Towards Interpretable Scientific Foundation Models: Sparse Autoencoders for Disentangling Dense Embeddings of Scientific Concepts

Anonymous Author(s)
Affiliation
Address
email

Abstract

The prevalence of foundation models in scientific applications motivates the need for interpretable representations and search of scientific concepts. In this work, we present a novel approach using sparse autoencoders (SAEs) to disentangle dense embeddings from large language models, offering a pathway towards more interpretable scientific foundation models. By training SAEs on embeddings of over 425,000 scientific paper abstracts spanning computer science and astronomy, we demonstrate their effectiveness in extracting interpretable features while maintaining semantic fidelity. Our method reveals and analyzes SAE features that directly correspond to scientific concepts, and introduces a novel method for identifying 'families' of related concepts at varying levels of abstraction. To illustrate the practical utility of our approach, we demonstrate how interpretable features from SAEs can precisely steer semantic search over scientific literature, allowing for fine-grained control over query semantics. This work not only bridges the gap between the semantic richness of dense embeddings and the interpretability needed for scientific applications, but also offers new directions for improving literature review and scientific discovery. For use by the scientific community, we open-source our embeddings, trained sparse autoencoders, and interpreted features, along with a web app for interactive literature search.

1 Introduction

Foundation models have revolutionised natural language processing and are increasingly impacting scientific domains, enabling the representation of complex scientific concepts in rich semantic spaces (Devlin et al., 2018; Brown et al., 2020). Dense neural vector embeddings capture nuanced semantic relationships, enhancing downstream applications such as scientific information retrieval (IR) and semantic search (Reimers et al., 2019; Gao et al., 2022; Wang et al., 2024). However, the power of these representations comes at a cost: reduced interpretability and limited user control (Cao et al., 2023a). In scientific applications, where explainability is critical, these challenges pose a barrier to embeddings-based tools for literature reviews and scientific discovery.

To address these limitations, recent research has explored methods to disentangle and interpret the information encoded in dense representations (Trifonov et al., 2018). Sparse autoencoders (SAEs) have emerged as a promising solution for extracting interpretable features from high-dimensional representations (Ng et al., 2011; Makhzani et al., 2013). By learning to reconstruct inputs as linear combinations of features in a higher-dimensional sparse basis, SAEs can disentangle complex representations into individually interpretable components. This approach has shown success in analysing and steering generative models (Conmy et al., 2024; Lee, 2024; Cunningham et al., 2023b), but its application to dense text embeddings remains unexplored.

Submitted to Foundation Models for Science Workshop, 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

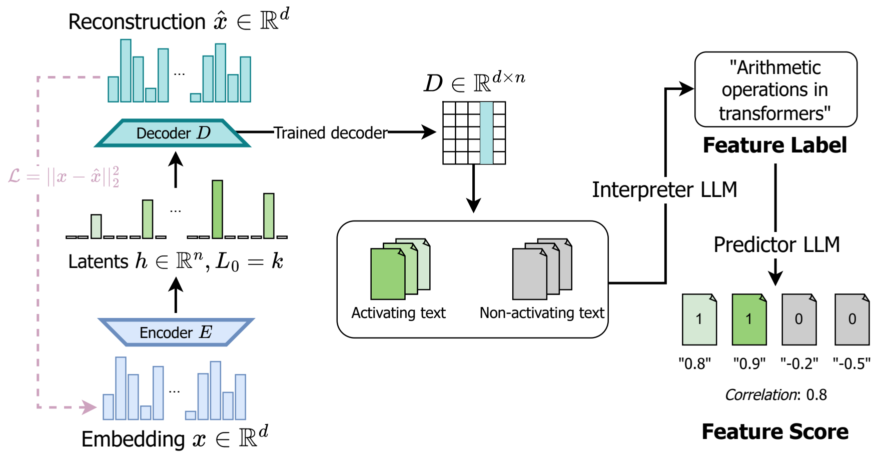


Figure 1: Training and feature labelling process for our sparse autoencoder (SAE). The SAE is trained to minimise reconstruction loss on embeddings from astronomy and computer science paper abstracts. Each feature corresponds to a column in the decoder matrix, representing a direction in embedding space. Feature interpretation involves two steps: (1) An *Interpreter* language model identifies topics present in text that activates each feature but absent in non-activating text. (2) A separate *Predictor* language model assesses feature interpretability by stating its confidence that the feature will activate on unseen text, with confidence correlated with ground truth activations to quantify interpretability.

In this work, we present the first application of SAEs to dense text embeddings derived from language foundation models, focusing on scientific literature. We demonstrate that this approach offers new possibilities for searching, understanding, and manipulating scientific concept spaces. By applying our method to embeddings from two diverse scientific domains - computer science and astronomy - we showcase its potential for cross-domain applicability in scientific AI. In a direct demonstration of their utility for scientific exploration, we show how SAE features can be used to steer scientific literature search results, building on previous work applying similar techniques to decoder-only transformers and diffusion models for guided generation (Elhage et al., 2022b; Daujotas, 2024). By causally manipulating features in the SAE's hidden representation of an embedding vector, we can perform precise adjustments of the semantic meaning of scientific concepts upon reconstruction.

Our research makes the following key contributions towards more interpretable scientific foundation models. We train SAEs with varying sizes on embeddings from a large corpus of scientific papers across two domains, demonstrating their effectiveness in learning interpretable document-level features from dense representations of scientific text. We conduct a comprehensive analysis of the learned features through the lens of scientific concepts, examining their interpretability, behaviour across different model capacities, and semantic properties. To extend this analysis, we introduce the concept of SAE "feature families", clusters of related features that allow for multi-scale analysis and manipulation of scientific concepts, and examine how features "split" across levels of abstraction. Finally, we demonstrate the practical utility of our approach by applying these interpretable features to enhance scientific semantic search, allowing for fine-grained control over query semantics in scientific literature exploration. We develop and open-source this as a tool that implements our SAE-enhanced semantic search system for scientific literature, as well as open-sourcing the underlying SAEs.

2 Background and Related work

While dense embeddings have dramatically improved performance across various NLP tasks, they present significant challenges in terms of interpretability.

2.1 Embeddings and Representation Learning

The evolution of word representations in NLP has progressed from simple one-hot encodings to sophisticated dense vector embeddings, culminating in contextual models like BERT (Devlin et al.,

2018) and sentence-level embeddings like Sentence-BERT (Reimers et al., 2019). While these dense embeddings have significantly improved NLP performance, particularly in semantic search and information retrieval (Gao et al., 2021), they present challenges in interpretability and fine-grained control due to their high-dimensional, continuous nature (Liu et al., 2019). This opacity is particularly problematic in applications requiring explainability or precise semantic manipulation. Moreover, dense embeddings face challenges such as the "curse of dense retrieval" (Reimers et al., 2022), where performance degrades rapidly with increasing index size, and difficulties in fine-tuning search results (Cao et al., 2023b; Turian et al., 2010).

2.2 Sparse autoencoders

In large language models, the superposition hypothesis suggests that dense neural networks are highly underparameterised, and perform computations involving many more concepts than neurons (Elhage et al., 2022a). Because these semantic concepts, or *features*, are quite sparse, models compensate encoding multiple features within the same set of neurons. However, this also leads to complex, overlapping representations that are difficult to interpret on a single-neuron basis. Similarly, in embedding spaces, features are not represented monosemantically in individual dimensions. Instead, each feature is typically distributed across multiple dimensions, and conversely, each dimension may contribute to the representation of multiple features. This distributed representation allows embedding models to efficiently encode a large number of features in a relatively low-dimensional space, but it also makes the embeddings challenging to interpret directly.

To address this challenge, sparse autoencoders (SAEs) have emerged as a promising solution. SAEs learn to reconstruct inputs using a sparse set of features in a higher-dimensional space, potentially disentangling superposed features (Elhage et al., 2022b; Olshausen et al., 1997). By encouraging this disentanglement, SAEs aim to reveal more interpretable and semantically meaningful representations, demonstrating efficacy in uncovering interpretable features in large language model activations (Donoho, 2006; Gao et al., 2024). In a well-trained SAE, individual features in the hidden dimension align with the underlying sparse, semantically meaningful features.

2.2.1 Architecture and training

Sparse autoencoders (SAEs) are neural network models designed to learn compact, interpretable representations of high-dimensional data while enforcing sparsity in the hidden layer activations. The architecture of an SAE consists of an encoder network that maps the input to a hidden representation, and a decoder network that reconstructs the input from this representation.

Let $\mathbf{x} \in \mathbb{R}^d$ be an input vector, and $\mathbf{h} \in \mathbb{R}^n$ be the hidden representation, where typically $n \gg d$. The encoder and decoder functions are defined as:

$$\text{Encoder}: \quad \mathbf{h} = f_\theta(\mathbf{x}) = \sigma(W_e\mathbf{x} + \mathbf{b}_e) \tag{1}$$

$$\text{Decoder}: \quad \hat{\mathbf{x}} = g_\phi(\mathbf{h}) = W_d\mathbf{h} + \mathbf{b}_d \tag{2}$$

where $W_e \in \mathbb{R}^{n\times d}$ and $W_d \in \mathbb{R}^{d\times n}$ are the encoding and decoding weight matrices, $\mathbf{b}_e \in \mathbb{R}^k$ and $\mathbf{b}_d \in \mathbb{R}^d$ are bias vectors, and $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU or sigmoid). The parameters $\theta = \{W_e, \mathbf{b}_e\}$ and $\phi = \{W_d, \mathbf{b}_d\}$ are learned during training.

The training objective of our SAE combines three main components: a reconstruction loss, a sparsity constraint, and an auxiliary loss. The overall loss function is given by:

$$\mathcal{L}(\theta, \phi) = \frac{1}{d}\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda\mathcal{L}_{\text{sparse}}(\mathbf{h}) + \alpha\mathcal{L}_{\text{aux}}(\mathbf{x}, \hat{\mathbf{x}})$$

where $\lambda > 0$ and $\alpha > 0$ are hyperparameters controlling the trade-off between reconstruction fidelity, sparsity, and the auxiliary loss.

For the sparsity constraint, we use a k-sparse constraint: only the k largest activations in $\mathbf{h}$ are retained, while the rest are set to zero (Makhzani et al., 2013; Gao et al., 2024). This approach avoids issues such as shrinkage, where L1 regularisation can cause feature activations to be systematically lower than their true values, potentially leading to suboptimal representations *shrinkage*, (Wright et al., 2024; Rajamanoharan et al., 2024). We also use an auxiliary loss, similar to the “ghost grads” technique (Jermyn et al., 2023), to model the reconstruction error using the top k_{aux} dead latents, where we typically set $k_{\text{aux}} = 2k$ (Gao et al., 2024); see Appendix A for details.

2.2.2 Structure in SAE features

State-of-the-art automated interpretability techniques have resulted in the discovery of a large volume of highly interpretable, monosemantic features in SAEs trained over language models (Cunningham et al., 2023a; Bricken et al., 2023). With features being the base unit of interpretability for SAEs, recent work has focused on understanding the geometric structure of features. Bricken et al. (2023) report *feature splitting* in geometrically close groups of semantically related features, where number of learned features in the cluster increases with model size. They also report the existence of *universal* features which re-occur between independent SAEs and which have highly similar activation patterns. Templeton (2024) find feature splitting also occurs in SAEs trained over production-scale models, with larger SAEs also exhibiting *novel* features for concepts that are not represented in smaller SAEs. Makelov et al., 2024 report *over-splitting* of binary features. Engels et al., 2024 find clusters of SAE features that represent inherently multi-dimensional, non-linear subspaces.

2.3 Language foundation models in science

A number of domain-specific large language models have been developed for question-answering in specific areas of science, such as medicine or astronomy (Rasmy et al., 2021; Taylor et al., 2022; Nguyen et al., 2023). Neural vector embeddings from language models have also been leveraged to enhance scientific question answering and literature search (Kinney et al., 2023; Iyer et al., 2024; Lála et al., 2023). Recent work has also demonstrated the ability of language models to complete problem-solving and knowledge synthesis tasks relevant to scientific research (AI4Science et al., 2023; Romera-Paredes et al., 2024), and even generate novel research hypotheses (Si et al., 2024). Some research has suggested that some latent scientific knowledge in models exists in structured representations, and that these representations may be leveraged for scientific discovery (Tshitoyan et al., 2019; Qu et al., 2024). However, it has also been demonstrated that language models can exhibit human-like biases or generate false information, limiting their usefulness as scientific tools (Birhane et al., 2023); here, improved interpretability and steerability could unlock capabilities.

3 Training SAEs and automated labelling

3.1 Training and automated interpretability methods

We trained top-k Sparse Autoencoders (SAEs) on embeddings of arXiv paper abstracts from astrophysics (`astro-ph`, 272,000 papers) and computer science (`cs.LG`, 153,000 papers) domains, using OpenAI's `text-embedding-3-small` model. We experimented with various hyperparameters, focusing primarily on SAEs with $k = 16$, 32, and 64 active latents. To interpret the learned features, we employed an automated two-step process using large language models: an Interpreter to generate feature labels, and a Predictor to assess interpretation confidence. We evaluated SAEs based on their reconstruction ability and feature interpretability, using metrics such as normalised mean squared error and Pearson correlation. Detailed training procedures, hyperparameters, and evaluation metrics are provided in Appendix A.

SAE Performance: We observe precise power-law scalings for sparse autoencoder (SAE) performance as a function of the number of total latents n, active latents k, and compute C used for training. The normalised mean squared error (MSE) scales as $L(n) = cn^{-\alpha}$ for fixed k, where α ranges from 0.12 to 0.18, increasing with k, while c generally decreases (Figure 2, left panel). For compute scaling, we calculate the number of training FLOPs C at each step for each SAE. We find $L(C) = aC^b$, where a generally increases with k (3.84 for $k = 16$ to 8.03 for $k = 64$) and b ranges from -0.11 to -0.16, becoming more negative as k increases from 16 to 64 (Figure 2, right panel). Both relationships show high accuracy with R-squared values above 0.93. Detailed fits are provided in Appendix A.

