# Sparse autoencoders for dense text embeddings reveal hierarchical feature sub-structure

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Sparse autoencoders (SAEs) show promise in extracting interpretable features from complex neural networks, enabling examination and causal intervention in the inner workings of black-box models. However, the geometry and completeness of SAE features is not fully understood, limiting their interpretability and usefulness. In this work, we train SAEs to detangle dense text embeddings into highly interpretable document-level features. Our SAEs follow precise scaling laws as a function of capacity and compute, and exhibit significantly higher interpretability scores compared to SAEs trained on language model activations. In embedding SAEs, we reproduce qualitative "feature splitting" phenomena previously reported in language model SAEs, and demonstrate the existence of universal, cross-domain features. Finally, we suggest the existence of "feature families" in SAEs, and develop a method to reveal distinct hierarchical clusters of related semantic concepts and map feature co-activations to a sparse block diagonal.

## 1 Introduction

Sparse autoencoders (SAEs) have emerged as a promising approach to neural network interpretability (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 previously shown success in analysing and steering generation, and has led to new insights on the inner workings of language models, while also motivating a number of empirical questions about SAE features and mechanisms (Conmy et al., 2024; Cunningham et al., 2023b; Bricken et al., 2023; Lieberum et al., 2024). In this work, we present the first application of SAEs to dense text embeddings derived from large language models. We empirically examine the interpretability, scalability, and feature structure of SAEs trained over text embeddings. Our research makes the following key contributions:

1. We demonstrate the effectiveness of SAEs in learning document-level features from dense representations. We examine their interpretability, scaling behavior, and feature geometry.
2. We introduce SAE "feature families", hierarchical clusters of features that allow for multi-scale semantic analysis and manipulation, and methodology for finding and verifying families. We also examine the proliferation of "split" features across levels of abstraction.

## 2 Background and Related work

**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 by representing many sparse concepts, or *features*, in superposition (Elhage et al., 2022a). Distributed representations allows models to efficiently encode a large number of

Submitted to Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

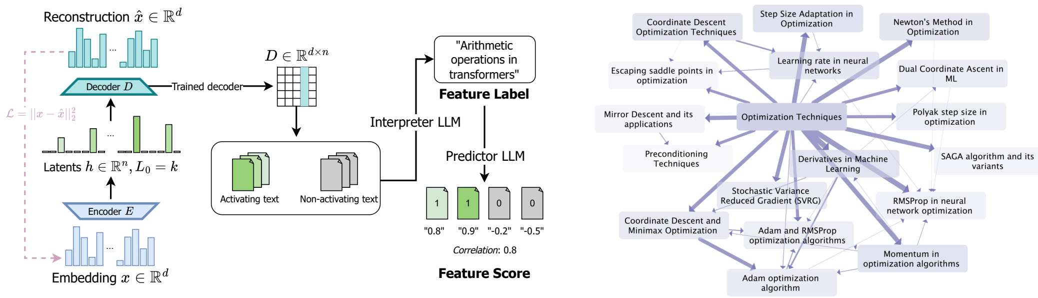



Figure 1: Left: SAE training and labelling. Right: `cs.LG` feature family; arrows represent $C > 0.1$.

features in a relatively low-dimensional space, but it also makes model layers challenging to interpret directly. Sparse autoencoders (SAEs) address this by learning to reconstruct inputs using a sparse set of features in a higher-dimensional space, encouraging disentanglement of distributed representations (Elhage et al., 2022b; Donoho, 2006; Olshausen et al., 1997). When applied to language model activations, SAEs recover semantically meaningful and human-interpretable sparse features (Gao et al., 2024; Bricken et al., 2023; Cunningham et al., 2023b). A number of automated interpretability approaches have been proposed and applied, such as Bills et al. (2023) and Foote et al. (2023).

**Structure in SAE features** A large volume of interpretable features have been discovered in SAEs trained over language models (Cunningham et al., 2023a; Bricken et al., 2023; Lieberum et al., 2024). This has motivated work studying the underlying 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 with SAE capacity. Engels et al., 2024 find clusters of SAE features that represent inherently multi-dimensional, non-linear subspaces.

# 3 Training SAEs and automated labelling

We trained top-$k$ Sparse Autoencoders (SAEs) on embeddings of arXiv abstracts from astrophysics (`astro-ph`, 272,000 papers) and computer science (`cs.LG`, 153,000 papers), using OpenAI's `text-embedding-3-small` model. We experimented with hyperparameters, focusing primarily on SAEs with $k = 16$, 32, and 64 active latents. To interpret 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 reconstruction ability using normalized mean squared error, and feature interpretability using Pearson correlation. Detailed training procedures, hyperparameters, and evaluation metrics are provided in Appendix A.

**Scaling performance:** Templeton, 2024 found compute-optimal scaling laws for SAEs over language model activations. Similarly, we observe precise ($R^2 > 0.93$) power-law scalings 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 $\alpha$ ranges from 0.12 to 0.18 and increases with $k$; `cs.LG` shows slightly higher $\alpha$ values compared to `astro-ph`. 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, decreasing with $k$. Figures and detailed fits are provided in Appendix A, Figure 4.

**Interpretability:** We find high correlation between predictor model confidence and the ground-truth firing, with median Pearson correlations ranging from 0.65 to 0.71 for `cs.LG` and 0.85 to 0.98 for `astro-ph`; see B for details. Bricken et al. (2023) report a median feature Spearman correlation of 0.58 from an SAE trained on MLP activations. Scores increase as $k$ and $n$ 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

SAEs trained over arXiv paper embeddings recover a wide range of features covering both scientific concepts, from niche to multi-disciplinary, and also abstract semantic artifacts, such as humorous writing or critiques of scientific theories; see Appendix B for detailed examples.

