SynthSAEBench: Evaluating Sparse Autoencoders on Scalable Realistic Synthetic Data

David Chanin [1 2 3] **Adrià Garriga-Alonso** [2]

Abstract

Improving Sparse Autoencoders (SAEs) requires benchmarks that can precisely validate architectural innovations. Current LLM-based SAE benchmarks are too noisy to differentiate architectural improvements, while commonly used synthetic-data experiments are too small-scale, unstandardized, and unrealistic to be meaningful. We introduce **SynthSAEBench**, a benchmark and toolkit for evaluating SAEs against large-scale synthetic data with realistic feature characteristics including correlation, hierarchy, and superposition, while providing ground-truth features and firings. SynthSAEBench acts as a controlled lower-bound test: SAE architectures that fail when the Linear Representation Hypothesis holds by construction have little hope on real LLMs. The benchmark reproduces known LLM SAE phenomena including the disconnect between reconstruction and latent quality, poor SAE probing, and a precision–recall trade-off mediated by L0, demonstrating that SynthSAEBench findings reproduce results on LLM SAEs. We further identify a novel failure mode: Matching Pursuit SAEs exploit superposition noise to improve reconstruction without learning ground-truth features, suggesting more expressive encoding procedures can easily overfit. SynthSAEBench complements LLM benchmarks with ground-truth features and controlled ablations for diagnosing SAE failure modes, while providing a clear target for SAE architecture work.

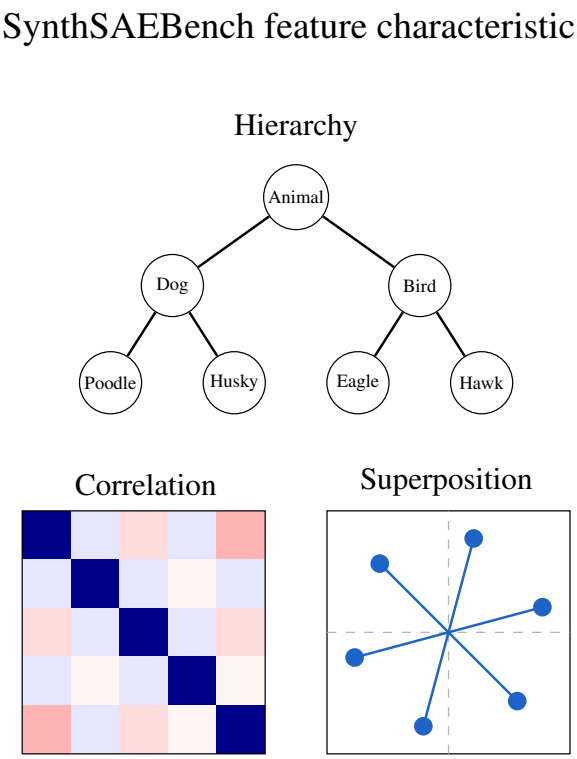

Figure 1. SynthSAEBench provides a large-scale synthetic data model with realistic feature characteristics including correlation, hierarchy, superposition and zipfian firing distributions, scalable to hundreds of thousands of features and realistic hidden dimension sizes.

1. Introduction

Large language models (LLMs) achieve remarkable performance but remain opaque, motivating interpretability research into how these models represent knowledge. The Linear Representation Hypothesis (LRH) (Park et al., 2024) posits that concepts (hereafter "features") are represented as nearly-orthogonal linear directions. Models can represent many more features than dimensions by allowing non-orthogonal directions, a phenomenon known as superposition (Elhage et al., 2022). Superposition is efficient but makes interpreting activations difficult, motivating the use of Sparse Autoencoders (SAEs) (Bricken et al., 2023; Cunningham et al., 2024) to recover underlying feature directions via sparse dictionary learning.

A key challenge in improving SAEs is that we lack ground-truth knowledge of the "true features" in an LLM. LLM

[1]University College London [2]MATS [3]Decode Research. Correspondence to: David Chanin <david.chanin.22@ucl.ac.uk>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

benchmarks such as SAEBench (Karvonen et al., 2025) evaluate SAE performance on tasks like sparse probing (Gurnee et al., 2023; Kantamneni et al., 2025), concept disentanglement (Karvonen et al., 2024), and autointerpretability (Paulo et al., 2025). However, SAEBench metrics exhibit substantial noise between runs (see Appendix J), making it difficult to evaluate small architectural improvements. Moreover, without ground-truth access, we cannot diagnose *why* SAEs score poorly, a critical obstacle given recent work showing that SAEs underperform supervised methods like logistic-regression probes (Kantamneni et al., 2025).

On the other extreme, SAE research already relies on bespoke synthetic models: toy experiments with fewer than 10 independent features (Song et al., 2025; Gribonval & Schnass, 2010; Elhage et al., 2022), or per-paper setups studying hierarchy (Chanin et al., 2025b; Costa et al., 2025; Bussmann et al., 2025) and correlation (Chanin et al., 2025a; Chanin & Garriga-Alonso, 2025). Nearly all SAE architecture work already involves experiments with synthetic models, but these models are too small, unrealistic, and unstandardized to compare architectures.

We introduce **SynthSAEBench**, an extension of the SAELens library (Bloom et al., 2024) that scales the synthetic model approach to 16k+ features with correlation, hierarchy, and superposition, providing ground-truth feature directions and firings. Data generation runs at hundreds of thousands of samples per second on a single GPU. We release a standard pretrained configuration[1] as the canonical benchmark, while allowing practitioners to change any part of the synthetic model to perform ablations and test SAE performance under different feature assumptions.

SynthSAEBench is a controlled lower-bound test for SAE architectures: If SAEs fail when the Linear Representation Hypothesis holds by construction – the setting SAEs are explicitly designed around – then we cannot expect them to succeed on real LLMs. Indeed, most previously observed LLM SAE phenomena also emerge in this regime: (1) Matryoshka SAEs overperform on SAEBench despite poor reconstruction (Bussmann et al., 2025; Karvonen et al., 2025), (2) Matching Pursuit SAEs overperform on reconstruction while scoring poorly on SAEBench (Chanin, 2025), (3) poor SAE probing performance (Kantamneni et al., 2025), (4) a precision-recall trade-off in probing mediated by SAE L0 (Chanin et al., 2025b), and (5) BatchTopK and JumpReLU both being nearly equivalent state-of-the-art architectures (Bloom et al., 2024; Bussmann et al., 2024; Rajamanoharan et al., 2024). This shows that performance on SynthSAEBench transfers to results on real LLM SAEs.

Additionally, using SynthSAEBench, we find that Matching Pursuit SAEs exploit superposition noise to improve reconstruction without learning ground-truth features, hinting at why the simple linear encoder of traditional SAEs is so hard to outperform despite known theoretical limitations (O'Neill et al., 2025).

No SAE architecture we evaluate achieves perfect performance on SynthSAEBench, highlighting a clear target for improvement in SAE architectures.

2. Background

Sparse autoencoders (SAEs). An SAE decomposes an input activation $a \in \mathbb{R}^D$ into a hidden state f consisting of L hidden neurons, called "latents". An SAE is composed of an encoder $W_{\text{enc}} \in \mathbb{R}^{L \times D}$, a decoder $W_{\text{dec}} \in \mathbb{R}^{D \times L}$, a decoder bias $b_{\text{dec}} \in \mathbb{R}^D$, and an encoder bias $b_{\text{enc}} \in \mathbb{R}^L$, and a nonlinearity σ, typically ReLU or a variant like JumpReLU (Rajamanoharan et al., 2024), TopK (Gao et al., 2024) or BatchTopK (BTK) (Bussmann et al., 2024).

$$f = \sigma(W_{\text{enc}}(a - b_{\text{dec}}) + b_{\text{enc}}) \tag{1}$$

$$\hat{a} = W_{\text{dec}} f + b_{\text{dec}} \tag{2}$$

The SAE is trained with a reconstruction loss, typically Mean Squared Error (MSE), and a sparsity-inducing loss consisting of a function $\mathcal{S}$ that penalizes non-sparse representation with corresponding sparsity coefficient λ. For standard L1 SAEs, $\mathcal{S}$ is the L1 norm of f. For TopK and BatchTopK SAEs, there is no sparsity-inducing loss ($\mathcal{S} = 0$) as the TopK function directly induces sparsity. There is sometimes also an additional auxiliary loss $\mathcal{L}_{aux}$ with coefficient α to ensure all latents fire. Standard L1 SAEs typically do not have an auxiliary loss (Olah et al., 2024). The general SAE loss is

$$\mathcal{L} = \|a - \hat{a}\|_2^2 + \lambda \mathcal{S} + \alpha \mathcal{L}_{\text{aux}}. \tag{3}$$

Matryoshka SAEs. A matryoshka SAE (Bussmann et al., 2025) extends the SAE definition by summing losses created by prefixes of SAE latents. This forces each sub-SAE to reconstruct input activations on its own, and incentivizes the SAE to place more common, general concepts into latents with smaller index number. A matryoshka SAE uses nested prefixes with sizes $\mathcal{M} = m_1, m_2, ...m_n$ where $m_1 < m_2 < \ldots < m_n = L$, where L is the number of latents in the full dictionary. Matryoshka SAE loss is:

$$\mathcal{L} = \sum_{m \in \mathcal{M}} \left(\|a - \hat{a}_m\|_2^2 + \lambda \mathcal{S}_m \right) + \alpha \mathcal{L}_{\text{aux}} \tag{4}$$

Where $\hat{a}_m$ is the reconstruction for the SAE using the first m latents, and $\mathcal{S}_m$ is the sparsity penalty applied to the first

[1] https://huggingface.co/decoderesearch/synth-sae-bench-16k-v1

m latents. For TopK and BatchTopK Matryoshka SAEs, there is no sparsity penalty ($\mathcal{S}_m = 0$) as the TopK function directly imposes sparsity.

Matching Pursuit SAEs. A Matching Pursuit (MP) SAE (Costa et al., 2025) acts like a TopK SAE where the k latents are selected in serial rather than in parallel. In an MP-SAE, there is no explicit encoder W_{enc} or encoder bias b_{enc}. Instead, at each iteration $t \in \{0, \ldots, k-1\}$, the latent with the highest projection onto the residual r_t is selected and its contribution is projected out:

$$l_t = \text{argmax}_i \, W_{\text{dec},i}^\top r_t \tag{5}$$

$$r_{t+1} = r_t - \left(W_{\text{dec},l_t}^\top r_t\right) W_{\text{dec},l_t} \tag{6}$$

with $r_0 = a$. The SAE latent vector f accumulates the projection coefficients across iterations into the corresponding latent indices:

$$f_l = \sum_{t\,:\,l_t = l} W_{\text{dec},l_t}^\top r_t, \tag{7}$$

so a latent selected on multiple iterations has its activations summed. The MP-SAE is trained with the standard MSE reconstruction loss $\|a - \hat{a}\|_2^2$, where the MP-SAE reconstruction is $\hat{a} = W_{\text{dec}} f$ (no decoder bias); by construction, this equals $\|r_k\|_2^2$.