Interpretability of SAE features: The most direct way to evaluate the interpretability of features is to look at the distribution of automated interpretability scores. Specifically: given a feature label from our interpreter model, how well can a predictor model predict the feature's activation on unseen text? We show in Figure 3 that the Pearson correlation between predictor model confidence of a feature firing and the ground-truth firing is quite high, with median correlations ranging from 0.65 to 0.71 for `cs.LG` and 0.85 to 0.98 for `astro-ph`. We note that Pearson correlation increases as k and n

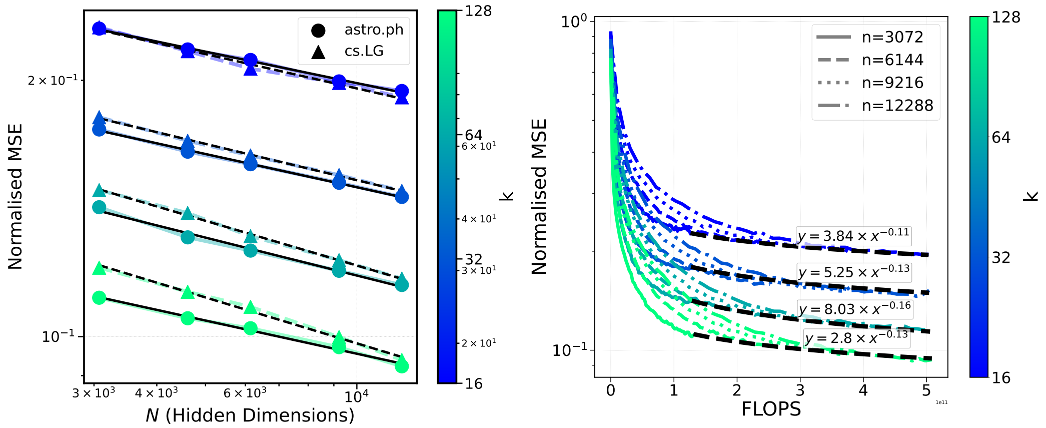


Figure 2: Scaling laws for sparse autoencoder performance. Left: Normalised mean squared error (MSE) as a function of the number of total latents n for different values of active latents k. The power-law scaling is evident for each k. Right: Reconstruction loss as a function of compute (FLOPs) for different k values, demonstrating the compute-optimal model size scaling.

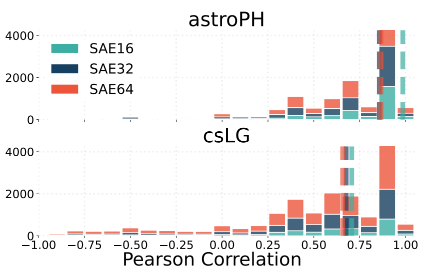


Figure 3: Pearson correlations between the ground-truth and predicted feature activation, using GPT-4o as the *Interpreter* and GPT-4o-mini as the *Predictor*.

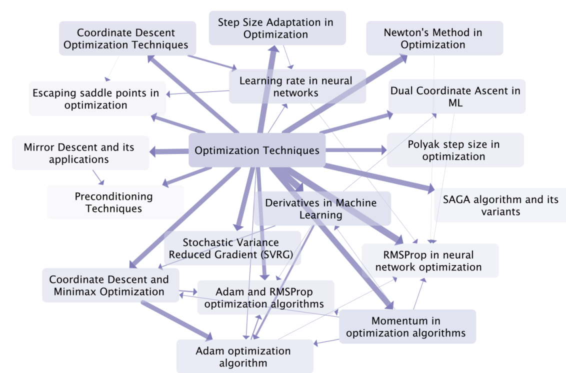


Figure 4: Sample feature family from `cs.LG`; arrows represent $C_{ij}^{norm} > 0.1$, with size $\propto C_{ij}^{norm}$.

decrease, likely due to models learning coarser-grained features that are easier for the interpreter to identify.

4 Constructing feature families through graph-based clustering

We find that our SAEs trained over arXiv paper embeddings recover a wide range of scientifically relevant features. These features cover both scientific concepts, from niche to broad and multi-disciplinary, and also more abstract semantic artifacts, such as humorous writing or critiques of scientific theories. Features and activating examples can be found in Appendix C. In the remainder of this work, we focus primarily on features that correspond directly to scientific concepts from the literature. Our analysis of feature evolution and grouping provides insights into how scientific concepts are represented and related within foundation models, potentially informing the development of more interpretable and efficient scientific AI systems.

To understand how features evolve across different SAE capacities and to identify meaningful groupings of related features, we studied two distinct phenomena: *feature splitting* and *feature families*. *Feature splitting* – the tendency of features appearing in larger SAEs to "split" the direction spanned by a feature from a smaller SAE, and activate on granular sub-topics of the smaller SAE's feature – has been observed in previous work on sparse autoencoders (e.g. Bricken et al., 2023). Examples of feature splitting, as well as features recurring across SAEs, can be found in Figures 16 and 17a/17b.

In contrast, *feature families* exist within a single SAE, and exhibit a clear hierarchical structure with a dense "parent" feature and several sparser "child" features; we suggest that the "parent" feature encompasses a broader, more abstract concept that is shared among the "child" features. An example feature family from `cs.LG` can be seen in Figure 4.

4.1 Feature splitting

We investigated how features in smaller SAEs relate to features in larger SAEs through a nearest neighbour approach. For each pair of SAEs (i.e. SAE16 and SAE32) with n_1 and n_2 features respectively, we calculated an $n_1 \times n_2$ similarity matrix S where $S_{ij} = \mathbf{w}_i^T \mathbf{w}_j / \|\mathbf{w}_i\| \|\mathbf{w}_j\|$. Here, $\mathbf{w}_i$ and $\mathbf{w}_j$ are decoder weight vectors for features in the smaller and larger SAE, respectively. For each feature in the larger SAE, we identified the most similar feature in the smaller SAE, allowing us to trace how features potentially "split" or become more refined as model capacity increases.

Our results are shown in Figure 9. We find that increasing both number of active latents k and the latent dimension n reduces the similarity between nearest neighbours in differently sized SAEs. This agrees with intuition: larger models with more capacity (higher k and n) can learn more fine-grained and specialised features, leading to greater differentiation from features in smaller models.

Qualitatively, matching features from small to large SAEs, we find both *recurrent* features and *novel* features. Recurrent features appear with extremely high S_{ij} and activation similarity across one or more model pairs, and have highly similar auto-generated interpretations, suggesting semantic closeness; these are much more common for lower k (see Figure 16). In contrast, novel features have distinct semantic meaning from their nearest-neighbour match, and activate similarly on some but not all documents; novel features thus *split* the semantic space covered by their nearest-neighbour match from a smaller SAE. However, some novel features share little semantic or activation overlap with their nearest-neighbour feature, as in Fig. 17b, indicating smaller SAEs may not sufficiently cover the feature space; see E.1 in the Appendix for more details.

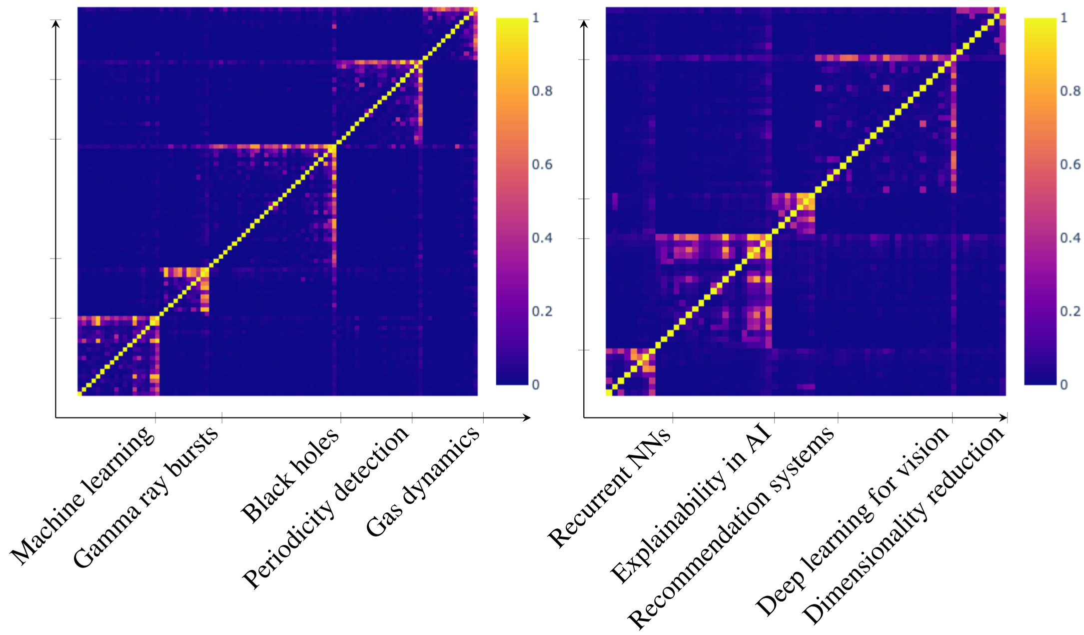
4.2 Feature families

Feature family identification To identify feature families, we developed a graph-based approach using co-activation patterns across the dataset. We consider only highly interpretable features (F1 ≥ 0.8, Pearson ≥ 0.8).

We first compute co-occurrence matrix C and activation similarity matrix D. For all data points k, $C_{ij} = \sum_k A_{ik} A_{jk}$, $D_{ij} = \sum_k B_{ik} B_{jk}$ where $A_{ik} = 1$ if feature i is active on example k (0 otherwise), and $B_{ik} = \mathbf{h_{k,i}}$ if feature i is active on example k with hidden vector $\mathbf{h_k}$ (0 otherwise). We normalise the co-occurrence matrix by feature activation frequencies and apply a threshold to focus on significant relationships: $C_{ij}^{norm} = \frac{C_{ij}}{f_i + \epsilon}$ where $f_i = \sum_k A_{ik}$ is the activation frequency of feature i and ϵ is a small constant for numerical stability. We then apply a threshold τ to obtain C_{ij}^{thresh} (hereafter just C). We construct a maximum spanning tree (MST) from C, capturing the strongest relationships between features while avoiding cycles. We convert the MST to a directed graph, with edges pointing from higher-density to lower-density features, representing a hierarchy from more general to more specific concepts. We identify feature families via depth-first-search in this directed graph, starting from root nodes (i.e., no incoming edges) and recursively exploring hierarchical sub-families.

We iterate this process, removing parent features after each iteration to re-form the MST and reveal overlapping, finer-grained feature families. We de-duplicate families with high set overlap ($\frac{|F_1 \cap F_2|}{|F_1 \cup F_2|} > 0.6$). In practice, we choose $\tau = 0.1$ and use $n = 3$ iterations.

Feature family interpretability To evaluate the interpretability of feature families and their relevance to scientific concept understanding, we analysed their collective properties and the effectiveness of high-level descriptions in capturing their behaviour in scientific contexts. For each family, we generated a "superfeature" description using GPT-4o, based on the individual feature descriptions within that family. We then uniformly sampled high-activating examples across all activations of child features, and assessed the interpretability of the superfeature using a prediction task, where GPT-4o predicted whether test abstracts would activate the superfeature. We compared these predictions to ground truth activations to compute Pearson correlation and F1 scores. Additionally, we calculated



(a) (b)

Figure 5: Co-occurrence matrix C organised by a subset of 5 feature families each. Features in families are ordered by firing density, and the right-most feature is the parent. The un-filled block structure reflects the hierarchical nature of the feature family: all children co-occur with the parent, but few children fire with each other. Visually, this supports our clustering approach.

several metrics to characterise the structure and coherence of the feature families. Table 1 presents the mean values of these metrics across all families for both the `astro-ph` and `cs.LG` datasets.

Matrix structure We conjecture that feature families are equivalent to diagonal blocks in some permutation of the co-occurrence matrix C and activation similarity matrix D. If feature families are indeed meaningful clusters in the graph, then in C and D in-block elements should co-activate much more strongly than off-diagonal elements. We also argue that due to the hierarchical nature of feature families, matrix "blocks" are highly sparse, since child features all co-occur with the parent feature but rarely co-occur with one another. Subsets of the co-occurrence matrix, permuted by feature family, are shown in 5.

Motivated by these structures, we compute the parent-child co-occurrence ratio $R(p, \mathcal{C})$ for every family with parent feature p and children $\mathcal{C}$, $\frac{\text{avg}(\sum_{i \in \mathcal{C}} A_{ip})}{\text{avg}(\sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{C}, j \neq i} A_{ij})}$. We also permute the co-occurrence and activation similarity matrices by greedily selecting feature families, and compute the in-block to off-diagonal ratios $C_{\text{diag}}/C_{\text{off}}$ and $D_{\text{diag}}/D_{\text{off}}$ (excluding the $i = j$ diagonal), capturing the clustering strength of the block diagonal. Statistics are listed in Table 1.

Dataset	(k, n)	Size	F1	Pearson	$\overline{R(p, \mathcal{C})}$	$C_{\text{diag}}/C_{\text{off}}$	$D_{\text{diag}}/D_{\text{off}}$	f_{inc}
`astro-ph`	(16, 3072)	6	0.86	0.76	10.99	5.13	5.47	0.36
	(32, 6144)	6	0.86	0.73	11.75	4.72	5.87	0.31
	(64, 9216)	7	0.80	0.7	6.87	2.0	3.05	0.24
`cs.LG`	(16, 3072)	5	0.73	0.6	2.44	8.35	0.89	0.23
	(32, 6144)	5	0.73	0.59	3.5	7.33	1.07	0.30
	(64, 9216)	7	0.80	0.71	1.22	1.78	2.57	0.41

Table 1: Interpretability and structure metrics for feature families from `astro-ph` and `cs.LG`; we report medians unless otherwise noted. f_{inc} refers to the fraction of features belonging to a clean (F1 ≥ 0.8, Pearson ≥ 0.8) feature family.

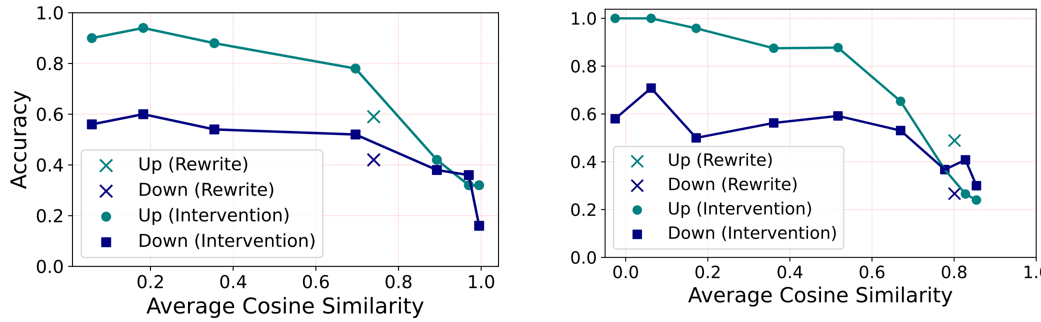


Figure 6: Relationship between intervention accuracy and query fidelity for SAE-based embedding interventions versus traditional query rewriting in scientific literature retrieval for computer science (`cs.LG`) and astronomy (`astro-ph`) domains. Intervention accuracy measures the precision of causal query modifications, while query fidelity is quantified by cosine similarity between original and modified query embeddings.