Building off previous work on the structure of SAE features and learned representations, we examine two distinct empirical 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 for language model activations (Bricken et al., 2023; Makelov et al., 2024; Bussmann et al., 2024). Examples of feature splitting, as well as features recurring across SAEs, can be found in Appendix D (Figs. 11 and 12a/12b). In contrast, *feature families* exist within a single SAE. Unlike SAE feature clusters found in other works (Daujotas, 2024; Engels et al., 2024), *feature families* empirically 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; see Figure 1 for an example.

## 4.1 Feature splitting

We study the proliferation of features in small to large SAEs using a nearest neighbour approach. For each pair of SAEs, we calculated the similarity matrix $S$ from decoder vectors $\mathbf{w}$, where $S_{ij} = \mathbf{w}_{1,i}^T \mathbf{w}_{2,j} / \|\mathbf{w}_{1,i}\| \|\mathbf{w}_{2,j}\|$. Given feature $j$ in the larger SAE, we identified the nearest neighbour in the smaller SAE, tracing how features "split" as model capacity increases. We find that increasing both active latents $k$ and latent dimension $n$ reduces the similarity between nearest neighbours. This matches intuition: larger models with more capacity (higher $k$ and $n$) may learn more fine-grained and specialised features, leading to greater differentiation. See Appendix A.4 (Figure 6).

Empirically, matching features from small to large SAEs, we detect both *recurrent* features and *novel* features. Recurrent features exhibit high $S_{ij}$ and activation similarity across model pairs, and have highly similar interpretations. These are much more common for lower $k$; in SAE16, >1100 out of 3216 features match features in both SAE32 and SAE64 at >0.95 similarity (see Figure 11). We also find $\sim$dozens of semantically close features with similar activation patterns appearing in both `cs.LG` and `astro-ph` SAEs (see C). Novel features span narrower semantic meaning than their nearest-neighbour match, and exhibit lower $S$, activating similarly on a document subset; they *split* the semantic space covered by a single feature from the smaller SAE. Some "novel" features share little semantic or activation overlap with their nearest-neighbour feature, as in Fig. 12b, indicating smaller SAEs may not sufficiently cover the feature space; see D.1 in the Appendix for more details.

## 4.2 Feature families

**Feature family identification** We identified feature families using a graph-based approach based co-activation patterns. We first compute co-occurrence matrix $C$ and activation similarity matrix $D$. For all data points $k$, $C_{ij} = \sum_k A_{ik} A_{jk}$ and $D_{ij} = \sum_k B_{ik} B_{jk}$ 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, obtaining $C_{ij}^{thresh}$ (hereafter just $C$). We construct a maximum spanning tree (MST) from $C$ and convert it to a graph directed by density, representing a hierarchy from more general to more specific concepts. Feature families $F$ are then constructed via depth-first-search in this directed graph, starting from root nodes and recursively exploring hierarchical sub-families. This process is then iterated with deduplication, removing parent features after each iteration to reveal new families. In practice, we use only highly interpretable features (Pearson $\geq 0.8$), choose $\tau = 0.1$, and run $n = 3$ iterations; see D for details. Finally, using the method from Section 3, we generated a "superfeature" description for each family, and assessed the family's interpretability using high-activating examples sampled across all child features.

**Matrix structure** We conjecture that feature families are equivalent to diagonal blocks in some permutation of co-occurrence matrix $C$ and activation similarity matrix $D$; then when permuted, 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

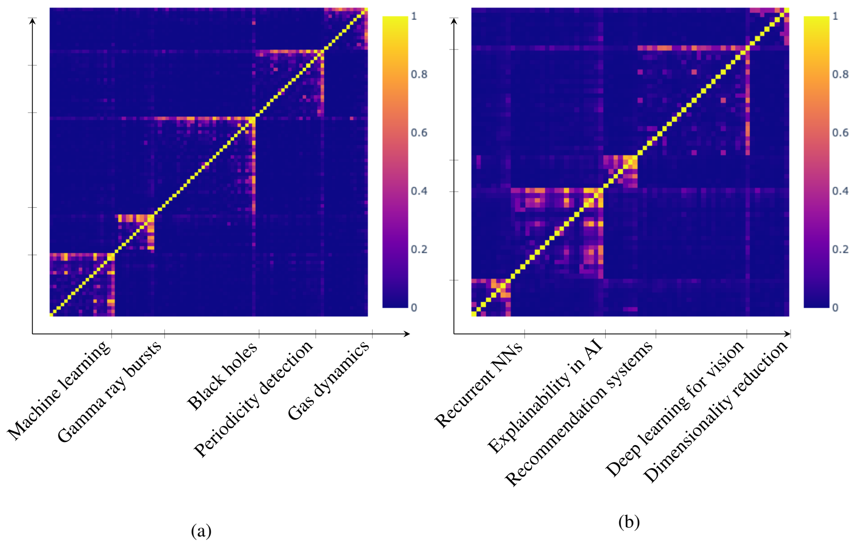



Figure 2: Co-occurrence matrix $C$ organised by a subset of feature families; the right-most feature is the parent, and child features are ordered by increasing density.

features all co-occur with the parent feature but rarely co-occur with one another. Motivated by this structure, 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 $C$ and $D$ by greedily selecting interpretable 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 diagonal), capturing the clustering strength of the block diagonal. Median values are listed in Table 1; subsets of $C$, permuted by feature family, are shown in 2.

| **Dataset** | $(k, n)$ | Size | F1 | Pearson | $\overline{R(p, \mathcal{C})}$ | $C_{\text{diag}}/C_{\text{off}}$ | $D_{\text{diag}}/D_{\text{off}}$ | $f_{\text{inc}}$ |
|---|---|---|---|---|---|---|---|---|
| `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 | 0.70 | 6.87 | 2.00 | 3.05 | 0.24 |
| `cs.LG` | (16, 3072) | 5 | 0.73 | 0.60 | 2.44 | 8.35 | 0.89 | 0.23 |
| | (32, 6144) | 5 | 0.73 | 0.59 | 3.50 | 7.33 | 1.07 | 0.30 |
| | (64, 9216) | 7 | 0.80 | 0.71 | 1.22 | 1.78 | 2.57 | 0.41 |

Table 1: Feature family metrics. $f_{inc}$ is the fraction of features in a clean family (Pearson $\geq 0.8$).

