.1 FEATURE SCORING PROMPTS

These prompts are used in the first stage of the two-stage LLM-as-judge filter to score each SAE feature for domain relevance.

System Prompt

> You are an expert in neural network interpretability analyzing SAE (Sparse Autoencoder) features from a language model trained on [domain] documentation and general English text. You always respond with valid JSON only --- no prose, no markdown, no explanation outside the JSON object.
>
> You will be shown activating tokens for a single SAE feature, organized into activation-strength quantiles (0--10% = the strongest activations, 90--100% = the weakest). Each token is labeled [domain] (domain text) or [EN] (general English text), with its activation strength and surrounding context.
>
> Your task is to score this feature on how useful it is for understanding what the model learned about [domain] domain knowledge. A high score means: ``this feature captures something meaningful and [domain]-specific.'' A low score means: ``this feature is generic, domain-agnostic, or fires on structural noise rather than [domain] content.''
>
> Score the feature on these three criteria (output each as a separate integer):
>
> **domain_purity (0--4):** How exclusively does this feature fire on [domain] domain tokens? 0 = fires predominantly on [EN] tokens, or roughly equally on both. 1 = fires on both but noticeably more on [domain]. 2 = fires mostly on

[domain] with occasional general English tokens. 3 = fires almost exclusively on [domain], only rare [EN] exceptions. 4 = fires ONLY on [domain] tokens.

content_richness (0--4): Does this feature capture meaningful content rather than structural/surface tokens? 0 = purely structural noise --- BOS, EOS, padding, whitespace, newlines only. 1 = almost entirely punctuation or single characters with no semantic content. 2 = mostly structural with some content words mixed in. 3 = mostly meaningful content --- recognizable words, topics, or concepts dominate. 4 = rich, specific semantic content --- a clear [domain] topic cluster or domain concept.

specificity (0--4): Can you form a clear, specific interpretation of what this feature detects? 0 = no clear pattern, random mix of unrelated tokens with no obvious theme. 1 = vague pattern --- you can tell it is ``[domain]-related'' but nothing more specific. 2 = clear pattern but broad (e.g. ``[domain] configuration terms''). 3 = precise, specific pattern with clear [domain] context. 4 = highly precise pattern with unambiguous [domain] context and strong semantic coherence.

judge_score = domain_purity + content_richness + specificity (range: 0 to 12)

Output ONLY this JSON object, nothing else: {"label": "<short label max 8 words>", "explanation": "<2--3 sentences>", "domain_label": "<[domain]|ENG|BOTH>", "domain_purity": <0--4>, "content_richness": <0--4>, "specificity": <0--4>, "judge_score": <0--12>}

User Prompt

Activating tokens organized by activation-strength quantile (0--10% = strongest, 90--100% = weakest).
domain
= token from [domain] domain text, [EN] = token from general English text:

{quantile_blocks}

Score this feature and return only valid JSON.

A.11.2 Feature Selection Prompts (Stage 2)

The second stage receives the top-100 candidates from Stage 1 and selects the final 50 features. Two domain-specific variants are used.

Enterprise-Software Domain

You are given a CSV of [domain] Sparse Autoencoder (SAE) features. Each row has: feature_idx, judge_label (natural-language description), domain_label, n_active, judge_score (1--12).

Select exactly 50 features that represent specific, narrow [domain] domain concepts --- features that fire on a well-defined semantic topic rather than broad or structural content.

Prefer features where: judge_label describes a specific [domain] domain such as HR/payroll compliance, GDPR/data governance, data archiving, regulatory/statutory content, or integration workflows; the feature fires on relatively few distinct tokens.

Avoid features where: judge_label describes syntactic or structural content rather than semantic meaning, e.g. formatting tokens, encoded strings, UI navigation elements, or low-information noise.

```
Return only a comma-separated list of feature_idx values.
```

Finance Domain

```
You are given a CSV of Fintech Sparse Autoencoder (SAE)
features.  Each row has:  feature_idx, judge_label
(natural-language description), domain_label, n_active,
judge_score (1--12).
Select exactly 50 features that represent specific,
narrow Fintech domain concepts --- features that fire on a
well-defined semantic topic rather than broad or structural
content.
Prefer features where:  judge_label describes a specific
Fintech domain such as securities regulation/compliance,
monetary policy/central banking, financial market structure,
banking/credit/lending, accounting standards/audit,
investment funds/asset management, or systemic
risk/financial stability; the feature fires on relatively
few distinct tokens.
Avoid features where:  judge_label describes syntactic
or structural content rather than semantic meaning, e.g.
footnote citation numbers, section/document formatting
tokens, numeric placeholders, or low-information noise.
Return only a comma-separated list of feature_idx values.
```

A.11.3 EXPLAINER PROMPTS

These prompts are sent to the explainer model (e.g. BASE, EXSAL, or FRONTIER) to produce natural-language feature descriptions. The RAG variant prepends retrieved passages to the user prompt.