5 Evaluating effectiveness of search interventions in scientific literature

5.1 Intervening on scientific embeddings with SAE features

As an implementation detail, we note that intervening on a feature by up- or down-weighting its hidden representation and then decoding is equivalent to directly adding the scaled feature vector to the final embedding. This approach allows for precise manipulation of scientific concepts within the embedding space. We explore an alternative process in Appendix G where we iteratively optimise the encoded decoded latents to minimise the difference between the desired feature activations and the actual activations, potentially offering even finer control over scientific concept representation.

5.2 Experiments in scientific literature retrieval

We incorporate SAE-based embedding interventions into a scientific literature retrieval system for computer science (`cs.LG`) and astronomy (`astro-ph`), demonstrating cross-domain applicability in scientific AI. To assess the effectiveness of SAE feature intervention on semantic search of scientific literature, we evaluate the *specificity* and *interpretability* of feature-centric query modifications. We select random samples ($N = 50$ each) of real literature retrieval queries relevant to machine learning and astronomy, which are answerable with information in papers from `cs.LG` and `astro-ph`.

For each scientific query, we return the top $k = 10$ most relevant papers using embedding cosine similarity, forming the original retrieval results $\mathcal{R}$. We then select a random feature i in the top-k from the query's hidden representation $\mathbf{h_q}$, and another orthogonal feature j that has no overlap with the top-k; we limit our selection only to features that are highly interpretable (F1 > 0.9, Pearson > 0.9). Given these features, we create a modified query embedding with $\mathbf{h'q}, \mathbf{i} = \lambda-$ and $\mathbf{h'q}, \mathbf{j} = \lambda+$, letting $\lambda_- = 0$ and sampling $\lambda_+ \in [0, 5]$. This effectively down-weights" and up-weights" the importance of specific scientific concepts i and j, respectively, in the modified query, which is used to generate new retrieval results $\mathcal{R}'$.

To evaluate the effect of up-weighting and down-weighting query modifications on the final retrieval results, we provide both $\mathcal{R}$ and $\mathcal{R}'$ to an external LLM instance. The external LLM then compares $\mathcal{R}$ and $\mathcal{R}'$ and determines which scientific concepts, out of a multiple-choice subset of 5 options, have been up-weighted or down-weighted; we use this to compute the intervention accuracy, which measures the precision and efficacy of causal query interventions in scientific literature search. As a baseline, we compare our SAE-based method against traditional query rewriting, by using another LLM instance to re-write the original query such that it up-weights j and down-weights i entirely using natural language. Our results, shown in Figure 6, demonstrate that SAE feature interventions consistently outperform traditional query rewriting across various levels of query fidelity in scientific literature search.

We also experiment with intervening on feature families, sampling highly interpretable families containing features in the query top-k. This allows us to manipulate scientific concepts at different levels of abstraction. We uniformly adjust weights for all features in the family, including the parent, using the auto-generated family interpretation as the multiple-choice option. Results show that feature family interventions achieve accuracy comparable to individual features, but only down-weighting interventions outperform query re-writing. This may be because feature families can comprehensively down-weight related scientific concepts, while up-weighting a general concept doesn't necessarily require activating all granular child features. Notably, lower cosine similarity isn't inherently undesirable, as changing the query will naturally reduce similarity.

6 Discussion

In this work, we have presented a novel approach towards more interpretable scientific foundation models and literature search, by applying sparse autoencoders (SAEs) to dense text embeddings to derived from large language models. We have demonstrated the usefulness of SAEs in disentangling embeddings of scientific paper abstracts into semantically relevant document-level concepts, an important step towards more transparent and controllable AI systems for scientific applications. We introduced the concept of "feature families" in SAEs, which allow for multi-scale semantic analysis and manipulation of scientific concepts. Furthermore, we showcased the practical utility of our approach by applying these interpretable features to enable fine-grained control over query semantics in scientific literature search, aligning with recent work on controlled text generation (Lee, 2024).

Our approach offers a novel solution to the growing challenge of scientific literature exploration. With the exponential growth in papers, traditional search methods are becoming increasingly ineffective (Tsang et al., 2016). Our SAE-based approach, for which we provide an open-source interface, provides a new way to navigate and find pertinent scientific papers, especially in interdisciplinary fields where relevant work may not be easily discoverable through conventional keyword searches, citation networks, or vector search (Sharma et al., 2022; Thomsett-Scott et al., 2016).

Foundation models, including large language models, are increasingly useful in scientific discovery (Si et al., 2024; Tshitoyan et al., 2019; AI4Science et al., 2023). By providing concept-level interpretability, our work also allows for probing the evolution of scientific fields over time, as captured through state-of-the-art language models and scientific literature corpora. Existing efforts to map the landscape of scientific research and understand domain and conceptual shifts have relied primarily on citation networks and keyword analysis (Boyack et al., 2005; Uzzi et al., 2013). However, SAE features more directly probe semantic meaning and are less sensitive to paper-level or keyword-level variations, potentially enabling more robust literature searches and meta-analyses. Statistics of SAE features representing scientific concepts—such as clustering patterns, co-occurrences, and temporal trends— could gain novel insights into how scientific domains have changed and interacted.

To more thoroughly evaluate our approach, we would like to collect human user evaluations of our SAE-based literature search and compare SAE interventions to other user-facing techniques, e.g. prompt rewriting. We'd like to evaluate our reconstructed embeddings against the original embeddings using a standard semantic embedding benchmark such as MTEB (Muennighoff et al., 2022). We'd also like to be able to conduct an evaluation of SAE features against some proxy of ground-truth features, much like Makelov et al. (2024) propose. For instance, the Unified Astronomy Thesaurus (Frey et al., 2018) could provide a basis for evaluating individual feature overlap with astronomy concepts, and even family features as groupings of these individual concepts.

Limitations: Our work focused on relatively small datasets from specific scientific domains. Although this specificity allowed us to demonstrate the effectiveness of our approach in targeted areas, future work should investigate how well these methods generalise to larger, more diverse datasets. Additionally, our automated interpretability process, while effective, does not utilise the full spectrum of activations, potentially missing nuanced patterns in feature behaviour.

The computational requirements for training SAEs on large embedding datasets also present scalability challenges that need to be addressed for wider adoption of this approach. Our SAEs are quite small in comparison to more general language model SAEs. This proved adequate given that we only require a single embedding vector per example (rather than one per token posiiton) and the narrow domains we trained on, but SAEs for general text embeddings would need to be scaled up by at least 2-3 the total number of latents. Further, while we've demonstrated the utility of our approach for literature search, further work is needed to integrate these interpretable representations into the real-world workflows of human scientists, from hypothesis generation to experimental design and analysis.

References

AI4Science, Microsoft Research and Microsoft Quantum (2023). "The Impact of Large Language Models on Scientific Discovery: a Preliminary Study using GPT-4". In: *ArXiv* abs/2311.07361. URL: https://api.semanticscholar.org/CorpusID:265150648.

Birhane, Abeba, Atoosa Kasirzadeh, David Leslie, and Sandra Wachter (2023). "Science in the age of large language models". In: *Nature Reviews Physics* 5, pp. 277–280. DOI: 10.1038/s42254-023-00581-4.

Boyack, Kevin W, Richard Klavans, and Katy B"orner (2005). "Mapping the backbone of science". In: *Scientometrics* 64, pp. 351–374.

Bricken, Trenton, Catherine Olsson, and Neel Nanda (2023). "Towards Monosemanticity: Decomposing Language Models With Dictionary Learning". In: *arXiv preprint arXiv:2301.05498*.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). "Language Models are Few-Shot Learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Cao, Wenqiang, Qing Li, Siying Zhang, Rixin Xu, and Youqi Li (2023a). "STEP: Generating Semantic Text Embeddings with Prompt". In: *2023 Eleventh International Conference on Advanced Cloud and Big Data (CBD)*, pp. 180–185. URL: https://api.semanticscholar.org/CorpusID:269628678.

Cao, Yichen, Xinyi Wang, Yiran Cao, Renfeng Xu, Zhihan Dong, Qi Fang, Yeyun Gong, Lei Li, Shuming Shi, Jiafeng Yan, et al. (2023b). "Step-gs: Guiding large language models via step-by-step prompting". In: *arXiv preprint arXiv:2305.11725*.

Conmy, Arthur and Neel Nanda (2024). *Activation Steering with SAEs*. Accessed 16-07-2024. URL: https://www.lesswrong.com/posts/C5KAZQib3bzzpeyrg/full-post-progress-update-1-from-the-gdm-mech-interp-team#Activation_Steering_with_SAEs.

Cunningham, Hoagy, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey (2023a). *Sparse Autoencoders Find Highly Interpretable Features in Language Models*. arXiv: 2309.08600 [cs.LG]. URL: https://arxiv.org/abs/2309.08600.

– (2023b). "Sparse autoencoders find highly interpretable features in language models". In: *arXiv preprint arXiv:2309.08600*.

Daujotas, Gytis (2024). *Interpreting and Steering Features in Images*. https://www.lesswrong.com/posts/Quqekpvx8BGMMcaem/interpreting-and-steering-features-in-images. [Accessed 16-07-2024].

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805*.

Donoho, David L (2006). "Compressed sensing". In: *IEEE Transactions on Information Theory* 52.4, pp. 1289–1306.

Elhage, Nelson, 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 (2022a). *Toy Models of Superposition*. arXiv: 2209.10652 [cs.LG]. URL: https://arxiv.org/abs/2209.10652.

Elhage, Nelson, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Johnston, Ben Mann, Amanda Askell, Danny Hernandez, Dawn Drain, Zac Hatfield-Dodds, et al. (2022b). "Softmax Linear Units". In.

Engels, Joshua, Isaac Liao, Eric J. Michaud, Wes Gurnee, and Max Tegmark (2024). *Not All Language Model Features Are Linear*. arXiv: 2405.14860 [cs.LG]. URL: https://arxiv.org/abs/2405.14860.

Frey, Katie and Alberto Accomazzi (2018). "The Unified Astronomy Thesaurus: Semantic metadata for astronomy and astrophysics". In: *The Astrophysical Journal Supplement Series* 236.1, p. 24.

Gao, Leo, John Thickstun, Anirudh Madaan, Zach Scherlis, Arush Guha, Sumanth Dathathri, Jared Kaplan, Azalia Mirhoseini, and Ilya Sutskever (2024). "Scaling Laws for Neurons in GPT Models". In: *arXiv preprint arXiv:2401.02325*.

Gao, Luyu, Xueguang Ma, Jimmy Lin, and Jamie Callan (2022). *Precise Zero-Shot Dense Retrieval without Relevance Labels*. arXiv: 2212.10496 [cs.IR]. URL: https://arxiv.org/abs/2212.10496.

Gao, Tianyu, Xingcheng Yao, and Danqi Chen (2021). "SimCSE: Simple contrastive learning of sentence embeddings". In: *arXiv preprint arXiv:2104.08821*.

Iyer, Kartheik G., Mikaeel Yunus, Charles O'Neill, Christine Ye, Alina Hyk, Kiera McCormick, Ioana Ciuca, John F. Wu, Alberto Accomazzi, Simone Astarita, Rishabh Chakrabarty, Jesse Cranney, Anjalie Field, Tirthankar Ghosal, Michele Ginolfi, Marc Huertas-Company, Maja Jablonska, Sandor Kruk, Huiling Liu, Gabriel Marchidan, Rohit Mistry, J. P. Naiman, J. E. G. Peek, Mugdha Polimera, Sergio J. Rodriguez, Kevin Schawinski, Sanjib Sharma, Michael J. Smith, Yuan-Sen Ting, and Mike Walmsley (2024). *pathfinder: A Semantic Framework for Literature Review and Knowledge Discovery in Astronomy*. arXiv: 2408.01556 [astro-ph.IM]. URL: https://arxiv.org/abs/2408.01556.

Jermyn, Adam and Adly Templeton (2023). *Ghost Grads: An improvement on resampling*. [Accessed 19-07-2024]. URL: https://transformer-circuits.pub/2024/jan-update/index.html#dict-learning-resampling.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kinney, Rodney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, Miles Crawford, Doug Downey, Jason Dunkelberger, Oren Etzioni, Rob Evans, Sergey Feldman, Joseph Gorney, David Graham, Fangzhou Hu, Regan Huff, Daniel King, Sebastian Kohlmeier, Bailey Kuehl, Michael Langan, Daniel Lin, Haokun Liu, Kyle Lo, Jaron Lochner, Kelsey MacMillan, Tyler Murray, Chris Newell, Smita Rao, Shaurya Rohatgi, Paul Sayre, Zejiang Shen, Amanpreet Singh, Luca Soldaini, Shivashankar Subramanian, Amber Tanaka, Alex D. Wade, Linda Wagner, Lucy Lu Wang, Chris Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Madeleine Van Zuylen, and Daniel S. Weld (2023). *The Semantic Scholar Open Data Platform*. arXiv: 2301.10140 [cs.DL]. URL: https://arxiv.org/abs/2301.10140.

Lála, Jakub, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G. Rodriques, and Andrew D. White (2023). *PaperQA: Retrieval-Augmented Generative Agent for Scientific Research*. arXiv: 2312.07559 [cs.CL]. URL: https://arxiv.org/abs/2312.07559.

Lee, Linus (2024). *Prism: mapping interpretable concepts and features in a latent space of language*. Accessed 16-07-2024. URL: https://thesephist.com/posts/prism.

Liu, Nelson F, Matt Gardner, Yonatan Belinkov, Matthew E Peters, and Noah A Smith (2019). "Linguistic knowledge and transferability of contextual representations". In: *arXiv preprint arXiv:1903.08855*.

Makelov, Aleksandar, George Lange, and Neel Nanda (2024). "Towards principled evaluations of sparse autoencoders for interpretability and control". In: *arXiv preprint arXiv:2405.08366*.

Makhzani, Alireza and Brendan Frey (2013). "K-sparse autoencoders". In: *arXiv preprint arXiv:1312.5663*.