The variant of MP-SAEs we use in this paper does not do any early stopping based on $\|r\|_2$ or based on selecting the same latent multiple times, as early stopping adds more complication to the training process and has not been shown to improve results (Chanin, 2025).

3. Synthetic Data Model

We now describe our synthetic data model in detail. We extend the traditional bernoulli-gaussian model commonly used in dictionary learning literature (Gribonval & Schnass, 2010; Wang et al., 2020) and follows the LRH by construction. This is also the standard synthetic setting used by SAE research (Song et al., 2025; O'Neill et al., 2025; Bussmann et al., 2025).

Our model contains a feature dictionary $\mathbf{D} \in \mathbb{R}^{N \times D}$ containing N ground-truth feature directions, each represented as a unit-norm vector $\mathbf{d}_i \in \mathbb{R}^D$, an optional bias $\mathbf{b} \in \mathbb{R}^D$, and an activation generator that samples sparse feature coefficients. Each feature d_i has a corresponding firing probability p_i.

Given a sample, the activation generator first determines which features fire using a Gaussian copula approach to generate correlated binary firing indicators. First, we sample from a multivariate Gaussian with correlation structure $\mathbf{\Sigma}$:

$$\mathbf{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}). \tag{8}$$

We then threshold these samples to obtain binary firing indicators that respect both the marginal firing probabilities p_i and the correlation structure. For feature i with firing probability p_i, we compute the threshold as $\tau_i = \Phi^{-1}(1 - p_i)$, where Φ^{-1} is the inverse standard normal CDF, and set $z_i = \mathbf{1}[g_i > \tau_i]$. When $\mathbf{\Sigma} = \mathbf{I}$ (no correlations between features), this is equivalent to $z_i \sim \text{Bernoulli}(p_i)$.

For features that fire ($z_i = 1$), coefficients are sampled from a rectified Gaussian distribution:

$$c_i = z_i \cdot \text{ReLU}(\mu_i + \sigma_i \epsilon_i), \quad \epsilon_i \sim \mathcal{N}(0, 1), \tag{9}$$

where μ_i and σ_i are the per-feature mean and standard deviation of firing magnitudes. Optionally, a post-processing function $h : \mathbb{R}^N \to \mathbb{R}^N$ can be applied to modify the coefficient vector, i.e., $\mathbf{c} \leftarrow h(\mathbf{c})$; we use this mechanism to implement hierarchical constraints (§3.3).

The hidden activation is then computed as:

$$\mathbf{a} = \sum_{i=1}^{N} c_i \mathbf{d}_i + \mathbf{b} = \mathbf{D}^\top \mathbf{c} + \mathbf{b}. \tag{10}$$

For scalability with large numbers of features, we use a low-rank correlation matrix:

$$\mathbf{\Sigma} = \mathbf{F}\mathbf{F}^\top + \text{diag}(\boldsymbol{\delta}), \tag{11}$$

where $\mathbf{F} \in \mathbb{R}^{N \times r}$ is a factor matrix of rank $r \ll N$, and $\boldsymbol{\delta} \in \mathbb{R}^N$ contains diagonal residual variances chosen to ensure unit diagonal in $\mathbf{\Sigma}$ (i.e., $\delta_i = 1 - \sum_j F_{ij}^2$). Sampling from this structure is efficient:

$$\mathbf{g} = \mathbf{F}\boldsymbol{\epsilon} + \sqrt{\boldsymbol{\delta}} \odot \boldsymbol{\eta}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_r), \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N), \tag{12}$$

where $\odot$ denotes elementwise multiplication. This requires only $O(Nr)$ computation per sample rather than $O(N^2)$ for full covariance sampling.

This generative model enables us to control several important phenomena that arise in real neural networks: superposition (§3.1), feature correlation (§3.2), and feature hierarchy (§3.3). Both per-feature firing probabilities p_i and magnitude parameters (μ_i, σ_i) admit configurable distributions; we describe the supported options in Appendix E. Figure 2 shows the data generating process.

3.1. Superposition

The Linear Representation Hypothesis (Park et al., 2024) posits that neural networks represent more concepts than

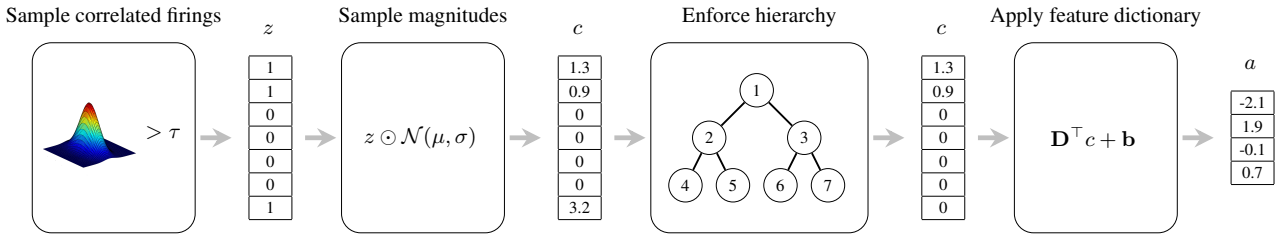

Figure 2. Overview of process to generate a single training activation, a.

they have dimensions, forcing features to share representational capacity—a phenomenon known as superposition (Elhage et al., 2022). We characterize the degree of superposition by *mean max absolute cosine similarity*, ρ_{mm}. $\rho_{\text{mm}} = \frac{1}{N}\sum_{i=1}^{N} \max_{j \neq i} |\mathbf{d}_i^\top \mathbf{d}_j|$. $0 \leq \rho_{\text{mm}} \leq 1$, with $\rho_{\text{mm}} = 0$ indicating no superposition (i.e., all features are mutually orthogonal).

Feature vectors are initialized as random unit vectors sampled from a standard normal distribution:

$$\mathbf{d}_i \leftarrow \frac{\mathbf{g}_i}{\|\mathbf{g}_i\|_2}, \quad \mathbf{g}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D). \tag{13}$$

While this initialization produces features with small expected pairwise cosine similarities (scaling as $O(1/\sqrt{D})$), some feature pairs may have higher overlap by chance. To reduce spurious correlations between feature directions, we optionally apply an orthogonalization procedure to minimize pairwise cosine similarity. Specifically, we optimize:

$$\mathcal{L}_{\text{ortho}} = \sum_{i \neq j} (\mathbf{d}_i^\top \mathbf{d}_j)^2 + \lambda \sum_i (\|\mathbf{d}_i\|_2 - 1)^2 \tag{14}$$

using gradient descent. After orthogonalization, all vectors are renormalized to unit length. This procedure encourages feature vectors to be as orthogonal as possible given the dimensionality constraints, although when using models with a large hidden dimension, we find that random initialization already results in features that are nearly orthogonal.

For scalability with large N, we use a memory-efficient chunked implementation that computes pairwise dot products in blocks. This reduces memory complexity from $O(N^2)$ to $O(\text{chunk_size} \times N)$, enabling orthogonalization of dictionaries with thousands or even millions of features.

3.2. Feature Correlation

Real neural network features are rarely independent: concepts that co-occur in data tend to co-fire in the network's representations. Previous work has shown that correlated features present challenges for SAEs, leading to phenomena like feature hedging (Chanin et al., 2025a; Chanin & Garriga-Alonso, 2025). To study these effects systematically, we implement configurable correlation structures between feature firings.

We support randomly generating a low-rank correlation matrix for use in the synthetic model. This random generation is controlled by two parameters: the rank r (which determines the complexity of the correlation patterns) and a correlation scale s (which controls the magnitude of off-diagonal correlations by scaling the factor matrix). The detailed generation procedure is described in Appendix D.

3.3. Feature Hierarchy

Concepts in natural language and vision exhibit hierarchical structure: "golden retriever" is a type of "dog," which is a type of "animal." Previous work has shown that SAEs struggle with hierarchical features, leading to phenomena like feature absorption (Chanin et al., 2025b). We implement configurable hierarchical dependencies between features to study these effects.

Our hierarchy is represented as a forest of trees, where each node corresponds to a feature and children can only fire when their parent is active. Formally, after sampling the initial firing indicators z_i, we apply the constraint:

$$c_{\text{child}} \leftarrow c_{\text{child}} \cdot \mathbf{1}[c_{\text{parent}} > 0], \tag{15}$$

which zeros out child activations whenever the parent is inactive.

Additionally, we support *mutual exclusion* among siblings: when a parent node is marked as having mutually exclusive children, at most one child can be active per sample. This models concepts like "dog" vs. "bird"—both are animals, but a single entity cannot be both simultaneously. When multiple siblings would fire, one is randomly selected as the winner and the others are deactivated.

We also support *parent-scaled magnitudes*, where child activation magnitudes are modulated by their parent's activation strength rather than simply being binary-gated. This models the intuition that the intensity of a specific concept (e.g., "golden retriever") should scale with the intensity of its parent concept (e.g., "dog"). Details are in Appendix C.

For efficiency, we precompute the hierarchy structure as sparse index tensors, enabling $O(\text{active features})$ processing rather than $O(N)$ per sample.

4. Evaluating SAEs on Synthetic Data

A key advantage of synthetic data is access to ground-truth feature directions and activations, enabling precise evaluation of SAE quality. We implement a comprehensive set of metrics organized into four categories: reconstruction quality, feature recovery, classification accuracy, and sparsity.

4.1. Reconstruction Metrics

Explained Variance (R^2). We measure how well the SAE reconstruction $\hat{\mathbf{a}}$ captures the variance in the input activations $\mathbf{a}$:

$$R^2 = 1 - \frac{\mathbb{E}[\|\mathbf{a} - \hat{\mathbf{a}}\|_2^2]}{\mathrm{Var}(\mathbf{a})}, \tag{16}$$

where $\mathrm{Var}(\mathbf{a}) = \mathbb{E}[\|\mathbf{a}\|_2^2] - \|\mathbb{E}[\mathbf{a}]\|_2^2$ is the total variance. A value of 1.0 is perfect reconstruction.

4.2. Feature Recovery Metrics

Mean Correlation Coefficient (MCC). Following Song et al. (2025), we measure how well SAE decoder columns $\mathbf{w}_j$ align with ground-truth feature directions $\mathbf{d}_i$ using optimal bipartite matching. We compute the absolute cosine similarity matrix $|S_{ij}| = |\mathbf{w}_j^\top \mathbf{d}_i|$ (both $\mathbf{w}_j$ and $\mathbf{d}_i$ are unit-norm), find the optimal one-to-one matching via the Hungarian algorithm, and report the mean similarity of matched pairs:

$$\mathrm{MCC} = \frac{1}{\min(L, N)} \sum_{(i,j) \in \text{matching}} |\mathbf{w}_j^\top \mathbf{d}_i|. \tag{17}$$

Feature Uniqueness. We measure what fraction of SAE latents track unique ground-truth features. For each latent j, we find its best-matching ground-truth feature: $i^*(j) = \arg\max_i |\mathbf{w}_j^\top \mathbf{d}_i|$. Uniqueness is the fraction of unique best matches divided by the number of latents:

$$\mathrm{Uniqueness} = \frac{|\{i^*(j) : j = 1, \ldots, L\}|}{L}. \tag{18}$$

A value of 1.0 means each latent tracks a different feature.