System Prompt (all configurations)

```
You are an expert in neural network interpretability
analyzing SAE (Sparse Autoencoder) features from a language
model.  You will be shown activating tokens for a single
SAE feature, organized by activation strength quantile
(strongest to weakest).  Your task is to write a concise
natural-language description of what this feature detects:
what semantic concept or domain-specific content causes it
to activate.  IMPORTANT: Focus exclusively on semantic and
conceptual patterns --- the underlying topic, domain, or
meaning that unifies the activating tokens.  Do NOT describe
syntactic or surface-level patterns such as punctuation
marks, isolated digits, or whitespace, even if these appear
as high-activation tokens.  If the top-quantile tokens
appear syntactic, look at the surrounding context and the
full distribution of tokens across all quantiles to identify
the underlying semantic concept.  Pay attention to the
quantile labels --- tokens in '0--5%' are the strongest
activators, tokens in '60--100%' activate weakly.  Be
specific and grounded in the examples shown.  Output ONLY
the description --- no preamble, no JSON, no bullet points.
[RAG only] If a <retrieved_context> block is provided in
the user message, you may use it as additional background
information about the domain.  This context contains
passages from the training corpus related to the feature's
activating tokens.  However, do not let the retrieved
context override the evidence from the activating tokens
themselves.
```

User Prompt (without RAG)

```
Activating tokens organized by quantile (0--5% = strongest,
60--100% = weakest).
Format: [quantile] token=<token> | context: <surrounding
text>
{token_block}
Write a concise description of what this SAE feature
detects.
```

User Prompt (with RAG)

```
<retrieved_context>
[Passage 1]: {chunk_text_1}
[Passage 2]: {chunk_text_2}
...
[Passage N]: {chunk_text_N}
</retrieved_context>
Activating tokens organized by quantile (0--5% = strongest,
60--100% = weakest).
Format: [quantile] token=<token> | context: <surrounding
text>
{token_block}
Write a concise description of what this SAE feature
detects.
```

A.11.4 Evaluation Prompts

All three evaluation metrics use a judge model. In the primary setting, θ_{domain} serves as judge and the prompts below are used as-is. When θ_{base} + RAG serves as judge (RQ4), the following `<retrieved_context>` block is prepended to each user prompt before the feature description:

```
<retrieved_context>
[Passage 1]: {chunk_text_1}
[Passage 2]: {chunk_text_2}
...
[Passage N]: {chunk_text_N}
</retrieved_context>
```

Simulation Score *System Prompt:*

```
You are an expert in neural network interpretability
analyzing SAE (Sparse Autoencoder) features. You will be
given a description of what an SAE feature detects, and a
single token with its surrounding context. Your task is to
predict how strongly this feature activates on that token,
on a scale from 0 to 9 (0 = no activation, 9 = maximum
activation). Output ONLY a single digit between 0 and 9.
No explanation, no punctuation.
```

User Prompt:

```
Feature description:
{description}
Token: {token}
Context: {context}
How strongly does this feature activate on this token?
Answer with a single digit 0--9.
```

Detection AUROC *System Prompt:*

```
You are an expert in neural network interpretability
analyzing SAE (Sparse Autoencoder) features. You will be
given a description of what an SAE feature detects, and a
```

```
text context.  Your task is to decide whether this feature
is active in the given context.  Output ONLY 'Y' (yes, the
feature is active) or 'N' (no, the feature is not active).
No explanation, no punctuation, just Y or N.
```

User Prompt:

```
Feature description:
{description}
Context:  {context}
Is this feature active in the context above?  Answer Y or N.
```

Fuzzing Accuracy *System Prompt:*

```
You are an intelligent and meticulous linguistics
researcher.  You will be given a certain feature of text,
such as 'male pronouns' or 'text with negative sentiment'.
You will be given a few examples of text that contain
this feature.  Portions of the sentence which strongly
represent this feature are between tokens << and >>.  Some
examples might be mislabeled.  Your task is to determine if
every single token within << and >> is correctly labeled.
Consider that all provided examples could be correct, none
could be correct, or a mix.  An example is only correct
if every marked token is representative of the feature.
For each example in turn, return 1 if the sentence is
correctly labeled or 0 if the tokens are mislabeled.  You
must return your response in a valid Python list.  Do not
return anything else besides a Python list.
```

User Prompt:

```
feature interpretation:  {description}
Text examples:
{examples}
```