Muennighoff, Niklas, Nouamane Tazi, Loic Magne, and Nils Reimers (2022). "MTEB: Massive text embedding benchmark". In: *arXiv preprint arXiv:2210.07316*.

Nanda, Neel (2023). *Open Source Replication & Commentary on Anthropic's Dictionary Learning Paper*. [Accessed 22-07-2024]. URL: https://www.alignmentforum.org/posts/fKuugaxt2XLTkASkk/open-source-replication-and-commentary-on-anthropic-s.

Ng, Andrew et al. (2011). "Sparse autoencoder". In: *CS294A Lecture notes*. Vol. 72. 2011, pp. 1–19.

Nguyen, Tuan Dung, Yuan-Sen Ting, Ioana Ciucă, Charlie O'Neill, Ze-Chang Sun, Maja Jabłońska, Sandor Kruk, Ernest Perkowski, Jack Miller, Jason Li, Josh Peek, Kartheik Iyer, Tomasz Różański, Pranav Khetarpal, Sharaf Zaman, David Brodrick, Sergio J. Rodríguez Méndez, Thang Bui, Alyssa Goodman, Alberto Accomazzi, Jill Naiman, Jesse Cranney, Kevin Schawinski, and UniverseTBD (2023). *AstroLLaMA: Towards Specialized Foundation Models in Astronomy*. arXiv: 2309.06126 [astro-ph.IM]. URL: https://arxiv.org/abs/2309.06126.

Olshausen, Bruno A and David J Field (1997). "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision Research* 37.23, pp. 3311–3325.

Qu, Jiaxing, Yuxuan Richard Xie, Kamil M. Ciesielski, Claire E. Porter, Eric S. Toberer, and Elif Ertekin (2024). "Leveraging language representation for materials exploration and discovery". In: *npj Computational Materials* 10, pp. 1–14. DOI: 10.1038/s41524-024-01231-8.

Rajamanoharan, Senthooran, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda (2024). "Improving dictionary learning with gated sparse autoencoders". In: *arXiv preprint arXiv:2404.16014*.

Rasmy, Laila, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi (2021). "Med-BERT: pretrained contextualized embeddings on large-scale structured electronic health records for disease prediction". In: *npj Digital Medicine* 4.1, pp. 1–13.

Reimers, Nils, Lucas Beyer, and Iryna Wang (2022). "The curse of dense low-dimensional information retrieval for large index sizes". In: *arXiv preprint arXiv:2112.07899*.

Reimers, Nils and Iryna Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 3982–3992.

Romera-Paredes, Bernardino, Mohammadamin Barekatain, Alexander Novikov, et al. (2024). "Mathematical discoveries from program search with large language models". In: *Nature* 625, pp. 468–475. DOI: 10.1038/s41586-023-06924-6.

Sharma, Ritu, Sarita Gulati, Amanpreet Kaur, Atasi Sinhababu, and Rupak Chakravarty (2022). "Research discovery and visualization using ResearchRabbit: A use case of AI in libraries". In: *COLLNET Journal of Scientometrics and Information Management* 16.2, pp. 215–237.

Si, Chenglei, Diyi Yang, and Tatsunori Hashimoto (2024). *Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers*. arXiv: 2409.04109 [cs.CL]. URL: https://arxiv.org/abs/2409.04109.

Taylor, Ross, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic (2022). *Galactica: A Large Language Model for Science*. arXiv: 2211.09085 [cs.CL]. URL: https://arxiv.org/abs/2211.09085.

Templeton, Adly (2024). *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic.

Thomsett-Scott, Beth and Patricia E Reese (2016). "Academic libraries and discovery tools: A survey of the literature". In: *Discovery Tools: The Next Generation of Library Research*, pp. 3–23.

Trifonov, Valentin, Octavian-Eugen Ganea, Anna Potapenko, and Thomas Hofmann (2018). *Learning and Evaluating Sparse Interpretable Sentence Embeddings*. arXiv: 1809.08621 [cs.CL]. URL: https://arxiv.org/abs/1809.08621.

Tsang, Daniel C and Julia M Gelfand (2016). "The Changing Landscape of Research Library Collections: Ensuring Realistic Sustainability." In.

Tshitoyan, Vahe, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A. Persson, Gerbrand Ceder, and Anubhav Jain (2019). "Unsupervised word embeddings capture latent knowledge from materials science literature". In: *Nature* 571, pp. 95–98. DOI: 10.1038/s41586-019-1335-8.

Turian, Joseph, Lev Ratinov, and Yoshua Bengio (2010). "Word representations: a simple and general method for semi-supervised learning". In: *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394.

Uzzi, Brian, Satyam Mukherjee, Michael Stringer, and Ben Jones (2013). "Atypical combinations and scientific impact". In: *Science* 342.6157, pp. 468–472.

Wang, Liang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei (2024). *Text Embeddings by Weakly-Supervised Contrastive Pre-training*. arXiv: 2212.03533 [cs.CL]. URL: https://arxiv.org/abs/2212.03533.

Wright, Benjamin and Lee Sharkey (2024). *Addressing Feature Suppression in SAEs*. https://www.alignmentforum.org/posts/3JuSjTZyMzaSeTxKk/addressing-feature-suppression-in-saes. [Accessed 16-07-2024].

Contents

A Training details

A.1 Training setup

Our sparse autoencoder (SAE) implementation incorporates several recent advancements in the field. Following Bricken et al. (2023), we initialise the bias b_{pre} using the geometric median of a data point sample and set encoder directions parallel to decoder directions. Decoder latent directions are normalised to unit length at initialisation and after each training step. For our top-k models, based on Gao et al. (2024), we set initial encoder magnitudes to match input vector magnitudes, though our analyses indicate minimal impact from this choice.

We also use an auxiliary loss, similar to the "ghost grads" technique (Jermyn et al., 2023), to model the reconstruction error using the top k_{aux} dead latents, where we typically set $k_{\text{aux}} = 2k$ (Gao et al., 2024). Latents are flagged as dead during training if they have not activated for a predetermined number of tokens (in our case, one full epoch through the training data). Given the reconstruction error of the main model $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$, we define the auxiliary loss as $\mathcal{L}_{\text{aux}}(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{e} - \hat{\mathbf{e}}\|_2^2$ where $\hat{\mathbf{e}} = W_d\mathbf{z}$ is the reconstruction using the top k_{aux} dead latents, and $\mathbf{z}$ is the sparse representation using only these dead latents. This additional loss term helps to revive dead features and improve the overall representational capacity of the model (Gao et al., 2024). We found that dead latents only occurred during training the $k = 16$ models, and all dead latents had disappeared by the end of training. We show how dead latents evolved over training the $k = 16$ SAEs for the `astro-ph` abstracts in Figure 7.

For optimisation, we employ Adam (Kingma et al., 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, maintaining a constant learning rate. We use gradient clipping. Our training uses batches of 1024 abstracts, with performance metrics showing robustness to batch size variations under appropriate hyperparameter settings.

The primary MSE loss uses a global normalisation factor computed at training initiation, while the AuxK loss employs per-batch normalisation to adapt to evolving error distributions. Following Bricken et al. (2023), we apply a gradient projection technique to mitigate interactions between the Adam optimiser and decoder normalisation.

A.2 Training and automated interpretability methods

Training: We train our top-k SAEs on the embeddings of abstracts from papers on arXiv with the `astro-ph` tag (astrophysics, 272,000 papers) and the `cs.LG` tag (computer science, 153,000 papers). The embeddings were generated with OpenAI's `text-embedding-3-small` model.[1] We train our SAEs on these collections of embeddings separately. We normalised the embeddings to zero mean and unit variance before passing them to the SAE as inputs. Our trained SAEs will be made available for download.

Hyperparameters: Notable hyperparameters include the number of active latents k, the total number of latents n, the number of auxiliary latents k_{aux}, the learning rate, and the auxiliary loss coefficient α. We found learning rate and auxiliary loss coefficient to not have a significant effect on final reconstruction loss; we set the former to 1e-4 and the latter to 1/32. We vary k between 16 and 128, and n between two to nine times the embedding dimension d_{input}. Whilst we train SAEs with many

[1] https://openai.com/index/new-embedding-models-and-api-updates/

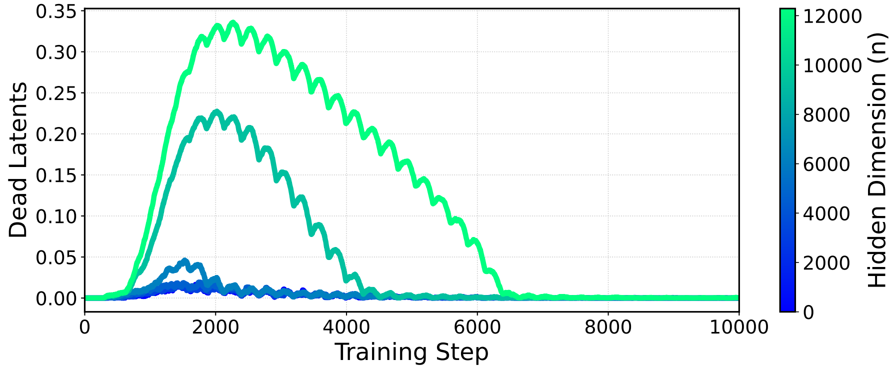


Figure 7: The proportion of dead latents, defined as features that haven't fired in the last epoch of training, for our $k = 16$ SAEs on the `astro-ph` abstract embeddings. All dead latents were gone by the end of training. We found that dead latents only occurred in $k = 16$ autoencoders.

different combinations of these hyperparameters, we largely focus on what we hereon refer to as SAE16 ($k = 16$, $n = 2d_{\text{input}} = 3072$), SAE32 ($k = 32$, $n = 4d_{\text{input}} = 6144$) and SAE64 ($k = 64$, $n = 6d_{\text{input}} = 9216$). We train each model for approximately 13.2 thousand steps.

Automated interpretability: Following the training of a Sparse Autoencoder (SAE), it becomes necessary to interpret its features, each corresponding to a column in the learned decoder weight matrix. To facilitate feature interpretation and quantify interpretation confidence, we employ two Large Language Model (LLM) instances: the *Interpreter* and the *Predictor*. The Interpreter is tasked with generating labels for each feature. It is provided with the abstracts that produce the top 5 activations of the feature across the dataset, along with randomly selected abstracts that do not activate the feature. The Interpreter then generates a label for the feature based on this input (for the complete prompt, refer to Appendix C). Subsequently, the generated label is passed to the Predictor. The Predictor is presented with three randomly sampled abstracts where the feature was activated and three where it was not. It is then instructed to predict whether a given abstract should activate the feature, expressing its confidence as a score ranging from -1 (absolute certainty of non-activation) to $+1$ (absolute certainty of activation).[2] We measure the Pearson correlation between this confidence and the true activation (binary; +1 or -1). We also measure the F1 score, when framing the confidence as a binary classification (active if confidence is above 0, inactive otherwise).

Evaluation metrics: In order to compare SAEs, we evaluate both their ability to reconstruct the embeddings, as well as the interpretability of the learned features. For the former, we examine the normalised mean squared error (MSE), where we divide MSE by the error when predicting the mean activations. We also report the log density of the activation of features across all papers. We do not report dead latents (those not firing on any abstract) as all models contained zero dead latents at the end of training. We also report the mean activation of features, when their activation is non-zero. To measure interpretability, we use Pearson correlation, as outlined above.

A.3 SAE training metrics

Table 2 shows the final training metrics for all combinations of SAEs trained. We note clear trends in normalised MSE, log feature density and activation mean as we vary the number of active latents k and the overall number of latents n.

[2] We use 3 activating and 3 non-activating abstracts for the Predictor, rather than 5, due to LLM costs. We used `gpt-4o` as the Interpreter and `gpt-4o-mini` as the Predictor. Notably, we predict each abstract separately, rather than batching abstracts like Bricken et al. (2023).

Table 2: Metrics for our top-k sparse autoencoders with varying k and hidden dimensions, across both astronomy and computer science papers. MSE is normalised mean squared error, Log FD is the mean log density of feature activations, and activation mean is the mean activation value across non-zero features. Note that MSE is normalised.

		`astro.ph`			`cs.LG`		
k	n	MSE	Log FD	Act Mean	MSE	Log FD	Act Mean
16	3072	0.2264	-2.7204	0.1264	0.2284	-2.7314	0.1332
	4608	0.2246	-4.7994	0.1350	0.2197	-3.0221	0.1338
	6144	0.2128	-3.1962	0.1266	0.2089	-3.2299	0.1342
	9216	0.1984	-3.4206	0.1264	0.1962	-3.4833	0.1343
	12288	0.1957	-6.2719	0.1274	0.1897	-3.6448	0.1347
32	3072	0.1816	-2.3389	0.0847	0.1831	-2.3008	0.0885
	4608	0.1691	-3.6091	0.0882	0.1697	-2.5152	0.0876
	6144	0.1604	-2.7761	0.0841	0.1641	-2.6687	0.0873
	9216	0.1554	-3.0227	0.0842	0.1540	-2.9031	0.0875
	12288	0.1520	-4.9505	0.0843	0.1457	-3.0577	0.0877
64	3072	0.1420	-1.9538	0.0566	0.1485	-1.8875	0.0584
	4608	0.1331	-2.7782	0.0622	0.1370	-2.0637	0.0570
	6144	0.1262	-2.2828	0.0545	0.1310	-2.1852	0.0558
	9216	0.1182	-2.4682	0.0539	0.1240	-2.3536	0.0545
	12288	0.1152	-3.4787	0.0583	0.1162	-2.4847	0.0548
128	3072	0.1111	-1.8876	0.0483	0.1206	-1.5311	0.0399
	4608	0.1033	-2.1392	0.0457	0.1137	-1.6948	0.0376
	6144	0.1048	-2.2501	0.0438	0.1076	-1.8079	0.0366
	9216	0.0975	-2.5352	0.0409	0.0999	-1.9701	0.0348
	12288	0.0936	-2.7025	0.0399	0.0942	-2.0858	0.0342

A.4 Scaling laws

For the left panel of Figure 2, which shows the scaling of normalised MSE with the number of total latents n, we observe the following power-law relationships:

$$k = 16 : L(n) = 0.61n^{-0.12} \text{ (astro.ph)};\ L(n) = 0.67n^{-0.13} \text{ (cs.LG)}$$
$$k = 32 : L(n) = 0.49n^{-0.13} \text{ (astro.ph)};\ L(n) = 0.56n^{-0.14} \text{ (cs.LG)}$$
$$k = 64 : L(n) = 0.46n^{-0.15} \text{ (astro.ph)};\ L(n) = 0.60n^{-0.17} \text{ (cs.LG)}$$
$$k = 128 : L(n) = 0.31n^{-0.13} \text{ (astro.ph)};\ L(n) = 0.51n^{-0.18} \text{ (cs.LG)}$$

For the right panel of Figure 2, which shows the scaling of normalised MSE with the amount of compute C (in FLOPs), we observe the following power-law relationships:

$$k = 16 : L(C) = 3.84C^{-0.11}$$
$$k = 32 : L(C) = 5.25C^{-0.13}$$
$$k = 64 : L(C) = 8.03C^{-0.16}$$
$$k = 128 : L(C) = 2.80C^{-0.13}$$

These equations demonstrate the consistent power-law scaling behaviour of sparse autoencoders across different values of k, n, and compute C.