# 5 Discussion

In this work, we presented the first application of sparse autoencoders to dense text embeddings, and an empirical assessment of the scaling, interpretability, and substructure of learned sparse features. Using state-of-the-art automated interpretability and training approaches, we demonstrated that SAEs are extremely effective on text embeddings, producing highly interpretable features and exhibiting precise scaling laws. Our analysis of different-scale SAEs confirms other empirical observations of feature splitting, demonstrating that features may be semantically split, recurrent, or entirely novel. Furthermore, our analysis introduces the concept of and search methodology for "families" in SAE features, which may allow for multi-scale semantic analysis and causal manipulation. We confirm the existence of "feature families" in our SAEs and demonstrate that these hierarchical clusters are reflected in the block diagonalization of co-occurrence and activation data. This motivates a number of new directions in multi-scale feature discovery, interpretation, and manipulation.

**Limitations:** Our work focused only on embeddings from relatively small datasets of scientific abstracts; the feature splitting and feature family phenomena differed even between domains `cs.LG` and `astro-ph`. Future work should investigate how well these methods generalise, and SAEs for more diverse embedding datasets would need to be scaled up by at least 2-3 the total number of latents. Moreover, the completeness of our learned dictionaries remains an open question; future work should evaluate SAE features from text embeddings against some proxy of ground-truth features, as proposed by Makelov et al. (2024). Finally, our analysis should be considered complementary to other methods that discover non-hierarchical features, such as non-linear manifolds.

# References


Bills, Steven, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders (2023). "Language models can explain neurons in language models". In: *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)* 2.

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

Bussmann, Bart, Patrick Leask, Joseph Isaac Bloom, Curt Tigges, and Neel Nanda (July 2024). *Stitching SAEs of Different Sizes*. Online. AI Alignment Forum. URL: https://www.alignmentforum.org/posts/baJyjpktzmcmRfosq/stitching-saes-of-different-sizes.

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 (Aug. 2024). *Case Study: Interpreting, Manipulating, and Controlling CLIP With Sparse Autoencoders*. Online. LessWrong. URL: https://www.lesswrong.com/posts/iYFuZo9BMvr6GgMs5/case-study-interpreting-manipulating-and-controlling-clip.

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.

Foote, Alex, Neel Nanda, Esben Kran, Ioannis Konstas, Shay Cohen, and Fazl Barez (2023). "Neuron to graph: Interpreting language model neurons at scale". In: *arXiv preprint arXiv:2305.19911*.

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*.

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*.

Lieberum, Tom, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda (2024). *Gemma Scope: Open Sparse Autoencoders Everywhere All At Once on Gemma 2*. arXiv: 2408.05147 [cs.LG]. URL: https://arxiv.org/abs/2408.05147.

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*.

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.

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.

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*.

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

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.

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:

$$\text{Encoder}: \quad \mathbf{h} = f_\theta(\mathbf{x}) = \sigma(W_e\mathbf{x} + \mathbf{b}_e) \tag{1}$$
$$\text{Decoder}: \quad \hat{\mathbf{x}} = g_\phi(\mathbf{h}) = W_d\mathbf{h} + \mathbf{b}_d \tag{2}$$

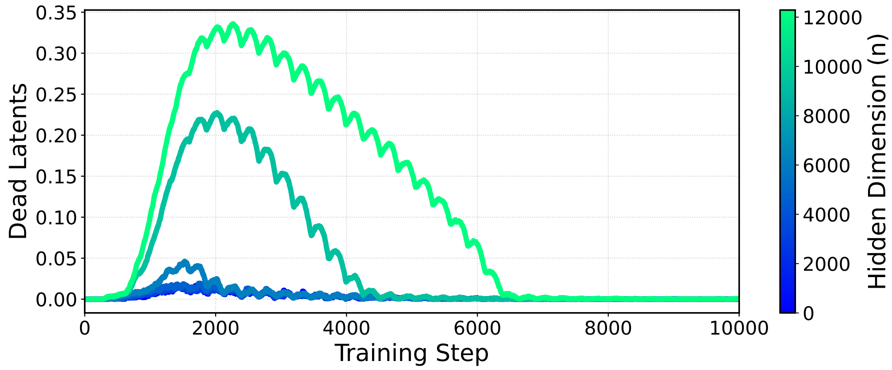



Figure 3: The proportion of dead latents, defined as features that haven't fired in the last epoch of training, for our $k = 16$ SAEs on the `astro-ph` abstract embeddings. All dead latents were gone by the end of training. We found that dead latents only occurred in $k = 16$ autoencoders.