4.3. Classification Metrics

We evaluate each SAE latent as a binary classifier for its best-matching ground-truth feature. For latent j matched to feature $i^*(j)$, we compute standard classification metrics over evaluation samples: precision, recall, and F1 score. A latent is considered to fire when $f_j > 0$, and a ground-truth feature is considered to fire when $z_i = 1$. We report the mean of each metric across all latents.

4.4. Sparsity Metrics

L0 Comparison. We track the average L0 (number of active features per sample) of both ground-truth activations and for SAE latent activations.

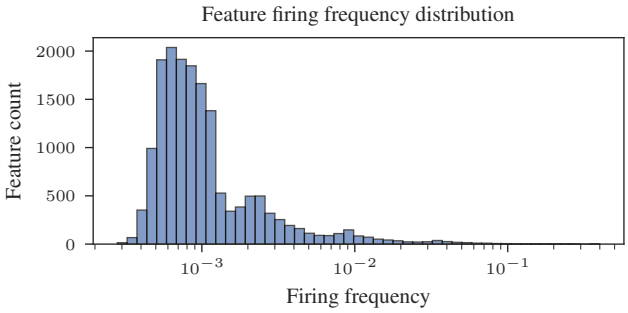

Figure 3. SynthSAEBench-16k feature firing probabilities.

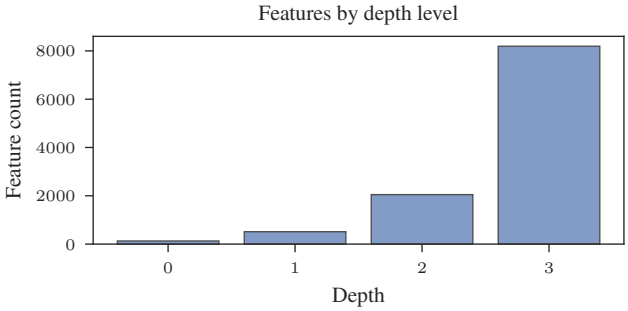

Figure 4. SynthSAEBench-16k hierarchy distribution.

Dead Latents. We count the number of SAE latents that never fire across the evaluation set. Dead latents represent wasted capacity and indicate training issues.

5. Benchmark Configuration

We define the standard SynthSAEBench configuration, chosen to be large-scale enough to elicit realistic SAE training dynamics while remaining tractable to run.

The model has $N = 16{,}384$ features in hidden dimension $D = 768$, yielding $\rho_{\text{mm}} \approx 0.15$. We estimate this is similar to the level of superposition 1B features would cause in a Gemma-2-2b sized model (Appendix K). Activations have mean L2 norm 28 and stdev 5, matching Pythia-160m (Biderman et al., 2023) layer 10. Feature vectors are random unit vectors orthogonalized for 100 steps (lr 3×10^{-4}); the bias is a random Gaussian direction normalized to magnitude 0.5. Base firing probabilities follow a Zipfian distribution ($\alpha = 0.5$, $p_{\max} = 0.4$, $p_{\min} = 5 \times 10^{-4}$); firing magnitudes have means interpolating linearly from 5.0 to 4.0, with stdevs $\sigma_i \sim |\mathcal{N}(0.5, 0.5^2)|$. The firing probability distribution is shown in Figure 3.

The hierarchy has 128 root trees with branching factor 4 and max depth 3, covering 10,880 features. All sibling nodes are mutually exclusive, child magnitudes scale with parent activation (Appendix C), and base probabilities are

compensated for hierarchy and mutual exclusion effects (Appendix B). The distribution of nodes in the hierarchy is shown in Figure 4. Beyond the hierarchy, a random low-rank correlation with rank $r = 25$ and scale $s = 0.1$ adds structured dependencies. The model has an L0 of 34, determined by the firing-probability distribution.

5.1. Benchmark instructions

We recommend training SAEs of width 4096 on SynthSAEBench, intentionally more narrow than the model dictionary, since LLM SAEs are always narrower than the (unknown) number of true features in an LLM (Templeton et al., 2024). Unless otherwise specified, we train SAEs on 200M samples (15–20 minutes per SAE on a single H100, batch size 1024). Sampling performance is in Appendix I.

6. Results

We train standard L1, Matching Pursuit, BatchTopK, Matryoshka BatchTopK, and JumpReLU SAEs on SynthSAEBench.[2] These SAEs have width 4096 and are trained on 200M samples, as recommended in Section 5.1. For standard L1 SAEs and JumpReLU SAEs, we implement a controller to automatically tune their sparsity coefficient during training to hit a target L0. This controller is described in more detail in Appendix G.

We vary the L0 of the SAEs from 15 to 45 with 5 seeds per L0. F1-score, MCC, and variance explained are shown in Figure 5. Matryoshka SAEs achieve the best probing and MCC scores, indicating the best latent quality, despite poor explained variance (poor reconstruction). This matches the results of Matryoshka SAEs on SAEBench (Karvonen et al., 2025), where this same pattern is observed for LLM SAEs. The performance drop-off for Matryoshka SAEs at low L0 seems to be due to dead latents rather than a fundamental architectural issue (see Appendix H). We introduce a per-prefix Matryoshka auxiliary loss that reduces, but does not eliminate, these dead latents (Appendix F.4). MP-SAEs, on the other hand, have the best variance explained, but have poor probing and MCC results, indicating that MP-SAEs' impressive reconstruction comes at the cost of latent quality. We see JumpReLU and BatchTopK achieving roughly similar levels of quality and reconstruction, further validating why these are both considered state-of-the-art architectures.

6.1. Precision-recall trade-off mediated by L0

One striking result from Figure 5 is that no SAE achieves near-perfect probing F1 at any L0, with the best performer, Matryoshka SAEs, peaking at around F1=0.88. This means no SAEs will act as a great classifier for ground-truth model features. This is consistent with results showing that LLM SAEs underperform logistic-regression probes (Kantamneni et al., 2025). Indeed, logistic regression probes trained directly on SynthSAEBench activations achieve a mean F1 of 0.974 (Table 1; training details in Appendix L), substantially outperforming the best SAE.

Table 1. Logistic regression probe results on SynthSAEBench (first 4,096 highest-frequency features). Supervised probes substantially outperform the best SAE probing F1 of ≈0.88.

Metric	Mean	Median
AUC	0.9999	1.0000
Accuracy	0.9998	0.9999
Precision	0.9800	0.9853
Recall	0.9680	0.9711
F1	0.9739	0.9775

This poor probing performance is one of the key criticisms of SAEs, so our ability to reproduce it in a setting with known ground-truth features is a key first step to address this problem with SAE architectural improvements.

We show the precision and recall for the probing task in Figure 6. Higher L0 increases recall at the cost of precision, reproducing the precision-recall trade-off mediated by SAE L0 seen in previous LLM SAE studies (Karvonen et al., 2025; Chanin et al., 2025b).

6.2. Near-equivalence between JumpReLU and BatchTopK

JumpReLU and BatchTopK are both considered state-of-the-art architectures (Bloom et al., 2024; Rajamanoharan et al., 2024; Bussmann et al., 2024), and this is borne out on SynthSAEBench as well. Among standard architectures, we see these both performing nearly identically on F1 score at L0=20 and L0=25, and very closely on both MCC and reconstruction. This further shows that SynthSAEBench results reflect what we see in LLM SAEs.

6.3. MP-SAEs overfit superposition noise

We next investigate the effect of superposition noise on SAE performance. We modify the hidden dim of the base SynthSAEBench model, ranging from 256 to 1536, scaling training samples by $(\frac{D}{768})^{(0.6)}$ to account for the change in SAE parameters as recommended by Gao et al. (2024). We train all SAEs at L0=25, as this seems to be a good setting for most SAEs on the base SynthSAEBench model. Results are shown in Figure 7.

We see the variance explained decrease with increasing superposition noise except, interestingly, for Matching Pursuit (MP) SAEs. MP-SAEs actually increase variance explained the more superposition noise is present while decreasing

[2]Experiment code is available at https://github.com/decoderesearch/synth-sae-bench-experiments.

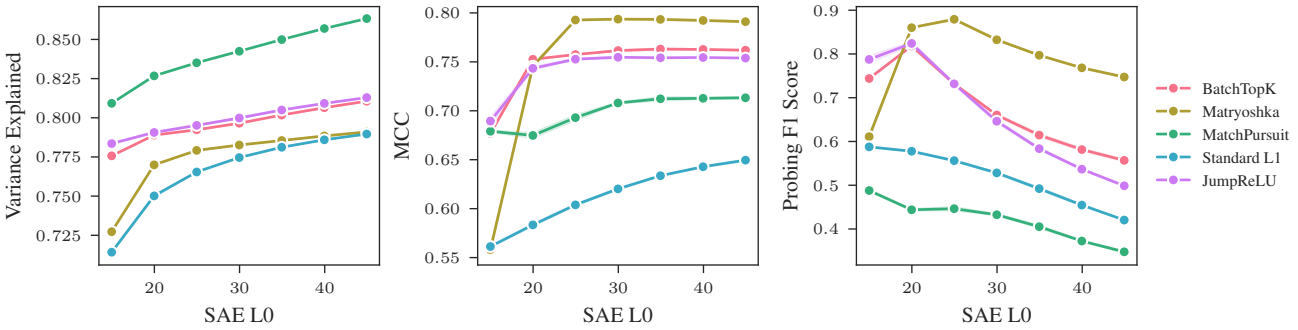

Figure 5. Variance explained (left), MCC (middle), and F1-score (right) for SynthSAEBench SAEs with varying L0. Shaded area is stdev with 5 seeds (too small to be visible for most SAEs).

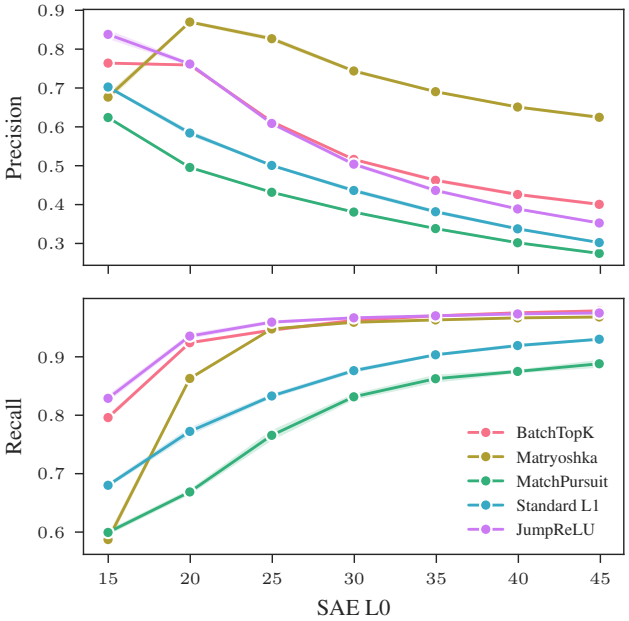

Figure 6. Probing precision and recall for SAEs trained on SynthSAEBench across varying L0 values. Higher L0 increases recall at the cost of precision. Shaded area is stdev.

