Kronecker Factorization Improves Efficiency and Interpretability of Sparse Autoencoders

Vadim Kurochkin [1] **Yaroslav Aksenov** [1] **Daniil Laptev** [1] **Daniil Gavrilov** [1] **Nikita Balagansky** [1]

Abstract

Sparse Autoencoders (SAEs) decompose language-model activations into sparse, interpretable features, but standard encoders usually treat the latent dictionary as a flat set of independent coordinates, leaving hierarchy and feature interactions to emerge only implicitly. We propose **KronSAE**, an encoder-side hierarchical SAE that factorizes the latent space into heads and forms post-latent features as pairwise compositions of lower-dimensional pre-latents using mAND, a differentiable AND-like interaction. This imposes a compositional co-activation prior while remaining compatible with standard SAE objectives and variants such as TopK, Matryoshka, and Switch SAEs. Under equal-token-budget training, KronSAE achieves reconstruction quality comparable to strong baselines, improves several interpretability and absorption metrics, better reflects correlated feature structure, and yields encoder parameter and FLOP reductions as an additional benefit.

1. Introduction

Interpreting the hidden states of large language models remains a central challenge for transparency and controllability in AI systems (Elhage et al., 2023; Heap et al., 2025). Sparse autoencoders (SAEs) have become a standard tool for this problem: they decompose dense activations into sparse latent features, enabling circuit-level analysis (Marks et al., 2025) and concept discovery in model internals (Elhage et al., 2023; Cunningham et al., 2024; Gao et al., 2025). Most modern SAEs use an encoder to produce feature activations and a decoder as a learned dictionary of feature atoms. This design is effective, but it usually treats the latent dictionary as a flat collection of independently encoded coordinates. As a result, higher-level structure among features, such as hierarchy, compositionality, or systematic co-activation, is not built into the architecture and must either emerge implicitly during training or be recovered after training through separate analysis.

This flat-latent default is not obviously aligned with the structure of language-model activations. Natural language data is highly structured: words and concepts have recurring statistical relationships, and transformer hidden states inherit regularities from those relationships. Recent work argues that even low-order language statistics, especially pairwise co-occurrence, can induce stable geometric structure in learned representations (Karkada et al., 2026). This suggests a corresponding design principle for SAEs: if correlations and co-activations in hidden states reflect recurring linguistic and conceptual structure, then the encoder need not remain fully unconstrained. Instead, an SAE can introduce an inductive bias that makes structured feature compositions easier to learn.

Recent SAE architectures have begun to make latent organization explicit. Switch SAEs introduce a routed expert decomposition of the encoder/decoder computation (Mudide et al., 2025), while Matryoshka SAEs impose a nested-dictionary hierarchy over features (Bussmann et al., 2025). These approaches show that explicit structure can be useful for interpretability and scaling. However, a complementary direction remains underexplored: imposing hierarchy by changing the default structure of the encoding process itself. Rather than waiting for feature interactions to appear only as post hoc correlations among flat latents, we ask whether the encoder can directly bias feature formation toward structured compositions.

The same factorization also yields a computational side benefit. Since KronSAE replaces a dense projection to F independent latents with projections to $h(m+n)$ pre-latents that generate $F = hmn$ post-latents, it reduces encoder parameters and per-token FLOPs. In this paper, however, we treat this reduction as an additional benefit rather than the primary motivation. Our main experimental comparison is based on an **equal token budget**: models are trained on the same number of activation tokens, and FLOP/parameter savings are reported as efficiency consequences of the

[1]T-Tech. Correspondence to: Yaroslav Aksenov <y.o.aksenov@t-tech.dev>.