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 augment the primary loss with an auxiliary component (AuxK), inspired by the "ghost grads" approach of Jermyn et al. (2023). This auxiliary term considers the top-$k_{aux}$ inactive latents (typically $k_{aux} = 2k$), where inactivity is determined by a lack of activation over a full training epoch. The total loss is formulated as $\mathcal{L} + \alpha \mathcal{L}_{aux}$, with $\alpha$ usually set to 1/32. This mechanism reduces the number of dead latents with minimal computational overhead (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 3.

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

**Data:** 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 are available for download here.

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

**Hyperparameters:** Notable hyperparameters include the number of active latents $k$, the total number of latents $n$, the number of auxiliary latents $k_{\text{aux}}$, the learning rate, and the auxiliary loss coefficient $\alpha$. 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 $d_{\text{input}}$. Whilst we train SAEs with many 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 B). 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. 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 $n$.

### A.3 Scaling laws

For the left panel of Figure 4, which shows the scaling of normalised MSE with the number of total latents $n$, we observe the following power-law relationships:

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

For the right panel of Figure 4, which shows the scaling of normalised MSE with the amount of compute $C$ (in FLOPs), we observe the following power-law relationships:

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

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

[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 |

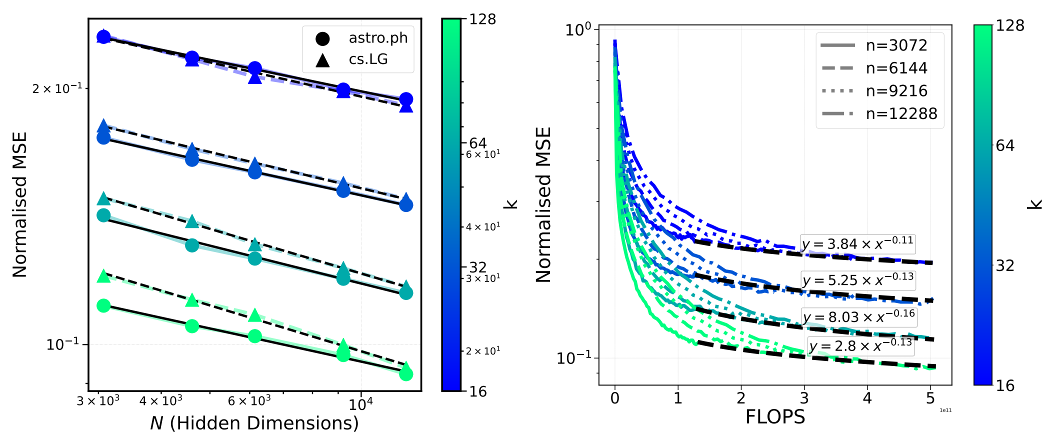



Figure 4: Scaling laws for sparse autoencoder performance. Left: Normalised mean squared error (MSE) as a function of the number of total latents $n$ for different values of active latents $k$. The power-law scaling is evident for each $k$. Right: Reconstruction loss as a function of compute (FLOPs) for different $k$ values, demonstrating the compute-optimal model size scaling.

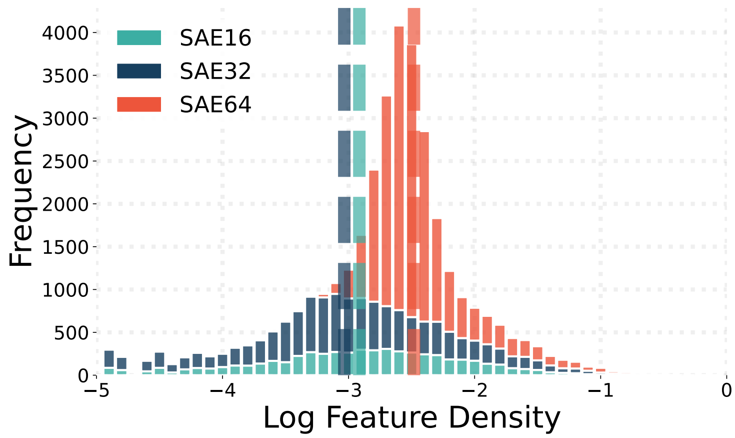



Figure 5: Log feature density for features in our three SAEs as a stacked histogram, showing the distribution of how often features fire across all paper abstacts (`cs.LG` and `astro-ph`). The larger SAE has a higher mean feature density than the smaller SAEs.

### A.4 Feature density and similarity

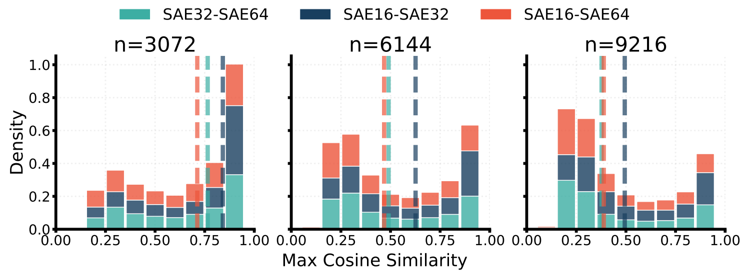
We find an intuitive relationship between $k$ and $n$ and the log feature density (essentially, how often a given feature fires). As $k$ increases, we get a sharper peak of log feature density, shifted to the right, suggesting features fire in a tighter range as we increase the instantaneous L0 of the SAE's encoder (Figure 5).

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; see 6. 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.

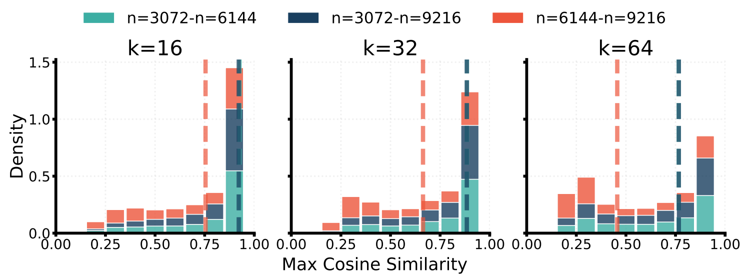
## B Automated interpretability details

### B.1 Examples of features

Most SAE features are highly interpretable; see 7. 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.