MCC and F1-score, implying that MP-SAEs are able to overfit feature overlap due to superposition to improve reconstruction. MP-SAEs use a more expressive encoding process than the simple linear encoder of traditional SAEs, and this extra expressivity can result in overfitting reconstruction without learning the underlying features of the model, perhaps explaining why traditional SAEs with simple linear encoders are so hard to outperform in practice (O'Neill et al., 2025). This insight was only possible because SynthSAEBench provides both ground-truth features (to measure MCC and F1 independently of reconstruction) and controlled superposition levels (to isolate its effect), neither of which is available when benchmarking on LLMs.

6.4. Additional ablations

We further ablate feature correlation strength and rank, base firing probabilities, firing magnitude stdev, and hierarchy depth (Appendix M). The overall pattern from Figure 5 is stable across all of these ablations: Matryoshka SAEs lead on latent quality (MCC and F1) while MP-SAEs lead on reconstruction. Notably, increasing correlation strength increases variance explained while slightly decreasing probing F1, consistent with prior work showing SAEs exploit correlations by mixing correlated features into a single latent (Chanin et al., 2025a; Chanin & Garriga-Alonso, 2025). Increasing firing magnitude stdev improves MCC while reducing F1 for every SAE: high variance makes feature directions more salient, but low-magnitude firings become hard to disambiguate from superposition noise. Surprisingly, standard L1 SAEs achieve the best MCC of all architectures in this high-stdev regime.

7. Related Work

SAE evaluation. Evaluating SAEs on LLMs is challenging due to the lack of ground truth. SAEBench (Karvonen et al., 2025) provides a suite of downstream tasks including sparse probing (Gurnee et al., 2023), concept erasure (Karvonen et al., 2024), and autointerp (Paulo et al., 2025). However, these metrics have high variance and measure indirect proxies rather than feature recovery directly. The MCC metric (O'Neill et al., 2025) provides a principled way to compare learned features to ground truth. While not directly applicable for SAEs, Interpbench (Gupta et al., 2024) provides a synthetic model for circuit discovery work.

Toy models and synthetic data. Elhage et al. (2022) introduced toy models of superposition to study how neural networks represent more features than dimensions. Subsequent work has used small-scale synthetic setups to study specific phenomena: feature absorption in hierarchical features (Chanin et al., 2025b), feature hedging under correlation (Chanin et al., 2025a), and incorrect L0 behavior

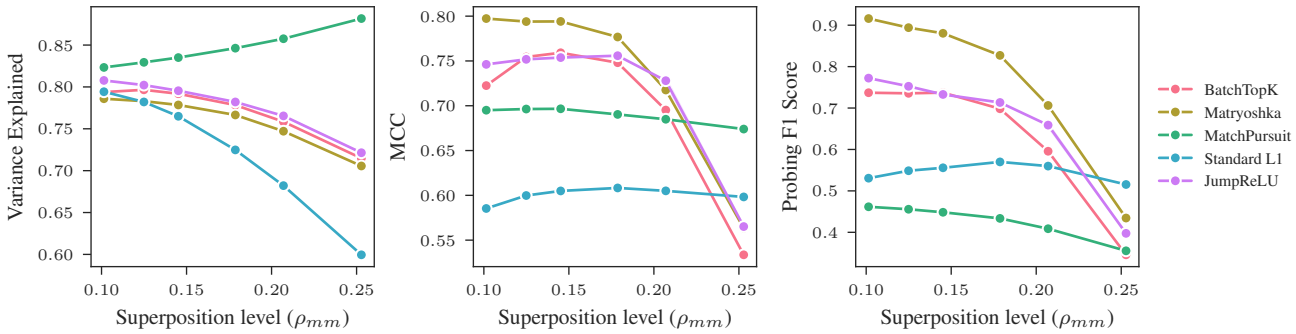

Figure 7. Variance explained (left), MCC (middle), and F1-score (right) for SAEs trained on variants of SynthSAEBench with different levels of superposition. Interestingly, MP-SAEs increase their variance explained at high superposition, implying they are able to overfit on superposition noise.

(Chanin & Garriga-Alonso, 2025). However, these studies use bespoke synthetic models that are not standardized or comparable. Our work provides a unified, large-scale synthetic framework that encompasses superposition, correlation, and hierarchy while providing ground-truth features.

Alternative representation hypotheses. Our work follows the LRH, but other feature hypotheses exist that extend the LRH and are thus natural extensions for our work. Fel et al. (2025) introduce the Minkowski Representation Hypothesis, where the softmax operation in the transformer results in polytopes in the representation space. Another extension is feature manifolds (Chen et al., 2018; Michaud et al.), where features span manifolds rather than corresponding to a single linear direction.

8. Discussion

SynthSAEBench represents a best-case scenario for SAEs: the Linear Representation Hypothesis holds by construction, with features that are truly linear directions. Yet no SAE architecture we evaluate achieves perfect feature recovery. This is significant because a common response to SAE failures on LLMs is that the representation hypothesis may not hold exactly. Our results show that even when the LRH is perfectly satisfied, current SAE architectures still struggle with superposition, correlation, and hierarchy. The bottleneck is in the SAE architectures and training dynamics themselves, not in the representation hypothesis, strengthening the case for continued architectural innovation.

The SAEs we train are deliberately narrower than the underlying model ($L = 4096 < N = 16{,}384$), reflecting the LLM regime where SAEs are always narrower than the number of true features in the model (Templeton et al., 2024). Under MSE, this capacity mismatch incentivizes learned latents to encode mixtures of ground-truth features (Chanin et al., 2025a; Chanin & Garriga-Alonso, 2025), and this feature mixing or "hedging" is precisely a failure mode current SAEs exhibit. We view imperfect recovery on SynthSAEBench not as a confound but as the target problem: the goal is architectures or training procedures that recover features cleanly even when narrower than the underlying model.

With ground-truth features and controlled ablations, we reproduce key LLM SAE phenomena: Matryoshka SAEs' high probing performance despite poor reconstruction (Bussmann et al., 2025), MP-SAEs' poor probing despite high reconstruction (Chanin, 2025), the gap between SAE probing and supervised probes (Kantamneni et al., 2025), and the precision-recall trade-off mediated by L0 (Chanin et al., 2025b). We also identify a new failure mode: MP-SAEs exploit superposition noise to improve reconstruction without learning ground-truth features, which may partly explain why traditional SAEs with simple linear encoders remain hard to outperform (O'Neill et al., 2025); more expressive encoding procedures can exploit spurious correlations from superposition.

We emphasize that SynthSAEBench is intended to complement, not replace, LLM benchmarks. Our synthetic model cannot capture all aspects of real neural network representations (see Appendix A for limitations). However, it offers capabilities that LLM benchmarks fundamentally cannot: ground-truth features, controlled ablations, low-noise metrics, and fast iteration. We envision researchers using SynthSAEBench to rapidly prototype and diagnose SAE architectures, then validating promising approaches on LLM benchmarks like SAEBench.

In future work, we hope to extend our framework to Minkowski representations (Fel et al., 2025) by introducing a "soft" variant of mutual exclusion via softmax, and to feature manifolds, though evaluating SAEs on manifold-structured data is challenging. More broadly, building synthetic models under different representation hypotheses and comparing SAE training dynamics to those observed in LLMs could help test the validity of those hypotheses.

Acknowledgements

David Chanin was supported thanks to EPSRC EP/S021566/1 and the Machine Learning Alignment and Theory Scholars (MATS) program. We are grateful to Lovkush Agarwal, Fred Bruford, and Tasos Spiliotopoulos for feedback during the project.

References

Åström, K. J. and Murray, R. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

Ayonrinde, K. Adaptive sparse allocation with mutual choice & feature choice sparse autoencoders. *arXiv preprint arXiv:2411.02124*, 2024.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Bloom, J., Tigges, C., Duong, A., and Chanin, D. Saelens. https://github.com/jbloomAus/SAELens, 2024.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

Bussmann, B., Leask, P., and Nanda, N. BatchTopK sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Bussmann, B., Nabeshima, N., Karvonen, A., and Nanda, N. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

Chanin, D. Training matching pursuit SAEs on LLMs. LessWrong, 2025. URL https://www.lesswrong.com/posts/rE43EfHigXcjTmomJ/training-matching-pursuit-saes-on-llms.

Chanin, D. and Garriga-Alonso, A. Sparse but wrong: Incorrect l0 leads to incorrect features in sparse autoencoders. *arXiv preprint arXiv:2508.16560*, 2025.

Chanin, D., Dulka, T., and Garriga-Alonso, A. Feature hedging: Correlated features break narrow sparse autoencoders. *arXiv preprint arXiv:2505.11756*, 2025a.

Chanin, D., Wilken-Smith, J., Dulka, T., Bhatnagar, H., Golechha, S., and Bloom, J. I. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=R73ybUciQF.

Chen, Y., Paiton, D., and Olshausen, B. The sparse manifold transform. *Advances in neural information processing systems*, 31, 2018.

Costa, V., Fel, T., Lubana, E. S., Tolooshams, B., and Ba, D. From flat to hierarchical: Extracting sparse representations with matching pursuit. *arXiv preprint arXiv:2506.03093*, 2025.

Cunningham, H., Smith, L. R., Ewart, A., Huben, R., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Engels, J., Michaud, E. J., Liao, I., Gurnee, W., and Tegmark, M. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=d63a4AM4hb.

Fel, T., Wang, B., Lepori, M. A., Kowal, M., Lee, A., Balestriero, R., Joseph, S., Lubana, E. S., Konkle, T., Ba, D., et al. Into the rabbit hull: From task-relevant concepts in dino to minkowski geometry. *arXiv preprint arXiv:2510.08638*, 2025.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Gribonval, R. and Schnass, K. Dictionary identification—sparse matrix-factorization via ℓ_1 -minimization. *IEEE Transactions on Information Theory*, 56(7):3523–3539, 2010. doi: 10.1109/TIT.2010.2048466.

Gupta, R., Arcuschin Moreno, I., Kwa, T., and Garriga-Alonso, A. Interpbench: Semi-synthetic transformers for evaluating mechanistic interpretability techniques. *Advances in Neural Information Processing Systems*, 37: 92922–92951, 2024.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a

haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=JYs1R9IMJr.

Kantamneni, S., Engels, J., Rajamanoharan, S., Tegmark, M., and Nanda, N. Are sparse autoencoders useful? a case study in sparse probing. *arXiv preprint arXiv:2502.16681*, 2025.

Karvonen, A., Rager, C., Marks, S., and Nanda, N. Evaluating sparse autoencoders on targeted concept erasure tasks. *arXiv preprint arXiv:2411.18895*, 2024.

Karvonen, A., Rager, C., Lin, J., Tigges, C., Bloom, J., Chanin, D., Lau, Y.-T., Farrell, E., McDougall, C., Ayonrinde, K., Wearden, M., Conmy, A., Marks, S., and Nanda, N. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability, 2025. URL https://arxiv.org/abs/2503.09532.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Michaud, E. J., Gorton, L., and McGrath, T. Understanding sparse autoencoder scaling in the presence of feature manifolds. In *Mechanistic Interpretability Workshop at NeurIPS 2025*.