A.5 Feature density and similarity

We find an intuitive relationship between k and n and the log feature density (essentially, how often a given feature fires). As k increases, we get a sharper peak of log feature density, shifted to the right, suggesting features fire in a tighter range as we increase the instantaneous L0 of the SAE's encoder (Figure 8).

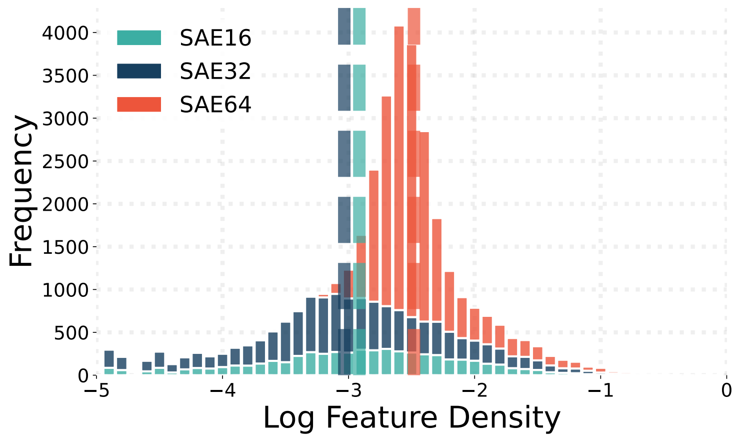
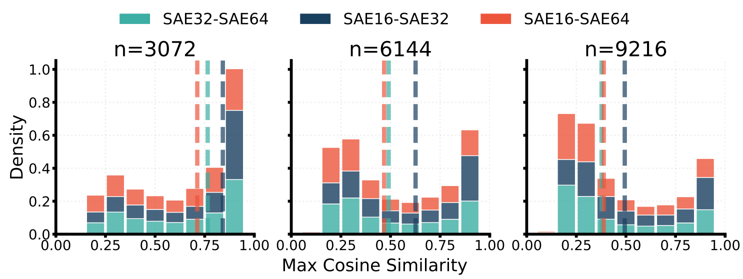
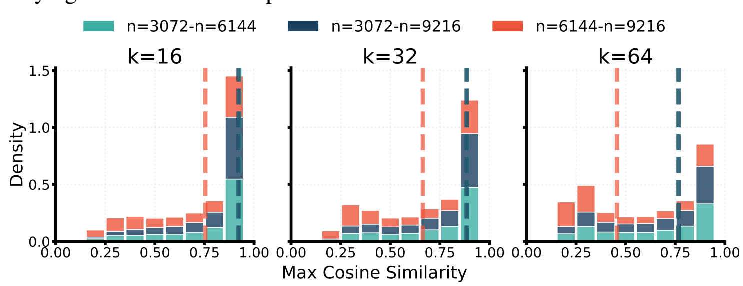


Figure 8: Log feature density for features in our three SAEs as a stacked histogram, showing the distribution of how often features fire across all paper abstacts (`cs.LG` and `astro-ph`). The larger SAE has a higher mean feature density than the smaller SAEs.



(a) k fixed, varying n. As n increases, the features between across SAEs with varying k become more disparate.



(b) n fixed, varying k. Higher values of k lead to less similarity regardless of n.

Figure 9: Nearest-neighbour cosine similarity distributions for SAE features. To find features in an SAE with a lower k that are most similar to those in an SAE with a larger k, we compute the cosine similarity between each feature in the larger model and each feature in the smaller model. We do this for several values of n, and combine the distributions for `astro.ph` and `cs.LG`.

To compare features across different SAEs trained on the same input data, we analyse the cosine similarity between the decoder weight vectors corresponding to each feature. Decoder weights, represented by columns in the decoder matrix, directly encode each feature's contribution to input reconstruction. Encoder weights, on the other hand, are optimised to extract feature coefficients while minimising interference between non-orthogonal features. This separation is important in the context of superposition, where we have more features than input dimensions, precluding perfect orthogonality.

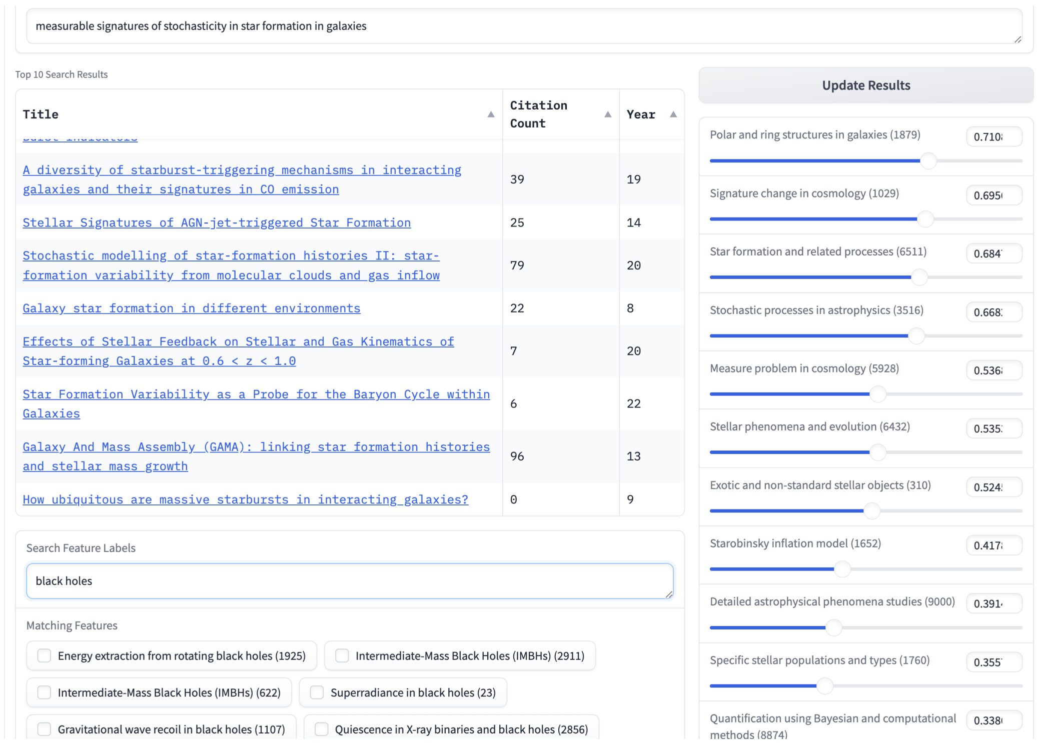


Figure 10: The SAErch tab of our web application, demonstrating a semantic search for "measurable signatures of stochasticity in star formation in galaxies" in the astrophysics domain. The interface displays the top 10 search results ranked by relevance, including title, citation count, and publication year. On the right, sliders represent the top activated SAE features for the query, allowing users to fine-tune the search by adjusting feature weights. On the bottom we have our feature addition interface. Users can search for specific semantic features (e.g., "black holes") and add them to their query. They can then adjust the strength of these features.

B `SAErch.ai`

To demonstrate the practical applications of our sparse autoencoder (SAE) approach to semantic search and feature interpretation, we developed a web application that allows users to interact with the SAE models trained on arXiv paper embeddings. The link will be made public at the end of the anonymity period.

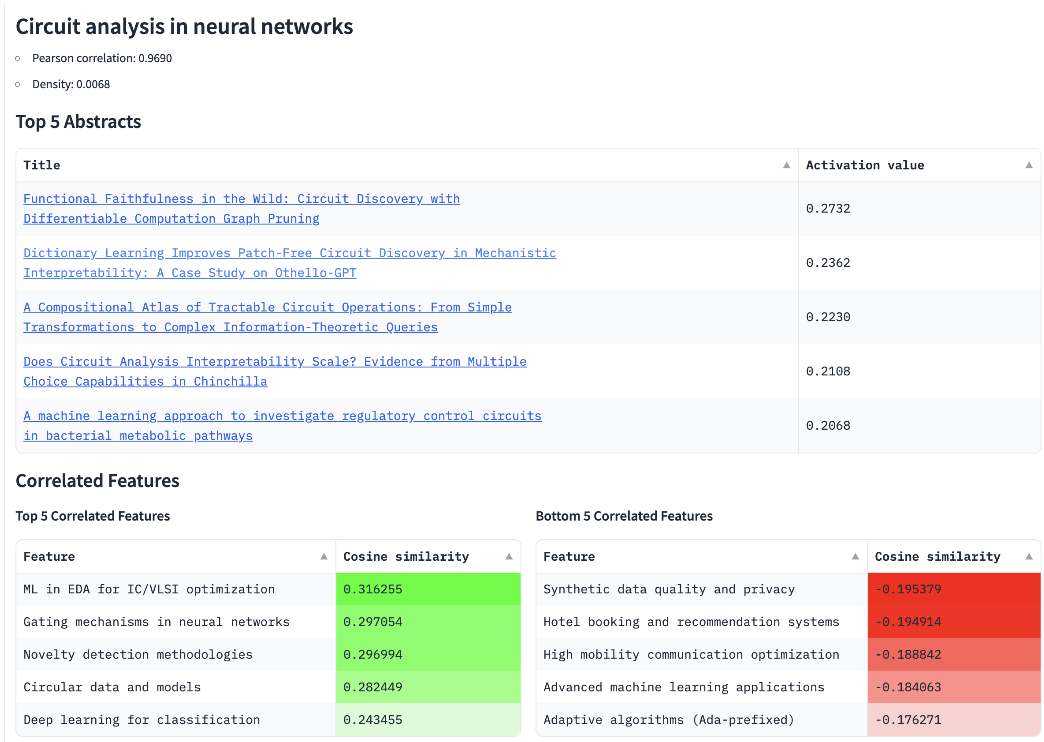
B.1 Overview

SAErch.ai is built using the Gradio framework and consists of three main tabs: Home, SAErch, and Feature Visualisation. The application allows users to switch between the Computer Science (`cs.LG`) and Astrophysics (`astro-ph`) datasets.

The SAErch tab implements the core functionality of our semantic search system, allowing users to:

- Input a search query
- View the top 10 search results based on embedding similarity
- Interact with the SAE features activated by their query

For each query, the system displays sliders corresponding to the top-k SAE features activated by the input. Users can adjust these sliders to modify the query embedding, effectively steering the search results towards or away from specific semantic concepts; see Figure 10. This directly demonstrates the fine-grained control over query semantics discussed in Section 5 of our paper. Users can also search for and add specific features not initially activated by their query (Figure **??**).



Circuit analysis in neural networks

- Pearson correlation: 0.9690
- Density: 0.0068

Top 5 Abstracts

Title	Activation value
Functional Faithfulness in the Wild: Circuit Discovery with Differentiable Computation Graph Pruning	0.2732
Dictionary Learning Improves Patch-Free Circuit Discovery in Mechanistic Interpretability: A Case Study on Othello-GPT	0.2362
A Compositional Atlas of Tractable Circuit Operations: From Simple Transformations to Complex Information-Theoretic Queries	0.2230
Does Circuit Analysis Interpretability Scale? Evidence from Multiple Choice Capabilities in Chinchilla	0.2108
A machine learning approach to investigate regulatory control circuits in bacterial metabolic pathways	0.2068

Correlated Features

Top 5 Correlated Features

Feature	Cosine similarity
ML in EDA for IC/VLSI optimization	0.316255
Gating mechanisms in neural networks	0.297054
Novelty detection methodologies	0.296994
Circular data and models	0.282449
Deep learning for classification	0.243455

Bottom 5 Correlated Features

Feature	Cosine similarity
Synthetic data quality and privacy	-0.195379
Hotel booking and recommendation systems	-0.194914
High mobility communication optimization	-0.188842
Advanced machine learning applications	-0.184063
Adaptive algorithms (Ada-prefixed)	-0.176271

Figure 11: Individual feature visualisation for the "Circuit analysis in neural networks" feature in the computer science domain. The interface displays key interpretability metrics, top activating abstracts, correlated and co-occurring features, and an activation distribution histogram. Further information (not shown in the image) includes co-occurring features and activation distribution.

B.2 Feature Visualisation Tab

The Feature Visualisation tab is divided into two sub-tabs: Individual Features and Feature Families. This section of the application directly relates to our analysis of SAE features and feature families discussed in Sections 3 and 4.

B.2.1 Individual Features

For any selected feature, this tab displays:

- Top 5 activating abstracts, demonstrating the semantic content captured by the feature
- Top and bottom 5 correlated features, illustrating the relationships between different SAE features
- Top 5 co-occurring features, showing which features tend to activate together
- A histogram of activation values, providing insight into the feature's behavior across the corpus
- The most similar features in SAE16 and SAE32

B.2.2 Feature Families

The Feature Families tab in our web application offers an in-depth exploration of related features discovered by our sparse autoencoder. We show an example feature family in Figure 12.

The table displays the parent feature (superfeature) and its child features, along with key metrics, such as the name of the parent and child features, the frequency of co-occurrence between the child feature and the parent feature, ranging from 0 to 1, and the F1 Score and Pearson correlation.