(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 6: 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`.

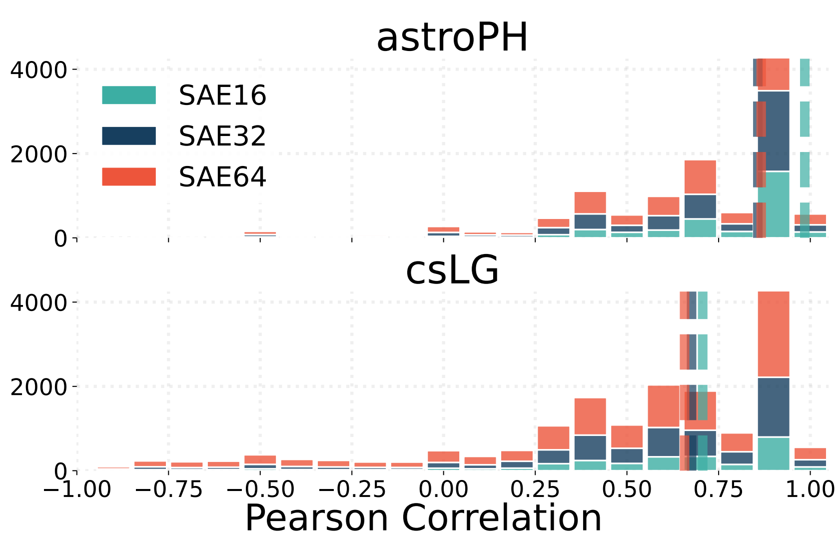



Figure 7: Pearson correlations between the ground-truth and predicted feature activation, using GPT-4o as the *Interpreter* and GPT-4o-mini as the *Predictor*.

| 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.

### B.2 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 8 and Figures 9. 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.

## C 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 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 10; this is unsurprising given how $k$ and $n_{dirs}$ correlate with feature granularity).


Figure 8: 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.

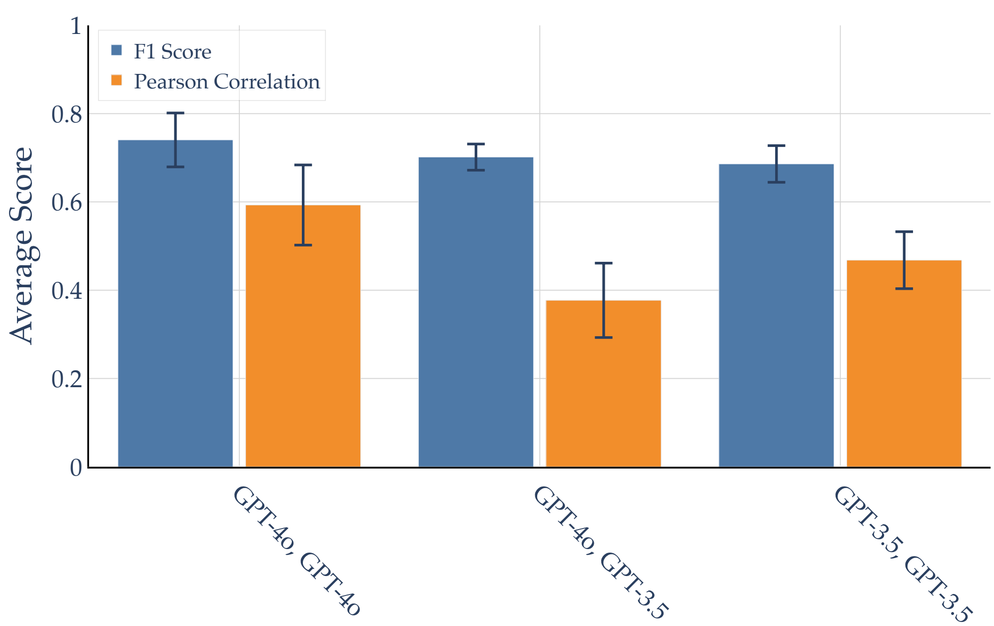



Figure 9: Mean F1 scores and Pearson correlations (according to ground-truth feature activations) across 50 randomly sampled features, for different combinations of `(Interpreter, Predictor)` models.

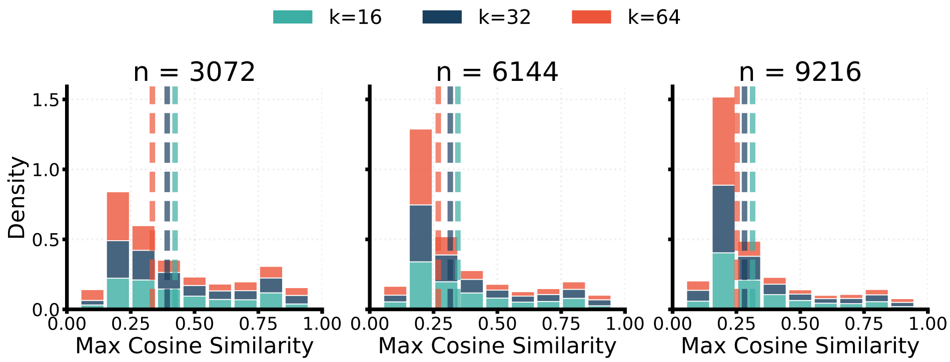



Figure 10: Maximum pair-wise cosine similarity of feature vectors between SAEs trained on different domains.

| Feature Name (`astro-ph`) | Best Match (`cs.LG`) | Cosine Sim. | Activation Sim. | $\Delta$ F1 | $\Delta$ 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`); $k = 64, n_{dir} = 9216$.