Michaud, E. J., Liu, Z., Girit, U., and Tegmark, M. The quantization model of neural scaling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Olah, C., Templeton, A., Bricken, T., and Jermyn, A. April update. https://transformer-circuits.pub/2024/april-update/index.html, 2024. URL https://transformer-circuits.pub/2024/april-update/index.html.

O'Neill, C., Gumran, A., and Klindt, D. Compute optimal inference and provable amortisation gap in sparse autoencoders. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=8forr1FkvC.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=UGpGkLzwpP.

Paulo, G. S., Mallen, A. T., Juang, C., and Belrose, N. Automatically interpreting millions of features in large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=EemtbhJOXc.

Piantadosi, S. T. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, 21(5):1112–1130, 2014. doi: 10.3758/s13423-014-0585-6.

Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J., and Nanda, N. Jumping ahead: Improving reconstruction fidelity with JumpReLU sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Rugh, W. J. and Shamma, J. S. Research on gain scheduling. *Automatica*, 36(10):1401–1425, 2000.

Song, X., Muhamed, A., Zheng, Y., Kong, L., Tang, Z., Diab, M. T., Smith, V., and Zhang, K. Position: Mechanistic interpretability should prioritize feature consistency in saes. *arXiv preprint arXiv:2505.20254*, 2025.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Tamkin, A., Durmus, E., Hume, T., Mosconi, F., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C., and Henighan, T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. https://transformer-circuits.pub/2024/scaling-monosemanticity/, May 2024. Accessed on May 21, 2024.

Wang, Y., Wu, S., and Yu, B. Unique sharp local minimum in l1-minimization complete dictionary learning. *Journal of Machine Learning Research*, 21(63):1–52, 2020.

A. Limitations

Synthetic data cannot capture all aspects of real neural network representations. Our generative model assumes linear feature directions, which may not hold for all concepts (Engels et al., 2025). The correlation and hierarchy structures, while configurable, are simplified approximations of the complex dependencies in real data. Most importantly, there may be "unknown unknowns", phenomena in real networks that we have not thought to model. Due to our lack of true ground-truth knowledge of features in LLMs, we do not know how to set hyperparameters like number of features, correlation levels, hierarchy degree, and superposition level. We also do not attempt to model complex feature geometry aside from superposition noise. Synthetic benchmarks should complement, not replace, evaluation on real models.

B. Compensating base probabilities for hierarchy

When hierarchy constraints are applied, the effective firing probability of child features is reduced because children

can only fire when their parent is active. Similarly, mutual exclusion further reduces effective probabilities since only one sibling can remain active. We implement optional probability compensation to correct for these reductions, ensuring that the *effective* firing rate of each feature matches its specified base probability.

Hierarchy correction. Consider a feature i with base firing probability p_i whose parent has base probability p_{parent}. Without compensation, the effective probability of feature i firing is approximately $p_i \cdot p_{\text{parent}}$, since the child can only fire when the parent fires. To compensate, we scale up the sampling probability by a correction factor:

$$\gamma_i^{\text{hier}} = \frac{1}{p_{\text{parent}}}. \tag{19}$$

After sampling with corrected probability $\tilde{p}_i = \min(1, p_i \cdot \gamma_i^{\text{hier}})$ and applying hierarchy constraints, the effective firing rate is approximately p_i. The clamp at 1 is active only when $p_i > p_{\text{parent}}$; in our setup this never occurs, because features are assigned to tree nodes in breadth-first order while base probabilities follow a non-increasing Zipfian distribution, so a child always has $p_i \leq p_{\text{parent}}$.

For deeper hierarchies, this correction is applied recursively. A feature at depth d with ancestors having probabilities $p_1, p_2, \ldots, p_{d-1}$ would naively have effective probability $p_i \cdot \prod_{k=1}^{d-1} p_k$. However, since each ancestor also receives its own correction, the child only needs to correct for its immediate parent's base probability.

Mutual exclusion correction. When a parent node has mutually exclusive children, at most one child can remain active per sample. This introduces additional probability reduction beyond hierarchy constraints.

Consider an ME group with children having base probabilities $p_1, p_2, \ldots, p_m$ under a parent with probability p_P. After hierarchy correction, conditioned on the parent firing, child j fires with probability approximately p_j/p_P. When multiple children fire, one is randomly selected as the winner.

For child i, the expected number of competing siblings (given parent fires) is:

$$\mathbb{E}[\text{competitors}] = \sum_{j \neq i} \frac{p_j}{p_P}. \tag{20}$$

To first order, the probability of child i being deactivated by ME is proportional to this expected competitor count. We apply a multiplicative ME correction:

$$\gamma_i^{\text{ME}} = 1 + \sum_{j \neq i} \frac{p_j}{p_P}. \tag{21}$$

The total correction factor for a feature in an ME group is $\gamma_i = \gamma_i^{\text{hier}} \cdot \gamma_i^{\text{ME}}$.

Limitations. This compensation is approximate and assumes independence between sibling firings. In practice, correlations between features (introduced via the correlation matrix $\mathbf{\Sigma}$) can cause deviations from the target probabilities. The correction also does not account for higher-order effects when multiple levels of hierarchy and ME interact. Nevertheless, we find empirically that compensation substantially improves the match between specified and effective firing probabilities.

C. Parent-scaled child magnitudes

By default, hierarchy constraints apply binary gating: a child feature retains its sampled magnitude when its parent is active, and is zeroed out otherwise. We additionally support *parent-scaled magnitudes*, where the child's activation is modulated by the parent's normalized activation strength. Formally, for a child feature with coefficient c_{child} whose parent has coefficient c_{parent} and mean firing magnitude $\bar{\mu}_{\text{parent}}$:

$$c_{\text{child}} \leftarrow c_{\text{child}} \cdot \frac{c_{\text{parent}}}{\bar{\mu}_{\text{parent}}}, \tag{22}$$

where $\bar{\mu}_{\text{parent}}$ is the precomputed mean magnitude of the parent feature. When the parent is inactive ($c_{\text{parent}} = 0$), the child is zeroed out as in the standard case. Dividing by $\bar{\mu}_{\text{parent}}$ normalizes the scaling so that the child's expected magnitude is preserved on average.

This models the intuition that more specific concepts should be modulated by the intensity of their parent concept. For instance, features associated with specific dog breeds should fire more strongly when the "dog" feature is strongly active. Without this scaling, the child magnitude is independent of how strongly the parent fires, which may be unrealistic for many natural concepts.

Parent-scaled magnitudes can be enabled independently per parent node, and compose with both mutual exclusion and the probability compensation described in Appendix B. The scaling is applied after binary gating and mutual exclusion resolution, so the cascading order remains: (1) parent gating, (2) mutual exclusion, (3) magnitude scaling.

D. Low-rank correlation matrix generation

For scalability with large numbers of features, we use a low-rank representation of the correlation matrix rather than storing the full $N \times N$ matrix. This reduces storage from $O(N^2)$ to $O(Nr)$ where r is the rank.

The correlation structure is represented as:

$$\mathbf{\Sigma} = \mathbf{F}\mathbf{F}^\top + \text{diag}(\boldsymbol{\delta}) \tag{23}$$

where $\mathbf{F} \in \mathbb{R}^{N \times r}$ is a factor matrix and $\boldsymbol{\delta} \in \mathbb{R}^N$ contains diagonal residual variances.

Generation procedure. Given rank r and correlation scale s, we generate the factor matrix by sampling from a scaled normal distribution:

$$F_{ij} \sim s \cdot \mathcal{N}(0, 1) \tag{24}$$

The diagonal term is then computed to ensure unit diagonal in the implied correlation matrix:

$$\delta_i = 1 - \sum_{j=1}^{r} F_{ij}^2 \tag{25}$$

Numerical stability. If any diagonal term δ_i falls below a minimum threshold (we use 0.01), the entire factor matrix is scaled down to ensure all diagonal terms remain valid. Specifically, we compute a scale factor:

$$\text{scale} = \sqrt{\frac{1 - \delta_{\min}}{\max_i \sum_j F_{ij}^2}} \tag{26}$$

and apply $\mathbf{F} \leftarrow \text{scale} \cdot \mathbf{F}$, then recompute $\boldsymbol{\delta}$.

Efficient sampling. Sampling from this low-rank structure requires only $O(Nr)$ computation per batch:

$$\mathbf{g} = \mathbf{F}\boldsymbol{\epsilon} + \sqrt{\boldsymbol{\delta}} \odot \boldsymbol{\eta}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_r), \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_N) \tag{27}$$

where $\odot$ denotes elementwise multiplication. The resulting $\mathbf{g}$ has the desired correlation structure $\mathbb{E}[\mathbf{g}\mathbf{g}^\top] = \boldsymbol{\Sigma}$.

E. Configurable firing distributions

Both per-feature firing probabilities p_i and per-feature magnitude parameters (μ_i, σ_i) admit several configurable distributions in our framework.

Firing probabilities. The distribution of p_i across features significantly impacts SAE training dynamics: very rare features are difficult to learn, while very frequent ones dominate reconstruction loss. We use a Zipfian distribution as the default, motivated by the observation that concept frequencies in natural data follow power laws (Piantadosi, 2014; Ayonrinde, 2024; Michaud et al., 2023): $p_i \propto i^{-\alpha}$, scaled to lie in $[p_{\min}, p_{\max}]$. We also support linear decay (p_i interpolates linearly from $p_{\max}$ to $p_{\min}$), uniform random ($p_i \sim \text{Uniform}(p_{\min}, p_{\max})$), and constant ($p_i = p$). Arbitrary distributions can be plugged in.

Firing magnitudes. When a feature fires, its coefficient is sampled from a rectified Gaussian: $c_i = \text{ReLU}(\mu_i + \sigma_i \epsilon_i)$, $\epsilon_i \sim \mathcal{N}(0, 1)$. The per-feature mean μ_i and standard deviation σ_i can each take constant, linear-interpolated, exponential-interpolated, or folded-normal forms (e.g., $\sigma_i \sim |\mathcal{N}(\mu_\sigma, \sigma_\sigma^2)|$); the folded normal creates heterogeneous magnitude variability. Arbitrary distributions over μ_i and σ_i are supported.

F. SAE training procedures

We train all SAEs using the Adam optimizer (Kingma & Ba, 2014) with learning rate 3×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and batch size 1024. The learning rate decays linearly to zero over the final third of training. All SAEs have width 4096 and are trained on 200M samples from SynthSAEBench. Below we describe architecture-specific training details.

F.1. BatchTopK SAEs

BatchTopK SAEs (Bussmann et al., 2024) use a soft top-k selection that allows the number of active features to vary per sample while maintaining a target average L0 across the batch. The target L0 is set directly via the k parameter, so no autotuning is required.

To prevent dead latents, we use the TopK auxiliary loss from Gao et al. (2024). This auxiliary loss has dead latents reconstruct the residual error from live latents, providing gradient signal to features that would otherwise receive none. Following the heuristic from Gao et al. (2024), we set $k_{\text{aux}} = D/2$ where D is the input dimension, and scale the loss by $\min(\text{num_dead}/k_{\text{aux}}, 1)$ to reduce its magnitude when few latents are dead.