The interactive directed graph provides a visual representation of the feature family structure. Each node represents a feature. The size of the node corresponds to the feature's density (frequency of

Family and Child Features

Feature	Parent Co-Occurrence	F1 Score	Pearson
Transformer models	1	1	0.9994
Positional Encoding in Neural Networks	0.34	1	0.9992
Vision Transformers (ViT) and training techniques	0.34	1	0.9977
Theoretical and Practical Aspects of Transformers	0.24	1	0.9992
Transformer architectures and enhancements	0.72	1	0.9993
Vision Transformers and Efficiency Techniques	0.49	1	0.9923
Swin Transformer in Vision Tasks	0.49	1	1
Mamba architecture and applications	0.95	1	0.9992
Transformer architectures and applications	1	1	0.9638

Family Graph

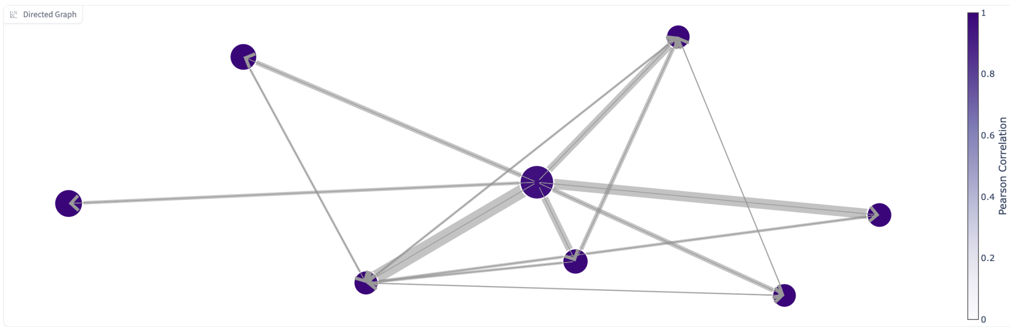


Figure 12: Directed graph visualization of a transformer models feature family. Nodes represent individual features, with size indicating feature density and color intensity showing Pearson correlation. Edges depict relationships between features, with arrow direction pointing from more general to more specific concepts. Users can hover over nodes to view detailed feature information.

activation), while the color intensity indicates the Pearson correlation (interpretability). Arrows between nodes show relationships between features, with the direction typically pointing from more general to more specific concepts. Users can hover over nodes to view detailed information about each feature, including its name and log density.

C Automated interpretability details

C.1 Examples of features

We show some examples of perfectly interpretable features (Pearson correlation > 0.99) in Table 3. The strength of the activation of the feature on its top 3 activating abstracts is shown in parentheses next to the abstract title.

Feature			
Astronomy			
Cosmic Microwave Background	CMB map-making and power spectrum estimation (0.1708)	How to calculate the CMB spectrum (0.1598)	CMB data analysis and sparsity (0.1581)
Periodicity in astronomical data	Generalized Lomb-Scargle analysis of decay rate measurements from the Physikalisch-Technische Bundesanstalt (0.1027)	Multicomponent power-density spectra of Kepler AGNs, an instrumental artefact or a physical origin? (0.0806)	RXTE observation of the X-ray burster 1E 1724-3045. I. Timing study of the persistent X-ray emission with the PCA (0.0758)
X-ray reflection spectra	X-ray reflection spectra from ionized slabs (0.3859)	The role of the reflection fraction in constraining black hole spin (0.3803)	Relativistic reflection: Review and recent developments in modeling (0.3698)
Critique or refutation of theories	What if string theory has no de Sitter vacua? (0.2917)	No evidence of mass segregation in massive young clusters (0.2051)	Ruling Out Initially Clustered Primordial Black Holes as Dark Matter (0.2029)
Computer Science			
Sparsity in Neural Networks	Two Sparsities Are Better Than One: Unlocking the Performance Benefits of Sparse-Sparse Networks (0.3807)	Truly Sparse Neural Networks at Scale (0.3714)	Topological Insights into Sparse Neural Networks (0.3689)
Gibbs Sampling and Variants	Herded Gibbs Sampling (0.2990)	Characterizing the Generalization Error of Gibbs Algorithm with Symmetrized KL information (0.2858)	A Framework for Neural Network Pruning Using Gibbs Distributions (0.2843)
Arithmetic operations in transformers	Arbitrary-Length Generalization for Addition in a Tiny Transformer (0.1828)	Carrying over algorithm in transformers (0.1803)	Understanding Addition in Transformers (0.1792)

Table 3: Activation strengths and titles for abstracts related to Astronomy and Computer Science features.

C.2 Automated interpretability prompts

We provide the prompts used for the Interpreter model and the Predictor model in the boxes below. Where `this` text is used, it represents an input to the model. We found that performance significantly increased when including the instruction to use "Occam's razor", whereby the simplest feature at the appropriate level of granularity was selected.

Interpreter Model Prompt

You are a meticulous <type> researcher conducting an important investigation into a certain neuron in a language model trained on <subject> papers. Your task is to figure out what sort of behaviour this neuron is responsible for – namely, on what general concepts, features, themes, methodologies or topics does this neuron fire? Here's how you'll complete the task:

INPUT DESCRIPTION: You will be given two inputs: 1) Max Activating Examples and 2) Zero Activating Examples.

1. You will be given several examples of text that activate the neuron, along with a number being how much it was activated. This means there is some feature, theme, methodology, topic or concept in this text that 'excites' this neuron.
2. You will also be given several examples of text that don't activate the neuron. This means the feature, topic or concept is not present in these texts.

OUTPUT DESCRIPTION: Given the inputs provided, complete the following tasks.

1. Based on the MAX ACTIVATING EXAMPLES provided, write down potential topics, concepts, themes, methodologies and features that they share in common. These will need to be specific - remember, all of the text comes from subject, so these need to be highly specific subject concepts. You may need to look at different levels of granularity (i.e. subsets of a more general topic). List as many as you can think of. Give higher weight to concepts more present/prominent in examples with higher activations.
2. Based on the zero activating examples, rule out any of the topics/concepts/features listed above that are in the zero-activating examples. Systematically go through your list above.
3. Based on the above two steps, perform a thorough analysis of which feature, concept or topic, at what level of granularity, is likely to activate this neuron. Use Occam's razor, as long as it fits the provided evidence. Be highly rational and analytical here.
4. Based on step 4, summarise this concept in 1-8 words, in the form FINAL: <explanation>. Do NOT return anything after these 1-8 words.

Here are the max-activating examples: <max activating examples>

Here are the zero-activating examples: <zero activating examples>

Work through the steps thoroughly and analytically to interpret our neuron.

Predictor Model Prompt

You are a <subject> expert that is predicting which abstracts will activate a certain neuron in a language model trained on <subject> papers. Your task is to predict which of the following abstracts will activate the neuron the most. Here's how you'll complete the task:

INPUT DESCRIPTION: You will be given the description of the type of paper abstracts on which the neuron activates. This description will be short. You will then be given an abstract. Based on the concept of the abstract, you will predict whether the neuron will activate or not.

OUTPUT DESCRIPTION: Given the inputs provided, complete the following tasks.

1. Based on the description of the type of paper abstracts on which the neuron activates, reason step by step about whether the neuron will activate on this abstract or not. Be highly rational and analytical here. The abstract may not be clear cut - it may contain topics/concepts close to the neuron description, but not exact. In this case, reason thoroughly and use your best judgement. However, do not speculate on topics that are not present in the abstract.
2. Based on the above step, predict whether the neuron will activate on this abstract or not. If you predict it will activate, give a confidence score from 0 to 1 (i.e. 1 if you're certain it will activate because it contains topics/concepts that match the description exactly, 0 if you're highly uncertain). If you predict it will not activate, give a confidence score from -1 to 0.
3. Provide the final confidence score in the form `PREDICTION: (your prediction)` e.g. `PREDICTION: 0.5`. Do NOT return anything after this.

Here is the description/interpretation of the type of paper abstracts on which the neuron activates: <description>

Here is the abstract to predict: <abstract>

Work through the steps thoroughly and analytically to predict whether the neuron will activate on this abstract.

C.3 Exploring the effectiveness of smaller models

Although we eventually used `gpt-4o-mini` as the Predictor model, we initially did some ablations to understand how effective `gpt-4o` and `gpt-3.5-turbo` would be as different combinations of the Interpreter and Predictor models. We measured this by randomly sampling 50 features from our SAE64 (trained on `astro-ph` abstracts) and measuring the interpretability scores of different model combinations, in terms of both F1 score (does the model's binary classification of a feature firing on an abstract agree with the ground-truth) and the Pearson correlation (described in the main body). Interestingly, we observe that using `gpt-4o` as the Interpreter and `gpt-3.5-turbo` as the Predictor leads to similar scores as using `gpt-3.5-turbo` for both, as shown in Figures 13 and Figures 14. This suggests that the challenging task in the autointerp is not necessarily labelling but rather predicting the activation of a feature on unseen abstracts.

Another observation is that using `gpt-3.5-turbo` as the Predictor only leads to a moderate degradation of F1 score, it leads to a significant degradation of Pearson correlation. This is likely because we only use 6 abstracts for each feature prediction (3 positive, 3 negative) and thus there are only a few discrete F1 scores possible. Additionally, it appeared that `gpt-3.5-turbo` was generally less likely to assign higher confidence scores in either direction, with a much lower variance in assigned confidence than when `gpt-4o` was the Predictor. This affects Pearson correlation but not F1.

D Cross-domain features

The intersection between our `cs.LG` ($n = 153,146$) and `astro.PH` ($n = 271,492$) corpora contains $n = 330$ cross-posted papers. Motivated by these papers, as well as the observation of similar features re-occurring in models of different sizes (see Section 4), we search for the max cosine similarity feature between `cs.LG` and `astro.PH` SAEs at a fixed k and n_{dir}. As expected, we find

Figure 13: Correlation between F1 scores and Pearson correlation scores of different combinations of `(labeller, predictor)` models. Interestingly, using GPT-3.5 as the predictor appears to degrade performance similarly regardless of whether the feature was labelled by GPT-4o or GPT-3.5.

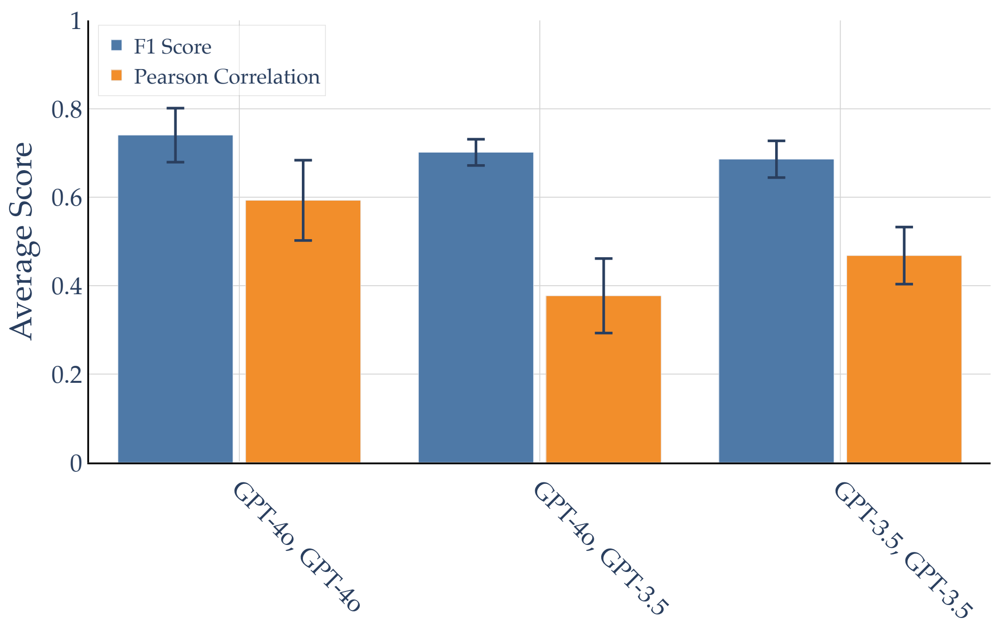


Figure 14: Mean F1 scores and Pearson correlations (according to ground-truth feature activations) across 50 randomly sampled features, for different combinations of `(Interpreter, Predictor)` models.

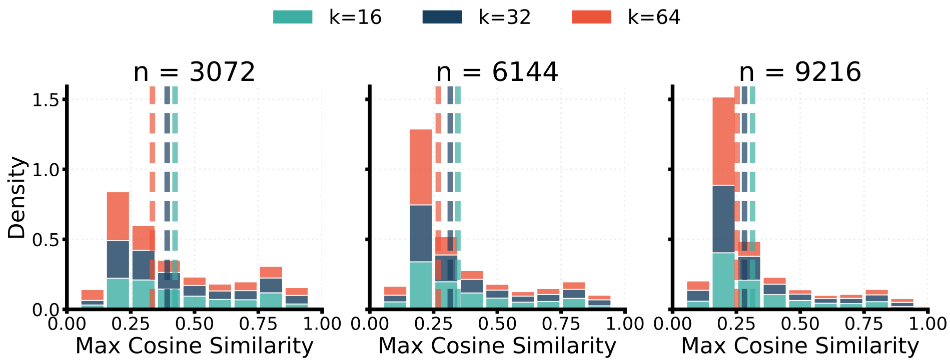


Figure 15: Maximum pair-wise cosine similarity of feature vectors between SAEs trained on different domains.

Feature Name (`astro-ph`)	Best Match (`cs.LG`)	Cosine Sim.	Activation Sim.	Δ F1	Δ Pearson
Deep learning	CNNs and Applications	0.39	0.33	-0.2	-0.17
Generative Adversarial Networks	Generative Adversarial Networks (GANs)	0.61	0.26	0	0
Transformers	Transformer architectures and applications	0.5	0.33	0	-0
Artificial Neural Networks	Artificial Neural Networks (ANNs)	0.64	0.02	0	0
Artificial Intelligence	AI applications in diverse domains	0.61	0.45	0	0.02
Automation and Machine Learning	Automation in computational processes	0.9	0.77	-0.25	-0.47
Gaussian Processes	Gaussian Processes in Machine Learning	0.59	0.54	0	0.03
Regression analysis	Regression techniques and applications	0.81	0.53	0	-0.01

Table 4: Feature matches from the "Machine Learning" family (`astroPH`); $k = 64, n_{dir} = 9216$.

significant mis-alignment between the vast majority of feature vectors between SAEs trained on different domains, with mis-alignment increasing with k and n_{dir} (see Figure 15; this is unsurprising given how k and n_{dirs} correlate with feature granularity).

However, a small subset of features appear in both sets of SAEs, with relatively high max cosine similarity. For example, Table 4 shows the nearest `cs.LG` neighbours for every feature in the `astro.PH` "Machine Learning" feature family (average cosine similarity = 0.59, average activation similarity = 0.40). To test whether the features represent the same semantic concepts, we substitute the natural language description of the best-match `cs.LG` feature for each listed `astro.PH` feature and test the interpretability of the substituted descriptions; we find $\Delta_{\text{Pearson}} = -0.07$ and $\Delta_{F1} = -0.06$. The existence of these features suggests that both sets of SAEs learn a semi-universal set of features that span the domain overlap between `astro.PH` and `cs.LG`.