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.

# D Feature family details

## D.1 Feature splitting structures

Figure 11 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 12a and 12b we show two examples of feature splitting across SAE16 – SAE32 – SAE64 trained on `astro-ph`. 12a 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.

In contrast, 12b 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

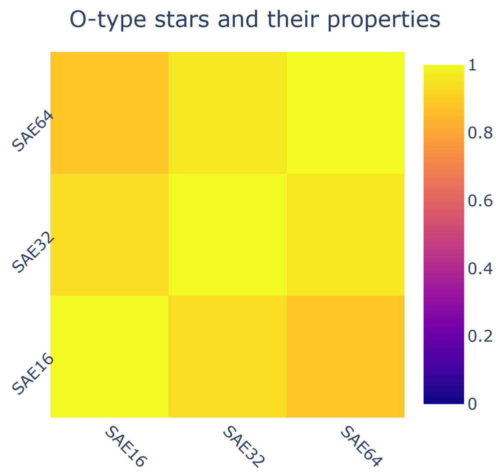



Figure 11: Recurrent features across SAEs trained on `astro-ph`; heatmap colored by activation similarity $D$; all feature vector cosine similarities are $> 0.98$.

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/LT-B/LBG as similar acronyms).

**Feature triplets** In Figure 13a, 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.

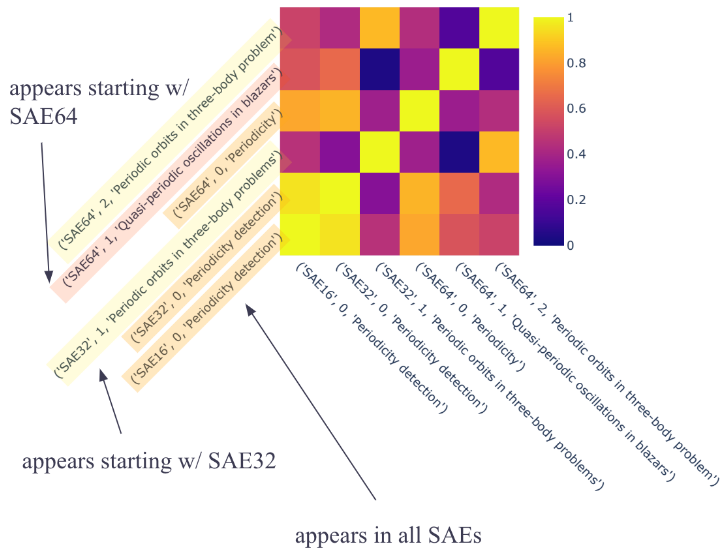
**Self-consistency** In 13b we show the set overlap between nearest-neighbour matches between SAE16 and SAE64 found directly, and nearest-neighbour matches between SAE16 and SAE64 found via nearest-neighbour matches to SAE32. If features exhibit perfectly clean splitting geometry, then these two sets 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 $\leq 3$ features in size. This corroborates findings in 6 which suggests features across models with different $k$ are not well-aligned.

### D.2 Feature family structure

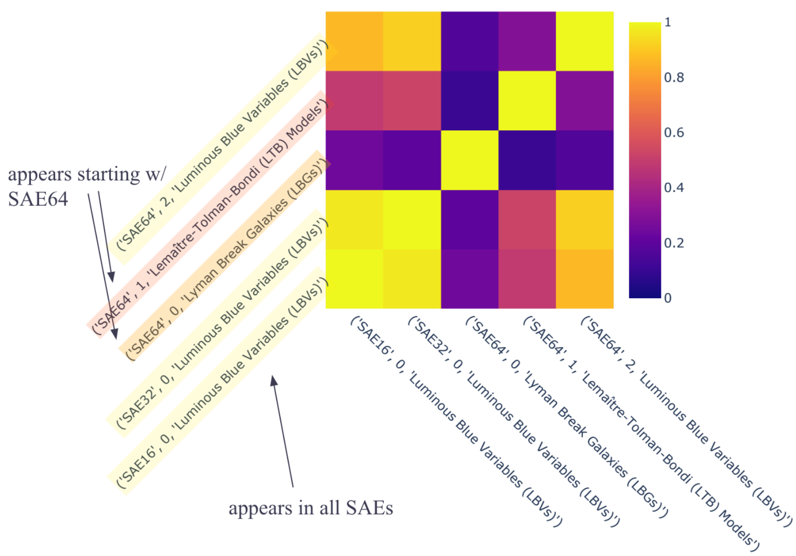
We de-duplicate families with high set overlap ($\frac{|F_1 \cap F_2|}{|F_1 \cup F_2|} > 0.6$). We compute feature family sizes (including the parent), co-occurrence ratios ($\overline{R(p, \mathcal{C})}$, see section 4), and activation similarity ratios (computed identically to $\overline{R(p, \mathcal{C})}$, just using activation similarities). Statistics for variants of `cs.LG` and `astro-ph` are shown in 14. 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 15. 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.