F.2. JumpReLU SAEs

Our JumpReLU training procedure follows the original procedure in Rajamanoharan et al. (2024) matching the procedure used by SAEBench, except we tune some hyperparameters to perform better in the synthetic setting. We use initial JumpReLU threshold of 1.0, bandwidth of 1.0, and use a latent norm of 0.5 at initialization. We find that with these settings, JumpReLU SAEs perform well on SynthSAEBench and we do not see many dead latents.

For sparsity control, we use an initial penalty of 1.0 with no warm-up and immediately adjust using an L0 coefficient autotuner (Appendix G) to hit a target L0.

F.3. Standard L1 SAEs

For standard L1 SAEs, we follow the procedure from Olah et al. (2024), training with a weighted combination of reconstruction MSE and an L1 penalty on feature activations. The L1 coefficient is warmed up linearly over the first third of training, after which we use the L1 coefficient autotuner (Appendix G) to achieve target L0 values. No auxiliary loss

is used.

F.4. Matryoshka SAEs

We train Matryoshka SAEs (Bussmann et al., 2025) using the BatchTopK activation function with nested prefixes. We use prefix sizes $\mathcal{M} = \{128, 512, 2048, 4096\}$. Each prefix is trained to reconstruct the input independently.

While the original Matryoshka SAEs work uses the standard TopK SAE auxiliary loss described in Appendix F.1, we find that using a Matryoshka-optimized auxiliary loss results in better performance, especially at low L0.

Matryoshka auxiliary loss. The standard TopK auxiliary loss (Gao et al., 2024) trains dead latents to reconstruct the residual error of the full SAE. However, in a Matryoshka SAE, dead latents at early prefixes (e.g., the first 128 latents) face a very different reconstruction residual than dead latents at later prefixes. Training all dead latents against the full SAE's residual provides a poor learning signal for latents in smaller prefixes, since those latents need to help reconstruct a much larger residual.

Our Matryoshka auxiliary loss instead computes a separate auxiliary loss for each matryoshka prefix. For prefix $m \in \mathcal{M}$, let $\hat{a}_m$ be the reconstruction using the first m latents, and let the dead latents within the range $[m_{\text{prev}}, m)$ be denoted $\mathcal{D}_m$. The auxiliary loss for prefix m is:

$$\mathcal{L}_{\text{aux},m} = s_m \cdot \|W_{\text{dec},\mathcal{D}_m} f_{\text{aux},m} - (a - \hat{a}_m)_{\text{detach}}\|_2^2 , \quad (28)$$

where $f_{\text{aux},m}$ are the top-$\min(k_{\text{aux}}, |\mathcal{D}_m|)$ activations among dead latents in $[m_{\text{prev}}, m)$, using only the corresponding portion of the encoder pre-activations. The scale factor $s_m = \min(|\mathcal{D}_m|/k_{\text{aux}}, 1)$ reduces the loss magnitude when few latents in that prefix are dead. The residual $(a - \hat{a}_m)$ is detached to prevent the auxiliary loss from affecting the main reconstruction pathway.

This per-prefix formulation ensures that dead latents in early prefixes receive gradient signal appropriate to their level's reconstruction error, rather than the much smaller residual of the full SAE.

We explore dead latents further in Appendix H.

G. Autotuning sparsity coefficients

For SAE architectures that use a sparsity-inducing loss (standard L1 SAEs and JumpReLU SAEs), the sparsity coefficient λ controls the trade-off between reconstruction and sparsity. However, the relationship between λ and the resulting L0 sparsity is nonlinear and model-dependent, making it difficult to train SAEs at a specific target L0 for fair comparison. We implement an autotuning controller that dynamically adjusts a multiplier m on the sparsity coefficient during training to achieve a target L0.

Controller design. We use a rate-dampened integral controller (Åström & Murray, 2021) that adjusts the effective sparsity coefficient $\lambda_{\text{eff}} = \lambda \cdot m$ based on the deviation from target L0. The controller maintains:

- A smoothed L0 estimate $\bar{\ell}_t$ using exponential moving average (EMA) with smoothing factor α
- A smoothed rate of L0 change $\dot{\ell}_t$ (the derivative estimate)

At each training step, given batch L0 measurement ℓ_t:

$$\bar{\ell}_t = \alpha \bar{\ell}_{t-1} + (1 - \alpha)\ell_t \quad (29)$$

$$\dot{\ell}_t = \alpha_r \dot{\ell}_{t-1} + (1 - \alpha_r)(\bar{\ell}_t - \bar{\ell}_{t-1}) \quad (30)$$

where $\alpha = 0.99$ and $\alpha_r = 0.95$ are the smoothing factors for position and rate respectively.

Gain scheduling. Gain scheduling (Rugh & Shamma, 2000) adapts controller parameters based on operating conditions. The key insight is that when the system is converging toward the target (error decreasing), we should reduce the controller gain to prevent overshoot. We detect convergence when the error and rate have opposite signs:

$$\text{converging} = (\bar{\ell}_t - \ell^*) \cdot \dot{\ell}_t < 0 \quad (31)$$

where ℓ^* is the target L0. When converging, we reduce the gain by a factor $\gamma_c = 0.01$.

Bounded adjustment. To ensure stable behavior, we use a tanh nonlinearity to bound the adjustment magnitude:

$$\Delta = K_i \cdot g \cdot \tanh\left(\left|\frac{\bar{\ell}_t - \ell^*}{\ell^*}\right| \cdot s\right) \quad (32)$$

where $K_i = 3 \times 10^{-4}$ is the integral gain, $g \in \{\gamma_c, 1\}$ is the scheduled gain, and $s = 10$ is the gain scale. The multiplier is then updated multiplicatively:

$$m_{t+1} = \begin{cases} m_t \cdot (1 + \Delta) & \text{if } \bar{\ell}_t > \ell^* \\ m_t \cdot (1 - \Delta) & \text{if } \bar{\ell}_t < \ell^* \end{cases} \quad (33)$$

The multiplier is clamped to $[0.01, 100]$ to prevent extreme values.

Integration with SAE training. For standard L1 SAEs, the autotuner modulates the L1 coefficient: $\lambda_1^{\text{eff}} = \lambda_1 \cdot m$. For JumpReLU SAEs, it modulates the L0 penalty coefficient similarly. The autotuner state is updated after each training batch, and the new multiplier is applied to the next batch's loss computation.

This approach allows us to train SAEs at precisely matched L0 values across different architectures, enabling fair comparison of their reconstruction and feature recovery quality.

H. Dead latents in SynthSAEBench

Dead latents are a problem in SAE training in general, with SynthSAEBench being no exception. We evaluate dead latents over 1M samples for the experiments in this section.

H.1. Dead latents in BatchTopK and Matryoshka SAEs

BatchTopK and especially Matryoshka BatchTopK SAEs seem to struggle with dead latents at low L0 in SynthSAEBench, even with the aux loss reviving dead latents. We can see this in Figure 8, where dead latents increase at L0=15 for BatchTopK and especially Matryoshka SAEs. We find that these latents are actually not entirely "dead", as they will still fire occasionally, but far less frequently than our dead latent window. This suggests that SAEs are settling into poor local minima.

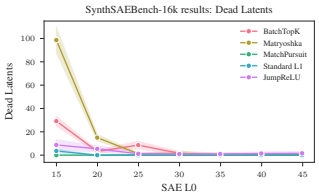

Figure 8. Dead latents vs L0 for SynthSAEBench SAEs. Shaded area is 1 stdev with 5 random seeds.

We also find that using the Matryoshka-optimized variant of the auxiliary loss described in Appendix F.4 results in a better Matryoshka SAE with fewer dead latents, especially at low L0. We show SAE quality results comparing these auxiliary losses for Matryoshka SAEs in Figure 9, and show dead latents in Figure 10.

H.2. Dead latents in Matching Pursuit and Standard L1 SAEs.

Matching Pursuit (MP) SAEs seem to never have any dead latents in SynthSAEBench, no matter the setting. This is impressive and a clear benefit of this architecture. Standard L1 SAEs also did not have any problems with dead latents in our experiments as long as the L1 coefficient is linearly warmed up as suggested by Olah et al. (2024).

I. SynthSAEBench sample generation performance

We next investigate the performance characteristics of the synthetic data generation process as a function of number of features in the synthetic model. We use our base SynthSAEBench model, but vary the number of features in the model from 2^7 (128) to 2^{20} (1M). We keep the same 3-level mutually-exclusive hierarchy scaled relative to the size of the data model. We then sample 100 batches of size 1024 on an Nvidia H100 GPU and benchmark the sample throughput of the model. Results are shown in Figure 11.

Feature generation ranges from 600K samples / sec (2.5 min for 100M samples) for models with under 1000 features to 7K samples / sec (4hr for 100M samples) for a model with 1M features. We decided on 16k as a good compromise between these two extremes. At 16k features, the model generates at 300K samples / sec, or 5 min for 100M samples.

J. Noise in SAEBench metrics

One motivation for this work is that existing SAE benchmarks like SAEBench, while extremely important and indispensable for SAE architecture development, tend to be noisy. For instance, we show SAEBench Spurious Correlation Removal (SCR), Targeted Probe Perturbation (TPP), and Sparse Probing metrics from the SAEBench paper for Gemma-2-2b layer 12 width 16k SAEs in Figure 12.

While there are clear trends in some metrics, overall there is still a lot of random noise that makes it difficult to make fine-grained judgements about SAE architecture differences. Cutting through this noise requires running many seeds of SAEs, which is often infeasibly expensive for LLM SAEs.

K. Exploring Superposition

How much pairwise feature interference does a given hidden dimension impose, and how does it scale with the number of features? Below we give a closed-form characterisation of ρ_{mm} for random unit-norm directions, which is essentially indistinguishable from the optimum in the regimes relevant to neural networks.

Distribution of pairwise cosine similarity. For two independent unit-norm vectors $u, v \in \mathbb{R}^D$ drawn uniformly from the sphere, the squared cosine similarity $\langle u, v\rangle^2$ follows a $\mathrm{Beta}(1/2, (D-1)/2)$ distribution. Letting F_D denote its CDF, the maximum absolute cosine similarity of one direction against the other $N-1$ directions has CDF

$$P\left(\max_{j\neq i} |\langle u_i, u_j\rangle| \leq x\right) \;=\; \left[F_D(x^2)\right]^{N-1}, \tag{34}$$

and $\mathbb{E}[\rho_{\text{mm}}]$ follows by direct integration, with no Monte Carlo simulation required. Spherical packing bounds further show that, in the regime $N \ll 2^D$ that covers all practical neural network widths, optimally-placed directions achieve essentially the same ρ_{mm} as random ones, so the analysis applies whether or not features are explicitly orthogonalised.