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

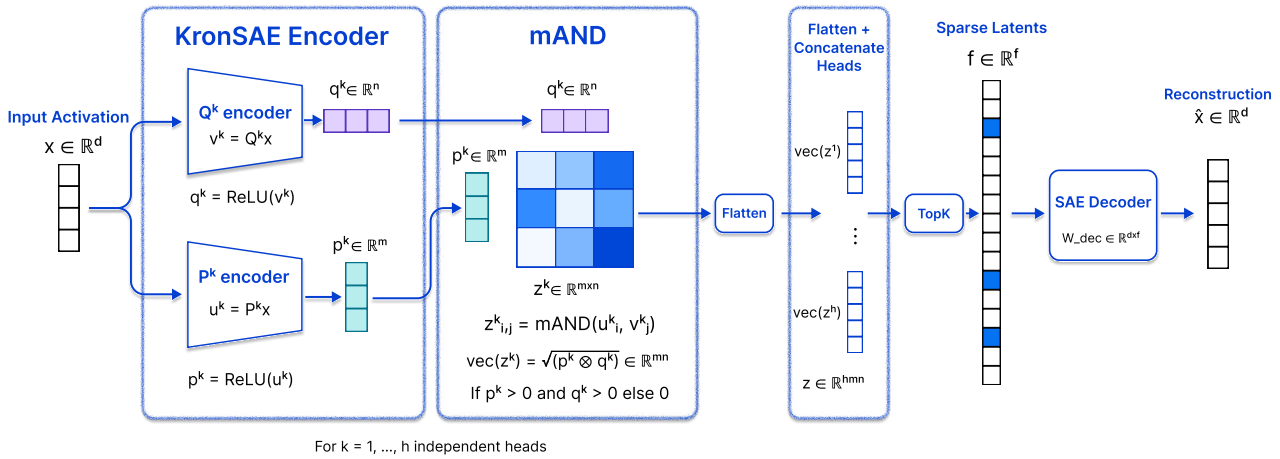

Figure 1. Overview of the proposed KronSAE method. We decompose $x \in^d$ into two vectors and then interact via mAND. Sparse latents is combination of features combined by mAND operator.

hierarchical encoder. This framing separates the question of whether the inductive bias improves feature organization from the secondary question of how much computation the factorization saves.

This work makes three primary contributions:

1. We identify encoder-side hierarchy as a missing design axis for sparse autoencoders. Motivated by the structured statistics of language-model activations, we study whether feature interactions can be encouraged directly through the encoding process rather than discovered only after training.

2. We introduce **KronSAE**, a Kronecker-factorized SAE encoder equipped with the mAND interaction. The architecture forms post-latent features from reusable pre-latent factors, giving a two-level compositional structure while remaining compatible with standard SAE objectives and existing architectural variants.

3. We show on multiple language models that KronSAE decreases feature absorption and yields more interpretable latents.

4. Under equal-token-budget training, we evaluate reconstruction, absorption, automated interpretability, and correlation-structure diagnostics across models, sparsity levels, and KronSAE hyperparameters. The results characterize the reconstruction–interpretability trade-off induced by the hierarchical encoder and report encoder FLOP and parameter reductions as an additional efficiency benefit.

2. Related Work

Sparse Autoencoders. Early work demonstrated that SAEs can uncover human-interpretable directions in deep models (Cunningham et al., 2024; Templeton et al., 2024), but extending them to the large dictionary size F for modern language models with large hidden dimension d is expensive: each forward pass still requires a dense $\mathcal{O}(F\,d)$ encoder projection. Most prior optimization efforts focus on the decoder side: for example, Gao et al. (2025) introduce a fused sparse–dense TopK kernel that reduces wall-clock time and memory traffic, while Rajamanoharan et al. (2024a) decouple activation selection from magnitude prediction to improve the sparsity–MSE trade-off. Separately, Rajamanoharan et al. (2024b) propose JumpReLU, a nonlinearity designed to mitigate shrinkage of large activations in sparse decoders. (O'Neill et al., 2025) argues that classical SAE encoders are not expressive enough and more complex encoders can yield more faithful results without compromising interpretability.

Conditional Computation. These schemes avoid instantiating a full dictionary per token by routing inputs to a subset of expert SAEs via a lightweight gating network (Mudide et al., 2025) by employing the Mixture-of-Experts ideas (Shazeer et al., 2017), but still incur a dense per-expert encoder projection.

Factorizations and Logical Activation Functions. Tensor product representations (Smolensky, 1990) have been utilized to represent the compositional structures in dense embeddings, closely resembling our idea of AND-like compositions, and recent studies have been extended TPR to study the transformer hidden states and in-context learning (Soulos et al., 2020; Smolensky et al., 2024). Separately, tensor-factorization methods have been used to compress

large weight matrices in language models (Edalati et al., 2021; Wang et al., 2023). In parallel, differentiable logic activations were introduced to approximate Boolean operators in a smooth manner (Lowe et al., 2021). Our method synthesizes these lines of work: we embed a differentiable AND-like gate into a Kronecker-factorized efficient encoder to build compositional features while preserving end-to-end differentiability.

3. Method

Preliminaries. Let $\mathbf{x} \in \mathbb{R}^d$ denote an activation vector drawn from a pretrained transformer. A conventional TopK SAE (Gao et al., 2025) produces a reconstruction $\hat{\mathbf{x}}$ of $\mathbf{x}$ via

$$\mathbf{f} = \mathrm{TopK}\left(\mathrm{ReLU}(W_{\text{enc}}\mathbf{x}+\mathbf{b}_{\text{enc}})\right),\ \hat{\mathbf{x}} = W_{\text{dec}}\mathbf{f}+\mathbf{b}_{\text{dec}}, \quad (1)$$

where $W_{\text{enc}} \in \mathbb{R}^{F\times d}$ and $W_{\text{dec}} \in \mathbb{R}^{d\times F}$ are dense matrices and $\mathbf{f} \in \mathbb{R}^F$ is a sparse vector retaining only the K largest activations. The encoder cost scales as $\mathcal{O}(Fd)$ per token.