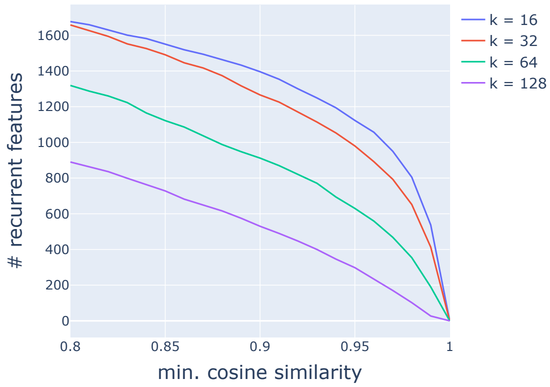
In 4 we use $n = 3$ iterations of feature family construction. We select this hyper-parameter based off Figure 16. In the first 2-3 iterations, removing parent nodes and re-constructing features preferentially creates additional smaller families, suggesting iterations are necessary to fully explore the graph.



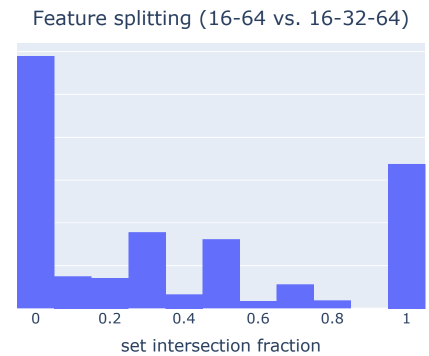

(a) We find both recurrent features and novel features at every level (i.e. the top-level "periodicity detection"/"periodicity" feature); heatmap colored by pairwise cosine similarity.



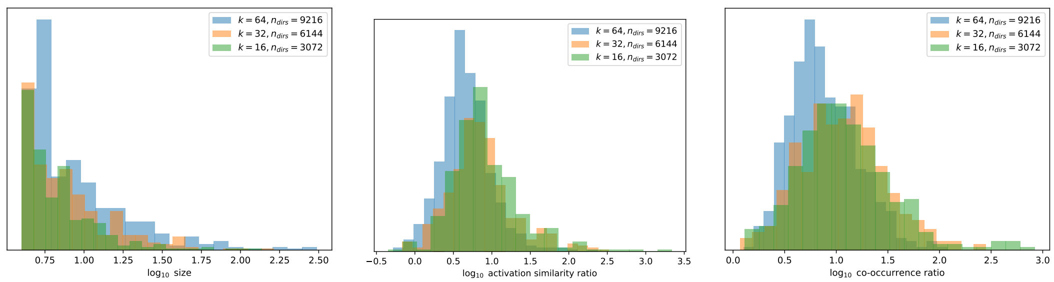

(b) While "Luminous Blue Variables" is a recurrent feature in each SAE, SAE64 also exhibits 2 other nearest-neighbour features to "Luminous Blue Variables" that are not semantically related; heatmap colored by pairwise cosine similarity.



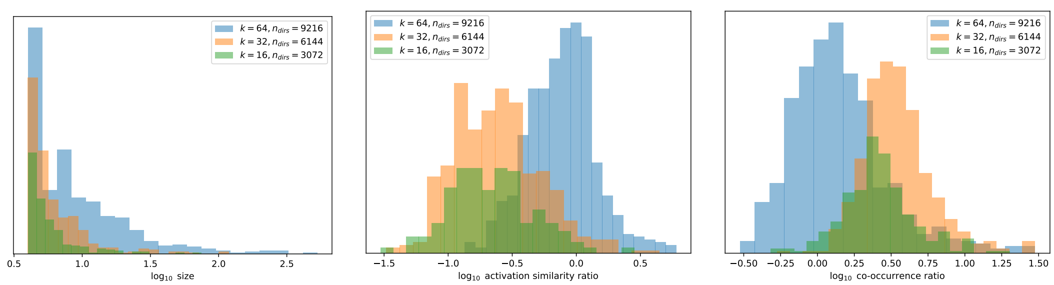

(a) Number of features from the smallest SAE that re-occur in all SAEs, by cosine similarity threshold.




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




(a) (b) (c)




(d) (e) (f)

Figure 14: Feature families statistics (left: size; middle: activation similarity ratio; right: co-occurrence ratio, $\overline{R(p, \mathcal{C})}$); $k = 64, n_{dir} = 9216$.

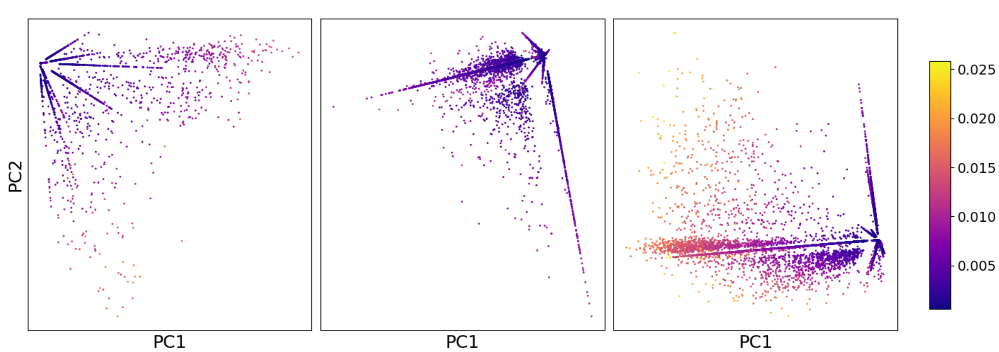



Figure 15: PCA projections of 3 example feature families from SAE64; points are latent representations of activating examples, colored by average activation for in-family features in the top $k$.