Asymptotic scaling. Taylor expanding Eq. (34) around $x = 0$ yields the simple approximation

$$\rho_{\text{mm}} \;\approx\; \sqrt{\frac{\ln N}{D}}. \tag{35}$$

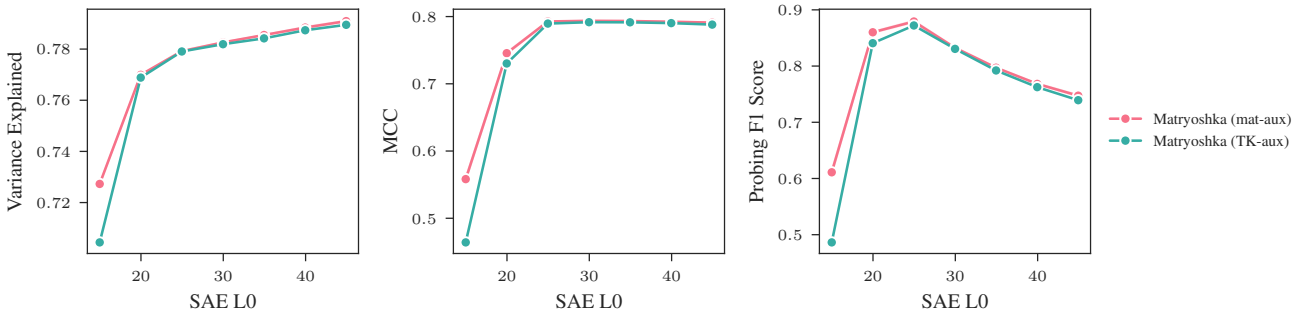

Figure 9. Comparing Matryoshka SAEs trained with standard TopK auxiliary loss (TK-aux) and a Matryoshka-optimized TopK auxiliary loss (mat-aux). The matryoshka-optimized loss results in better SAEs, but the difference is especially pronounced at low L0.

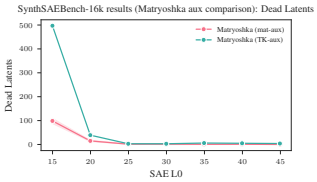

Figure 10. Dead latents for Matryoshka SAEs trained with standard TopK auxiliary loss (TK-aux) and a Matryoshka-optimized TopK auxiliary loss (mat-aux). The matryoshka-optimized loss results in fewer dead latents, especially pronounced at low L0.

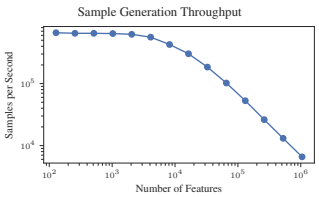

Figure 11. Synthetic model throughput by number of features.

Two consequences follow. First, interference shrinks as $O(1/\sqrt{D})$ in the hidden dimension. Second, the dependence on the number of features is only logarithmic: scaling N by $100\times$ adds only $\sqrt{\ln 100} \approx 2.1$ to the numerator, an effect easily absorbed by a modest increase in D. Hidden dimension dominates feature count in determining how much superposition is feasible.

Implications for SynthSAEBench. Figure 13 plots ρ_{mm} from Eq. (34) against D for N ranging from 10^5 to 10^{13}. Even at $N = 10^{13}$, ρ_{mm} falls below 0.1 once $D \geq 6000$. Conversely, SynthSAEBench ($N = 16{,}384$, $D = 768$) yields $\rho_{\text{mm}} \approx 0.15$, which is substantially more pairwise interference than would arise for trillions of features at a modern LLM hidden dimension. This is roughly equivalent to the amount of superposition we would expect in 1 Billion features in 2k hidden dim, roughly the size of Gemma-2-2b, which we feel is a reasonable baseline.

L. Logistic regression probes on SynthSAEBench

In Section 6.1, we note that the best SAE architecture (Matryoshka BatchTopK) achieves a peak probing F1 of approximately 0.88, consistent with LLM SAE findings that SAEs underperform supervised probes (Kantamneni et al., 2025). To quantify this gap on SynthSAEBench, we train logistic regression probes directly on hidden activations to classify ground-truth feature firings.

Setup. We sample 2M activations from SynthSAEBench and evaluate the first 4,096 features, which are the highest-frequency features due to the Zipfian ordering of firing probabilities. We train one linear probe per feature simultaneously using a batched logistic regression model (a single weight matrix $W \in \mathbb{R}^{F \times D}$ and bias $b \in \mathbb{R}^F$, where F is the number of probed features). Training uses Adam with learning rate 3×10^{-3}, cosine annealing over 10,000 steps, batch size 4,096, L_2 weight decay of 10^{-3}, and class-imbalance-corrected binary cross-entropy loss. We use an 80/20 train/test split, and tune a per-feature classification threshold on a 200K-sample validation subset drawn from the training set by sweeping 499 thresholds and selecting the one that maximizes F1 per feature.

Results. Results are shown in Table 1 in the main text. The probes achieve a mean F1 of 0.974 and mean AUC of 0.9999, substantially outperforming the best SAE probing F1 of ≈0.88. This confirms that the gap between SAE probing and supervised probing observed in LLMs (Kantamneni et al., 2025) is reproduced in SynthSAEBench, and that this gap is not an artifact of LLM evaluation noise but reflects a genuine limitation of current SAE architectures.

M. Extended results

We include extended results for the experiments in the paper.

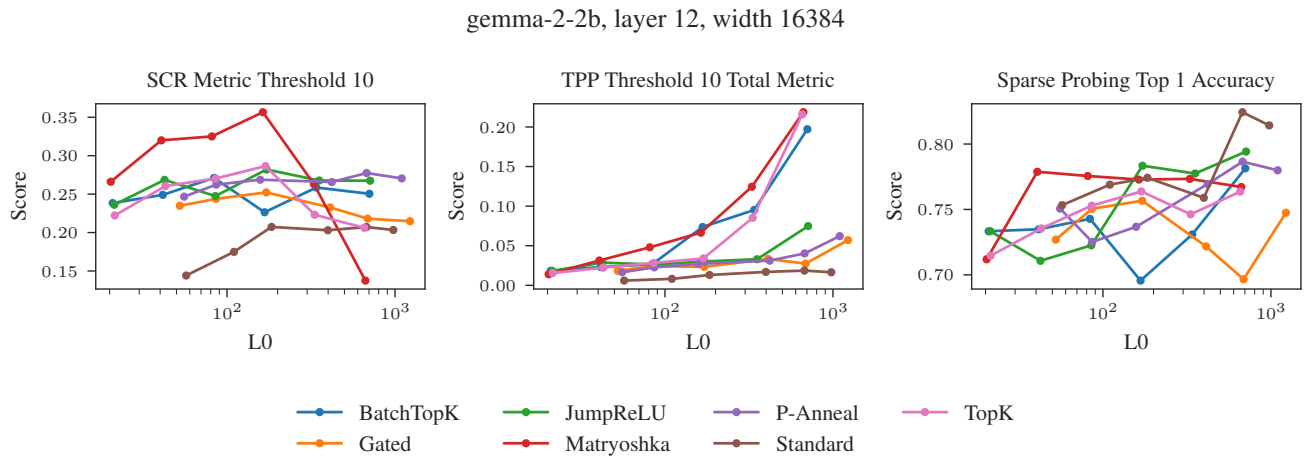

Figure 12. SAEBench SCR, TPP, and Sparse Probing metrics for Gemma-2-2b layer 12 width 16k SAEs.

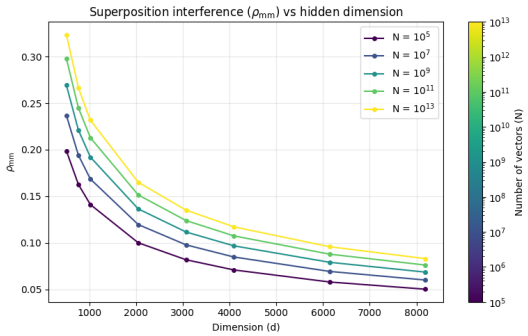

Figure 13. Mean max absolute cosine similarity ρ_{mm} as a function of hidden dimension D, computed analytically from Eq. (34) for random unit-norm feature directions, with the number of features N ranging from 10^5 to 10^{13}.

M.1. SynthSAEBench extended results

Full results for SynthSAEBench are shown in Figure 14.

M.2. Ablation: feature correlation strength

Next, we explore the effect of varying the strength of the random correlations between features. We vary the correlation scale used to generate the random correlation matrix from 0 to 0.25, and train SAEs with L0=25. Results are shown in Figure 15.

The effect of varying correlation strength are less dramatic than varying superposition, but still interesting. We see that variance explained increases for all SAEs as correlation increases except for Matryoshka SAEs, but this appears to be due to dead latents rather than a fundamental architectural issue. This is consistent with previous work showing that SAEs are able to exploit feature correlation to increase reconstruction by mixing correlated features (Chanin et al., 2025a; Chanin & Garriga-Alonso, 2025). We see slight decreases in probing F1 score as correlation increases, but mixed results on MCC that are harder to judge.

Likely the effect of feature correlation in SynthSAEBench is overshadowed by the effect of superposition noise and hierarchy (itself an even more extreme form of correlation).

M.3. Ablation: feature correlation rank

Next, we explore the effect of varying the rank of the low-rank correlations between features. We vary the rank from 1 to 100, and train SAEs with L0=25. We use correlation strength 0.075 throughout. Results are shown in Figure 16.

The effect of varying correlation rank is effectively negligible. This is not surprising, since the correlation rank just changes the types of correlation patterns that are possible, but does not change the underlying strength or quantity of correlations.

M.4. Ablation: feature firing probability

We perform an ablation study where we decrease the base firing probabilities of all features by a multiplier, effectively decreasing the L0 of the model. We evaluate each SAE at L0$= 25 * p$ where p is the firing multiplier. Results are shown in Figure 17. We see broadly the same pattern as in the main experiments, with Matryoshka being best at quality metrics, and Matching Pursuit having the best reconstruction. Interestingly, Matching Pursuit SAEs increase variance explained with higher L0 while other SAEs do not.