Interestingly, we find a number of near-perfectly aligned pairs (cosine similarity > 0.95) of highly interpretable features with little semantic overlap. A number of these features share similar wording but not meaning, such as "Substructure in dark matter and galaxies" (`astro-ph`) and "Subgraphs and their representations". Of these 10 feature pairs, the average activation similarity is 0.91.

E Feature family details

E.1 Feature splitting structures

Figure 16 shows an example of a recurrent feature across SAE sizes that does not exhibit feature splitting. While the feature has extremely high activation and cosine similarity across every model pair, each model only learns 1 feature in this direction. In Figures 17a and 17b we show two examples of feature splitting across SAE16 – SAE32 – SAE64 trained on `astro-ph`. 17a appears to show canonical feature splitting as originally described in Bricken et al., 2023, with an increasing number of features splitting the semantic space at each SAE size. There exists a top-level "periodicity"/"periodicity detection" feature universal to all three SAEs, with relatively high similarity to all other features, as well as novel, more granular features appearing in smaller SAEs, i.e. "Quasi-periodic oscillations in blazars", which only appears in SAE64 and is highly dissimilar from other split features.

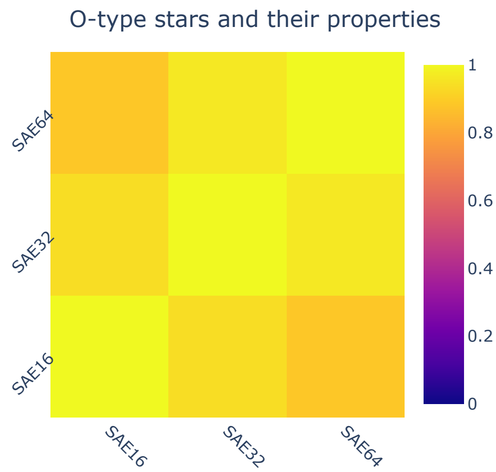


Figure 16: Recurrent features across SAEs trained on `astro-ph`; heatmap colored by activation similarity D; all feature vector cosine similarities are > 0.98.

In contrast, 17b demonstrates nearest-neighbour features across models that do not exhibit semantically meaningful feature splitting. While the top-level "Luminous Blue Variables (LBVs)" feature occurs at every model size, SAE64 also exhibits two additional features, "Lemaitre-Tolman-Bondi (LTB) Models" and "Lyman Break Galaxies (LBGs)", that are highly dissimilar to each other, the LBVs feature, and every other feature in the smaller models. We claim these are novel features, occurring for the first time in SAE64, and that SAE16/SAE32 do not learn features for any related higher-level concepts; instead, this grouping could be a spurious token-level correlation (LBV/LTB/LBG as similar acronyms).

Feature triplets In Figure 18a, we search for features that occur in $n_{dirs} = 3072$ models and have highly aligned features in larger ($n_{dirs} = 6144, 9216$) models; we use this as a rough proxy for the number of re-occurring features. We find that significantly more features re-occur between models for higher k, with over 1100 feature triplets at > 0.95 cosine similarity for $k = 16$; as k increases, the number of triplets drops sharply.

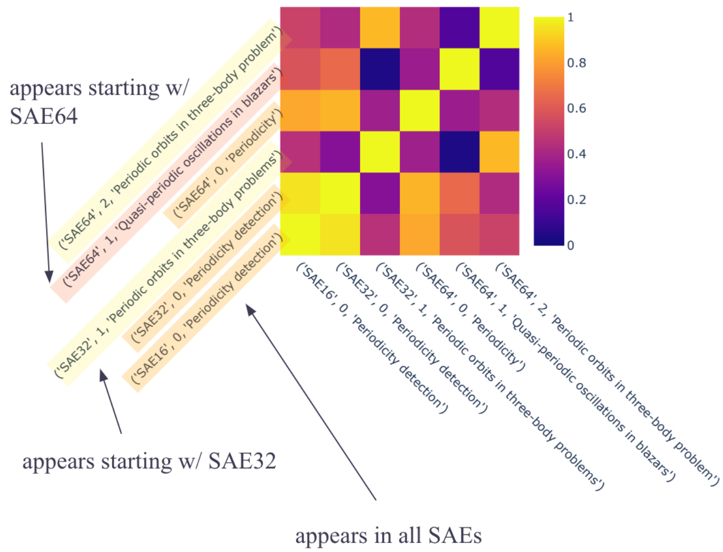
Self-consistency In 18b we show the set overlap between nearest-neighbour matches between SAE16 and SAE64 found directly, and nearest-neighbour matches between SAE16 and SAE64 found via nearest-neighbour matches to SAE32. If features exhibit perfectly clean splitting geometry, then these two sets of SAE64 features should be consistent. However, we find that the distribution of set overlap is roughly bimodal; other than triplet features with perfect overlap, overlap generally ranges from 0 to 0.6. The vast majority of intersection = 1 sets are ≤ 3 features in size. This corroborates findings in 9 which suggests features across models with different k are not well-aligned.

E.2 Feature family structure

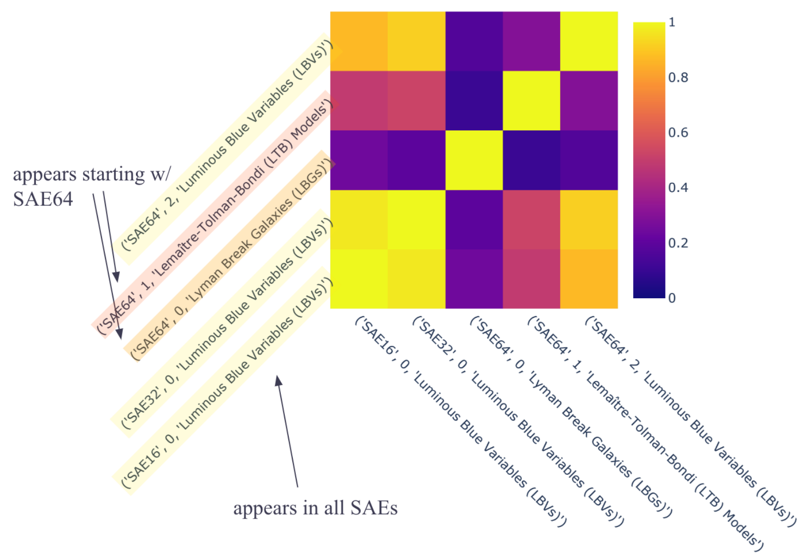
We compute feature family sizes (including the parent), co-occurrence ratios ($\overline{R(p, \mathcal{C})}$, see section 4), and activation similarity ratios (computed identically to $\overline{R(p, \mathcal{C})}$, just using activation similarities). Statistics for variants of `cs.LG` and `astro-ph` are shown in 19. We find a positive correlation (Spearman = 0.22) between $\overline{R(p, \mathcal{C})}$ and feature family interpretability.

We reproduce the projection method of Engels et al., 2024, running all documents through the SAE and ablating features not in the feature family, to produce Figure 20. Visualizing the resulting principal components confirms that the feature families we find do not represent manifolds or irreducible multi-dimensional structures. We can instead think of feature families as linear subspaces in the high-dimensional latent space; in fact, the component vectors can be seen in the lines of points representing documents only activating on one feature in the family.

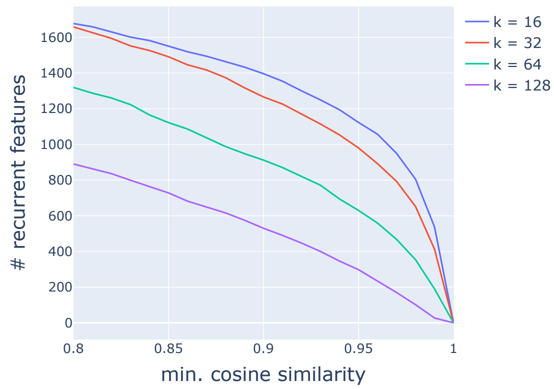
In 4 we use $n = 3$ iterations of feature family construction. We select this hyper-parameter based off Figure 21. In the first 2-3 iterations, removing parent nodes and re-constructing features preferentially



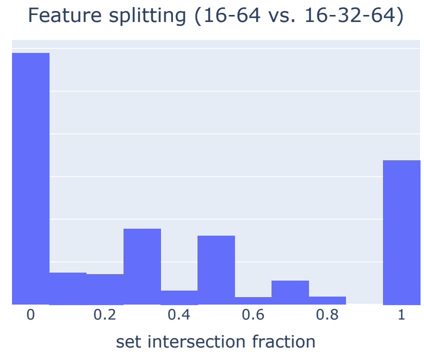
(a) We find both recurrent features and novel features at every level (i.e. the top-level “periodicity detection"/“periodicity" feature); heatmap colored by pairwise cosine similarity.



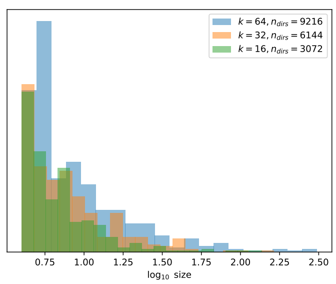
(b) While “Luminous Blue Variables" is a recurrent feature in each SAE, SAE64 also exhibits 2 other nearest-neighbour features to “Luminous Blue Variables" that are not semantically related; heatmap colored by pairwise cosine similarity.



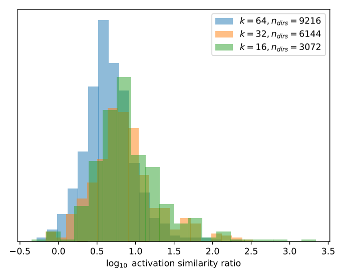
(a) Number of features from the smallest SAE that re-occur in all SAEs, by cosine similarity threshold.



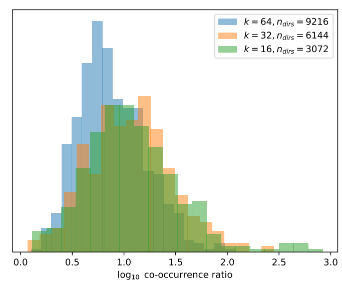
(b) Overlap in the recovered SAE64 features, propagating nearest neighbors from SAE16-SAE64 vs. SAE16-SAE32-SAE64.



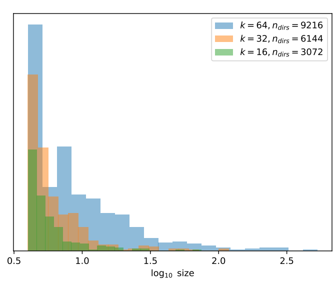
(a)



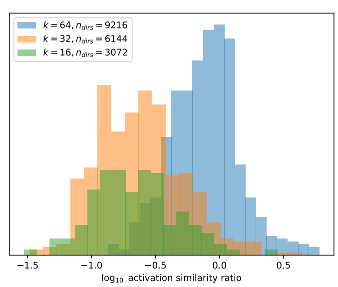
(b)



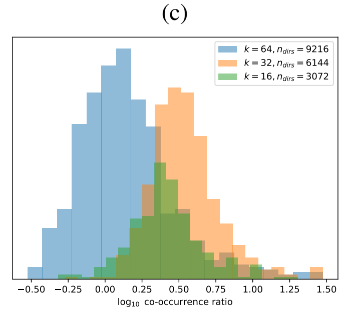
(c)



(d)



(e)



(f)

Figure 19: Feature families statistics (left: size; middle: activation similarity ratio; right: co-occurrence ratio, $\overline{R(p, \mathcal{C})}$); $k = 64, n_{dir} = 9216$.

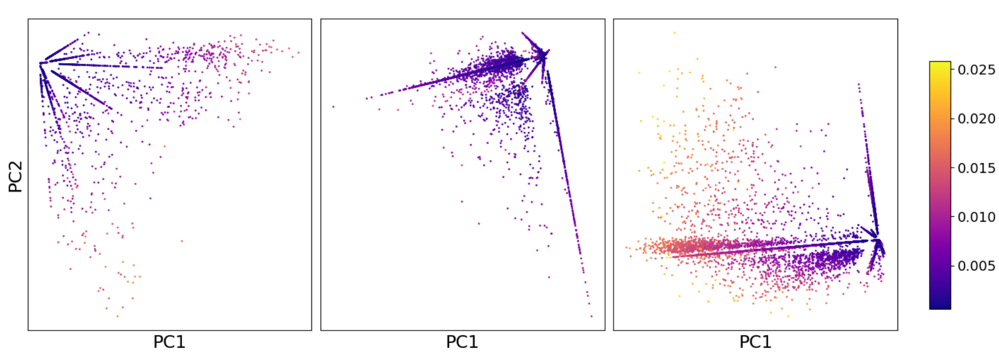


Figure 20: PCA projections of 3 example feature families from SAE64; points are latent representations of activating examples, colored by average activation for in-family features in the top k.

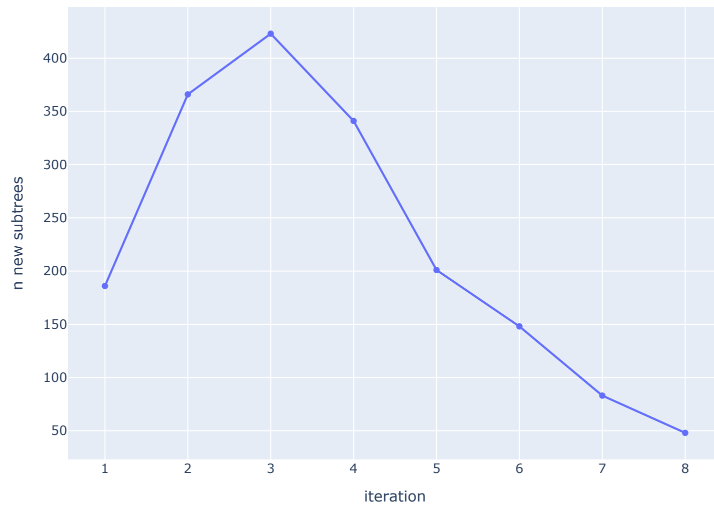


Figure 21: New feature families as a function of iteration; no deduplication is performed.

creates additional smaller families, suggesting iterations are necessary to fully explore the graph. But given the sparse co-occurrences ($C_{i,j} > 0.1$) used to build the graph, the number of additional feature families found at each iteration drops off steeply after $n = 3$.