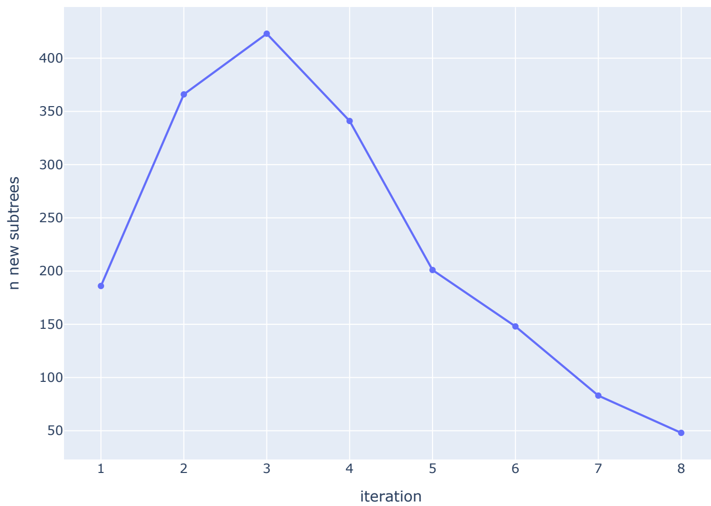



Figure 16: New feature families as a function of iteration; no deduplication is performed.

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 $n = 3$.

### D.3 Feature family interpretability

We show example feature families and their interpretability scores in Figure 17.

## E Exploring learned decoder weight matrices

**Encoder and decoder representations** Figure 18 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 18 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 $W$.

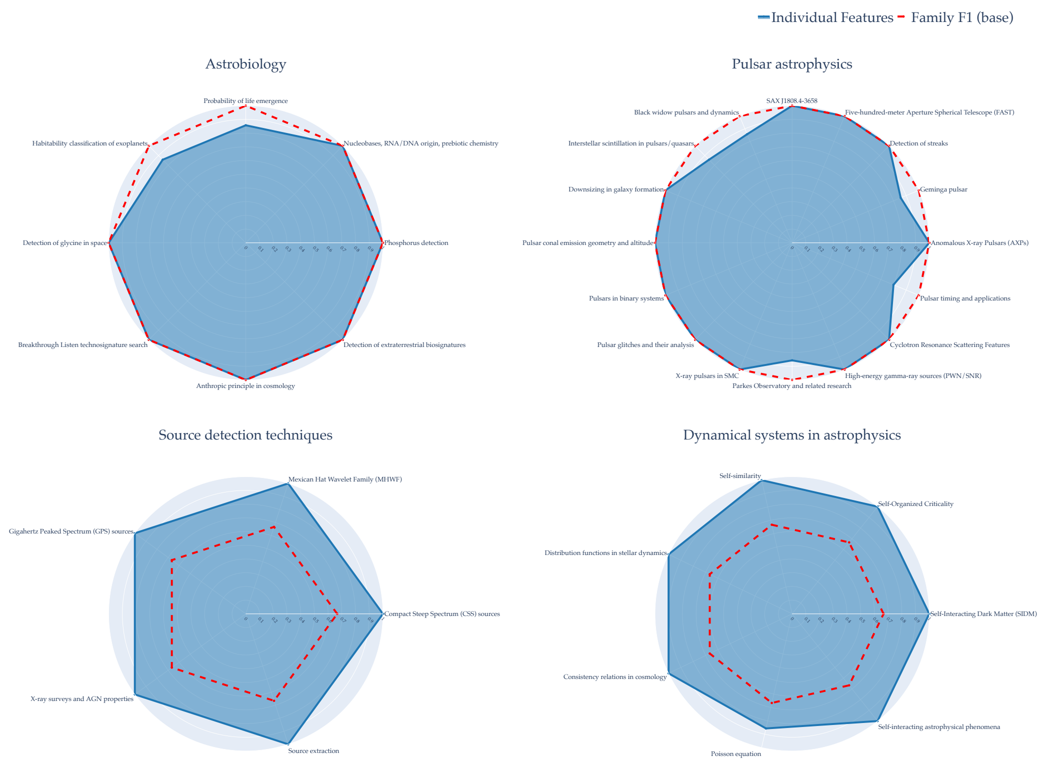



Figure 17: High-quality (top) and low-quality (bottom) feature families, scored through automated interpretability; radar charts show Pearson correlation scores for individual features (vertices) and the overall family (dashed line). While high-quality feature families truly have shared meaning, low-quality families appear to be mostly spurious and are not interpretable through short descriptions.

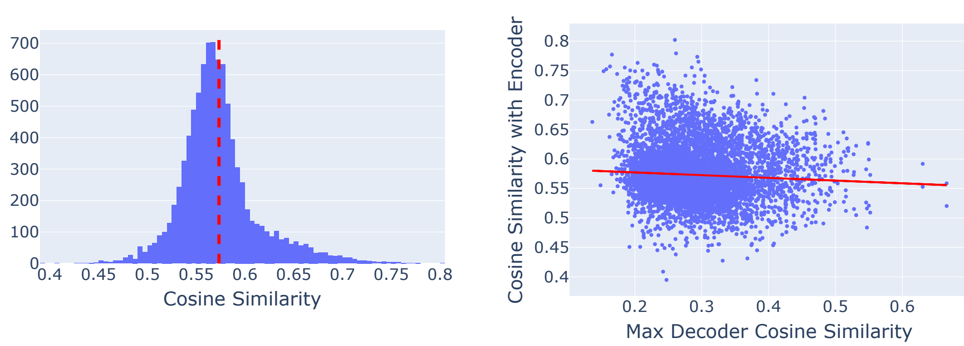



Figure 18: (Left) Cosine similarities between the encoder row and corresponding decoder column for SAE64 (`cs.LG`). The mean cosine similarity is 0.57, suggesting that encoder and decoder features are rather different, agreeing with Nanda (2023). (Right) We notice a slight negative correlation between 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.