M.5. Ablation: feature firing stdev

We perform an ablation study where we set all features to have the same firing magnitude stdev, and then vary the stdev from 0.5 to 10.0. Each SAE is trained at L0=25. Results are shown in Figure 18. At high stdev, F1 score and variance

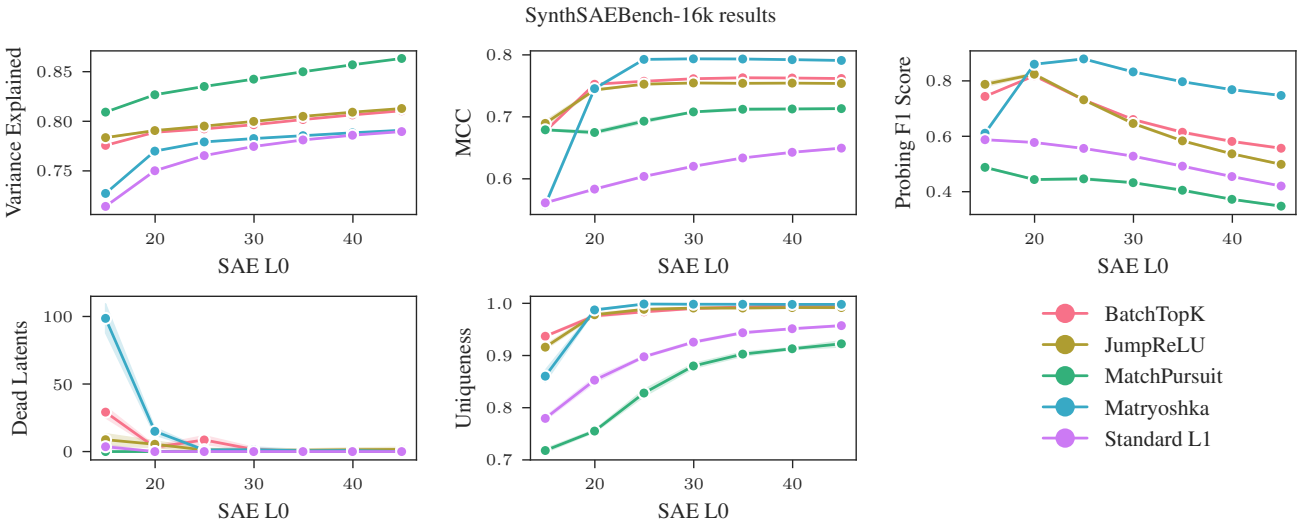

Figure 14. Full results for SynthSAEBench. Shaded area represents 1 stdev with 5 seeds per SAE.

explained drops for every SAE. This makes sense since it is hard to distinguish a low-magnitude firing from superposition noise when there is such a wide range of potential firing magnitudes for each feature. Interestingly, however, standard L1 SAEs achieve the best MCC of all SAEs at high firing magnitude stdev. We are not sure why this is, but it could be worth studying further to try to understand why a simple L1 penalty does so well in this regime compared to the other SAE architectures.

M.6. Ablation: deeper hierarchy

We perform an ablation study using a much deeper hierarchy (128 root nodes, 2 children per node, max depth of 6). All SAEs are trained at L0=25. Results are shown in Figure 19. We see the same general pattern as in SynthSAEBench, where the Matryoshka SAE is best by quality metrics (MCC and F1), despite poor reconstruction (variance explained). The MP-SAE achieves best reconstruction, but scores poorly on quality metrics. JumpReLU and BatchTopK are in-between on both SAE quality and reconstruction, and the Standard SAE performs poorly on all metrics.

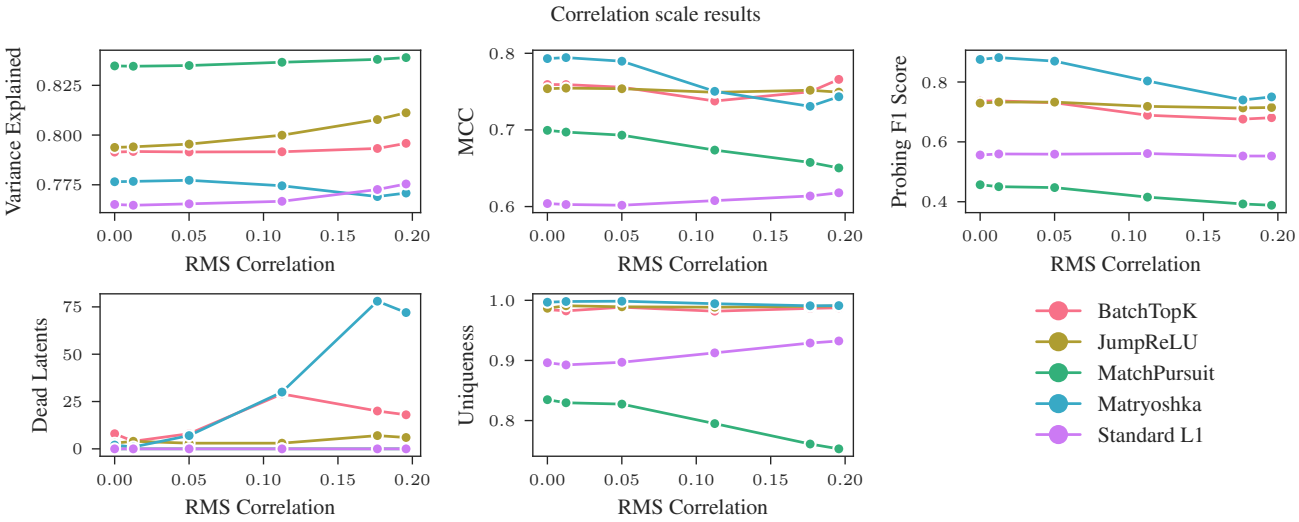

Figure 15. Results varying correlation strength, while keeping the remaining model hyperparameters set at default values for SynthSAEBench-16k.

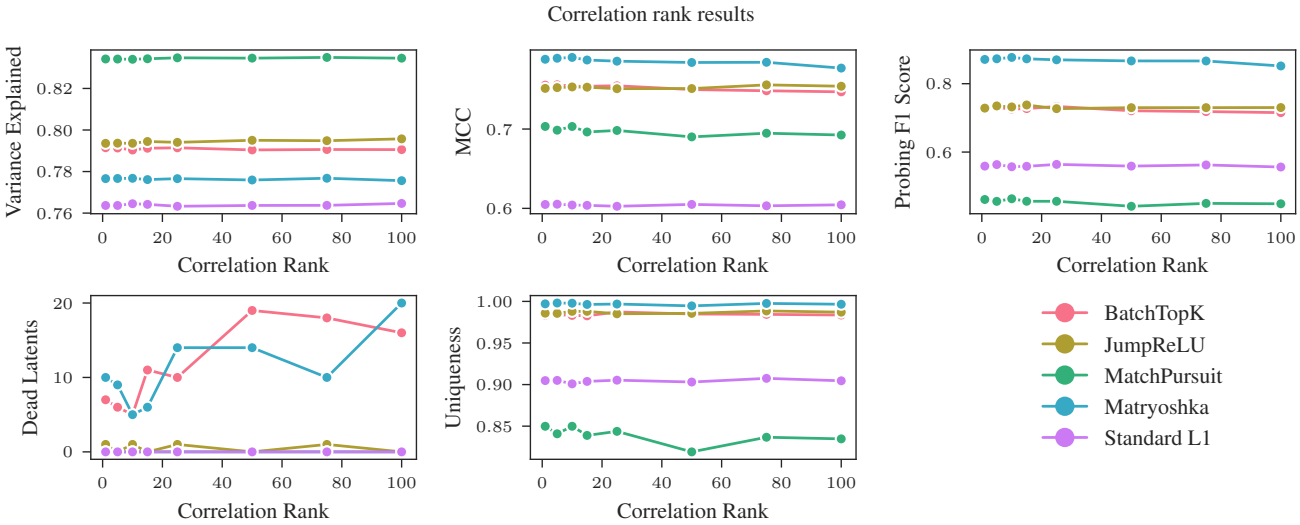

Figure 16. Results varying correlation rank, with correlation strength 0.075, and keeping the remaining model hyperparameters set at default values for SynthSAEBench-16k.

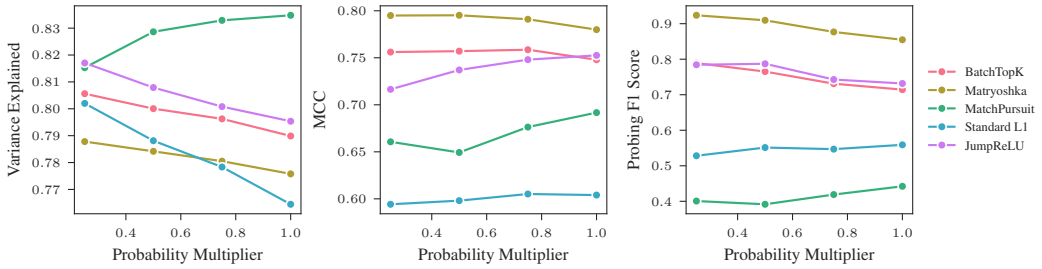

Figure 17. Ablation study varying the base firing probabilities of the SynthSAEBench model. We scale the L0 of the SAEs to match the scaled firing probabilities based on the default L0=25 (L0=12.5 when probability multiplier is 0.5). We see broadly the same pattern as in the main experiments, with Matryoshka being best at quality metrics, and Matching Pursuit having the best reconstruction. Interestingly, Matching Pursuit SAEs increase variance explained with higher L0 while other SAEs do not.

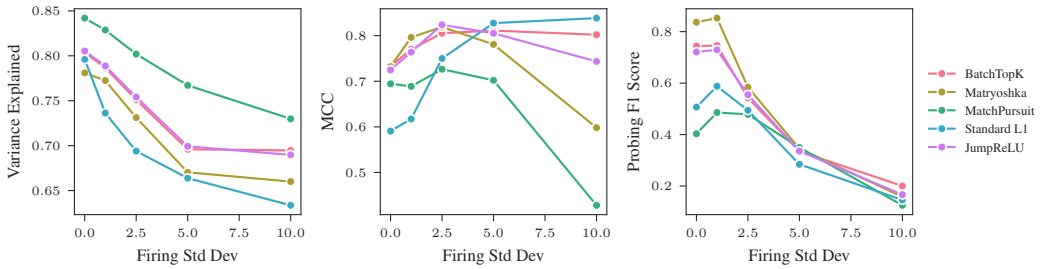

Figure 18. Ablation study varying the feature firing magnitude stdev of the SynthSAEBench model. We see broadly the same pattern as in the main experiments, with Matryoshka being best at quality metrics, and Matching Pursuit having the best reconstruction. Interestingly, we see a dichotomy between MCC and F1 score at high firing stdev, where high stdev improves MCC but reduces F1 score for all SAEs. This makes sense as high firing magnitude variance makes feature directions more prominent for the SAE, but means there are more low-magnitude samples that are difficult for the SAE to disambiguate from superposition noise.

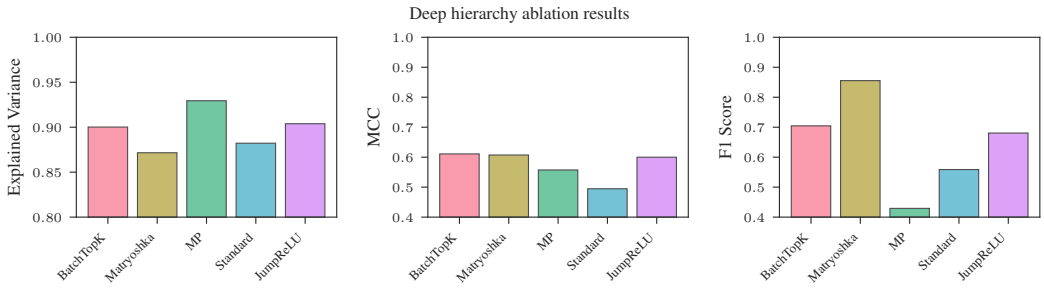

Figure 19. Ablation study using a much deeper hierarchy (128 root nodes, 2 children per node, max depth of 6). All SAEs are trained at L0=25. We see the same general pattern as in SynthSAEBench.