E.3 Feature family interpretability

We show example feature families and their interpretability scores in Figure 22.

F Exploring learned decoder weight matrices

Encoder and decoder representations Figure 23 reveals an intriguing relationship between feature distinctiveness and the similarity of encoder and decoder representations in our sparse autoencoder. In an ideal scenario with orthogonal features, encoder and decoder vectors would be identical, as the optimal detection direction (encoder) would align perfectly with the representation direction (decoder). This is because orthogonal features can be uniquely identified without interference. However, in our high-dimensional space with more features than dimensions, perfect orthogonality is impossible due to superposition.

The right panel of Figure 23 shows a negative correlation between a feature's decoder-encoder cosine similarity and its maximum similarity with other features. Features more orthogonal to others (lower maximum similarity) tend to have more similar encoder and decoder representations. This aligns with intuition: for more isolated features, the encoder's detection direction can closely match the decoder's representation direction. Conversely, features with higher similarity to others require the encoder to adopt a more differentiated detection strategy to minimise interference, resulting in lower encoder-decoder similarity. The left panel, showing a mean cosine similarity of 0.57 between corresponding encoder and decoder vectors, further emphasises this departure from orthogonality. This phenomenon points to the importance of untied weights in sparse autoencoders.

Clustering feature vectors Motivated by structure in the feature activation graph, we explore whether similar structure can be found in the decoder weight matrix W itself. Gao et al., 2024 find 2 such clusters; we reproduce their method across our embeddings and SAEs, permuting the left singular vectors U of W using a one-dimensional UMAP. We also experiment with permuting U and W using reverse Cuthill-McKee. We do not find any meaningful block diagonal structure or clustering in W.

G Iterative encoding optimisation

We noted in Section 5 that intervening on a feature by up- or down-weighting its hidden representation and then decoding is equivalent to directly adding the scaled feature vector to the final embedding.

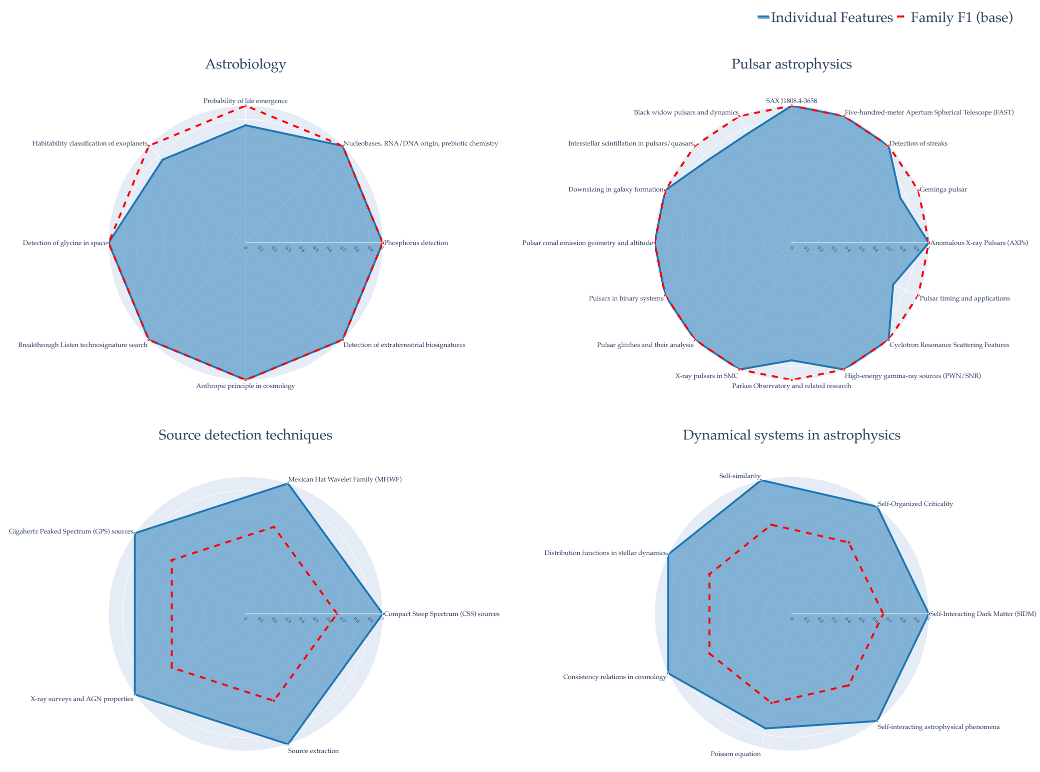


Figure 22: High-quality (top) and low-quality (bottom) feature families, scored through automated interpretability; radar charts show Pearson correlation scores for individual features (vertices) and the overall family (dashed line). While high-quality feature families truly have shared meaning, low-quality families appear to be mostly spurious and are not interpretable through short descriptions.

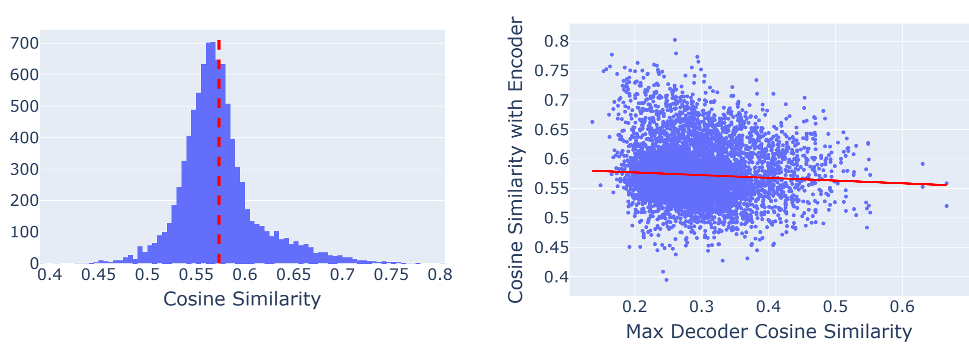


Figure 23: (Left) Cosine similarities between the encoder row and corresponding decoder column for SAE64 (`cs.LG`). The mean cosine similarity is 0.57, suggesting that encoder and decoder features are rather different, agreeing with Nanda (2023). (Right) We notice a slight negative correlation between a feature's decoder-encoder cosine similarity, and its maximum similarity with other features, possibly suggesting that features that are furthest removed from all other features in embedding space can have more similar corresponding decoders and encoder projections.

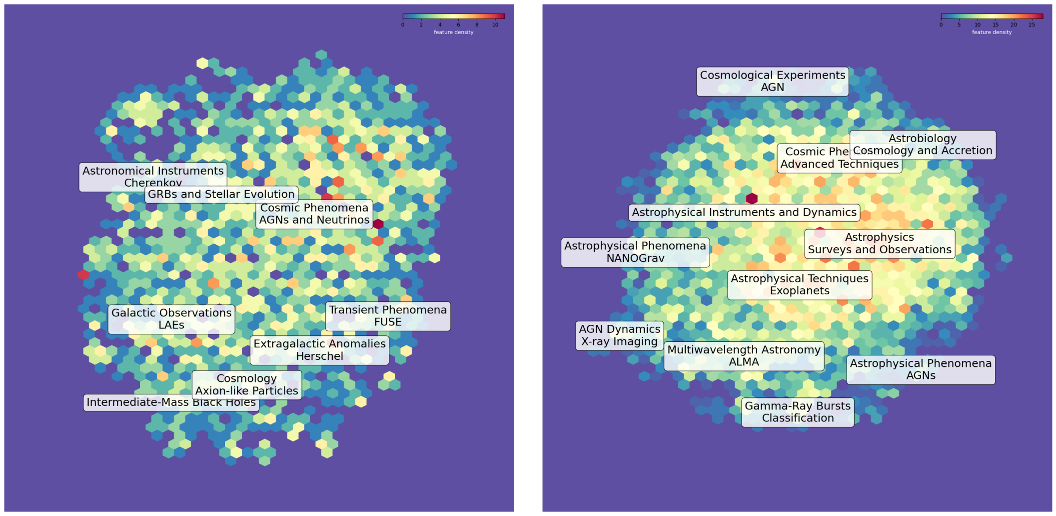


Figure 24: UMAP density plots along with LLM generated labels for SAE16 (left) and SAE64 (right) for the `astro-ph` features.

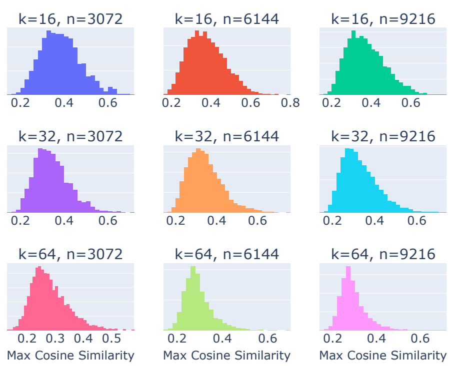


Figure 25: Distribution of maximum cosine similarity between a given feature vector and all other feature vectors, within the same SAE.

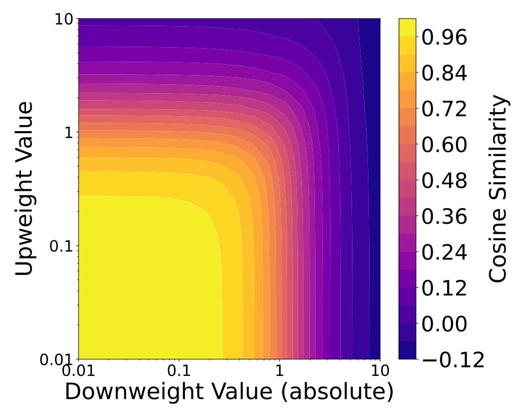


Figure 26: Cosine similarity between the original query embedding and the modified query embedding, with different values of upweighting random zero features and downweighting random active features.

To demonstrate this equivalence, let's consider an intervention on feature i by an amount δ. The modified hidden representation is $\mathbf{h}' = \mathbf{h} + \delta\mathbf{e}_i$, where $\mathbf{e}_i$ is the i-th standard basis vector. Decoding this modified representation gives $\hat{\mathbf{x}}' = W_d\mathbf{h}' = W_d\mathbf{h} + \delta W_d\mathbf{e}_i = \hat{\mathbf{x}} + \delta\mathbf{w}_i$, where $\mathbf{w}_i$ is the i-th column of W_d. Thus, intervening on the hidden representation and then decoding is equivalent to directly adding the scaled feature vector to the original reconstruction.

We show in Figure 26 how cosine similarity between the original query embedding and the modified query embedding changes as we change the upweighting and downweighting strength for different features. Cosine similarity drops rapidly as soon as upweight or downweight exceeds 0.1.

There is an implicit challenge in SAE-based embedding interventions: the trade-off between steering strength and precision. When directly manipulating feature activations, we observed that strong interventions often led to unintended semantic shifts, activating correlated features and potentially moving the embedding far from the SAE's learned manifold. Our goal is to achieve precise semantic edits that express the desired feature strongly while minimising interference with unrelated features. To this end, we developed an iterative optimisation approach that leverages the SAE's learned feature space to find an optimal balance between these competing objectives.

Let $\mathbf{x} \in \mathbb{R}^d$ be the original embedding, $f_\theta(\cdot)$ the SAE encoder, and $g_\phi(\cdot)$ the SAE decoder. We define a target feature vector $\mathbf{t} \in \mathbb{R}^k$ representing the desired feature activations after intervention, where k is the number of active features in our SAE. The iterative latent optimisation aims to find optimised latents $\mathbf{h}^*$ that satisfy:

$$\mathbf{h}^* = \text{argmin}_{\mathbf{h}'} \left\{ \| f_\theta(g_\phi(\mathbf{h}')) - \mathbf{t} \|_2^2 \right\}$$

We solve this optimisation problem using gradient descent, starting from the initial latents $\mathbf{h} = f_\theta(\mathbf{x})$ and iteratively updating $\mathbf{h}'$. We use the AdamW optimiser with a cosine annealing learning rate schedule.

To evaluate the effectiveness of this approach, we compare it to a direct intervention method where we simply set the target feature to a specific value in the latent space. For each abstract in our dataset, we embed the abstract using an OpenAI embedding model to obtain $\mathbf{x}$. We then encode the embedding to get initial latents $\mathbf{h} = f_\theta(\mathbf{x})$. We randomly select a target feature i and target value v. We then apply both intervention methods: our iterative optimisation of $\mathbf{h}'$ as described above, with $\mathbf{t}_i = v$ and $\mathbf{t}_j = \mathbf{h}_j$ for $j \neq i$, and direct intervention: setting $\mathbf{h}'_i = v$ and $\mathbf{h}'_j = \mathbf{h}_j$ for $j \neq i$.

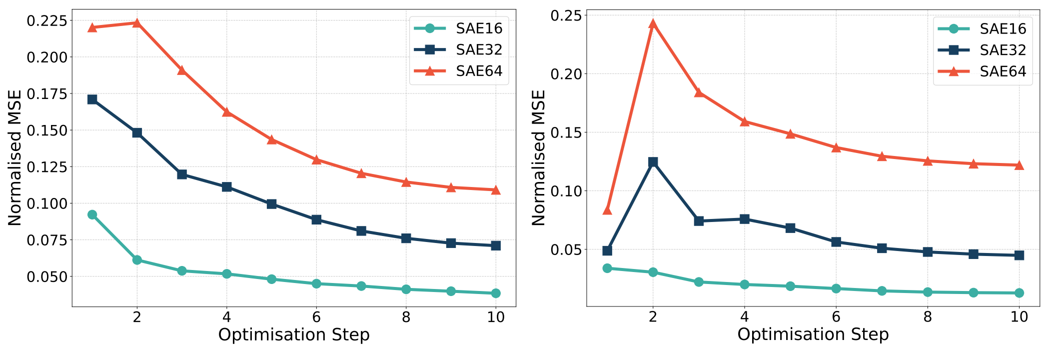


Figure 27: Normalised MSE at each of 10 steps across the iterative latent optimisation process. Left: Setting a random zero feature to active. Right: Setting a random active feature to zero.

Figure 27 (left panel) shows the trajectory of normalised MSE during the iterative optimisation process, when setting a random zero feature to active. Similarly, the right panel shows the optimisation when setting a random active feature to zero. Normalised MSE improves in the former case but not the latter.