KronSAE. Our method *reduces the encoder's computational cost* while *imposing a compositional inductive bias* of the latents. We decompose the latent space into h independent heads, and each head k is parameterised by the composition of two thin matrices $P^k \in \mathbb{R}^{m\times d}$ (composition base) and $Q^k \in \mathbb{R}^{n\times d}$ (composition extension), with dimensions $m \le n \ll d$ and $F = h\,m\,n$. The *pre-latents*

$$\mathbf{p}^k = \mathrm{ReLU}(\mathbf{u}^k) \qquad \text{and} \qquad \mathbf{q}^k = \mathrm{ReLU}(\mathbf{v}^k), \quad (2)$$

with $\mathbf{u}^k = P^k\mathbf{x}$ and $\mathbf{v}^k = Q^k\mathbf{x}$ acting as the elements from which compositional features would be built, are combined through an element-wise interaction kernel independently in each head:

$$z^k_{i,j} := \mathrm{mAND}(u^k_i, v^k_j) := \begin{cases} \sqrt{u^k_i\, v^k_j}, & u^k_i > 0 \text{ and } v^k_j > 0, \\ 0, & \text{otherwise}, \end{cases} \quad (3)$$

where $\mathbf{z}^k \in \mathbb{R}^{m\times n}$; it is then flattened in a row-major order to a vector equivalent to the element-wise square root of the Kronecker product of $\mathbf{p}^k$ and $\mathbf{q}^k$. Assuming that $\mathrm{vec}(\cdot)$ is in row-order, we get

$$\mathrm{vec}(\mathbf{z}^k) = \sqrt{\mathrm{vec}\left(\mathbf{p}^k(\mathbf{q}^k)^\top\right)} = \sqrt{\mathbf{p}^k \otimes \mathbf{q}^k} \quad (4)$$

Concatenating heads and applying TopK yields *post-latents* $\mathbf{f} \in \mathbb{R}^F$. Because $z_{i,j} > 0$ iff $p_i > 0$ and $q_j > 0$, the activation events follow an AND-like rule: let $\mathcal{P}$ and $\mathcal{Q}$ be the sets of hidden state vectors on which p_i and q_j, respectively, are active, and let $\mathcal{F}$ be the set of inputs on which $z_{i,j} > 0$, then before TopK selection $\mathcal{P} \cap \mathcal{Q} = \mathcal{F}$. This architectural prior on co-activations induce specific semantical structure, and many heads exhibit semantically coherent interaction patterns under this prior, as we empirically verify in Section 5.3.

The square root in equation 3 is a monotone rescaling that keeps output magnitudes comparable; in practice we add a small ε for numerical stability, applied before taking a square root so it does not introduce new nonzero activations. See additional discussion in Appendix B.

The encoder cost per token drops from $\mathcal{O}(Fd)$ to $\mathcal{O}\big(h(m+n)d\big)$ (see Appendix A.2). KronSAE thus reduces encoder FLOPs and parameter count without routing overhead. It is compatible with sparse decoder kernels (Gao et al., 2025), MoE-style routing, different loss functions and activations, as we show in the next section.

4. Experiments

We train SAEs on the residual streams of Qwen-2.5-1.5B-Base (Yang et al., 2024), Pythia-1.4B-deduped (Biderman et al., 2023), and Gemma-2-2B (Team et al., 2024) language models. Activations are collected on FineWeb-Edu, a filtered subset of educational web pages from the FineWeb corpus (Penedo et al., 2024). We measure reconstruction quality via explained variance (EV),

$$\mathrm{EV} = 1 - \frac{\mathrm{Var}(\mathbf{x}-\hat{\mathbf{x}})}{\mathrm{Var}(\mathbf{x})},$$

so that 1.0 is optimal, and use automated interpretability pipeline (Bills et al., 2023; Paulo et al., 2024) and SAE Bench (Karvonen et al., 2025) to evaluate properties of SAE features. We target for the needs of resource-constrained mechanistic interpretability settings where efficiency and interpretability are prioritized over maximal reconstruction performance, and 100M-1B token budgets are widely adopted (Bussmann et al., 2025; Kharlapenko et al., 2025; Heap et al., 2025; Mudide et al., 2025; Karvonen et al., 2025), so we choose 125M, 500M, 1B and 2B token budgets for the experiments.

Our experiments evaluate KronSAE variants of TopK, Matryoshka TopK, and Switch SAEs against their corresponding baselines (see detailed setup in Appendices A and D). We focus on these TopK-style baselines because they are the standard compute-efficient SAE family in recent large-scale work and admit highly optimized sparse decoder kernels; KronSAE targets the remaining encoder projection bottleneck. We include the gating-inspired JumpReLU and Gated SAE mechanisms and their Kronecker-factorized versions as an ablation in Appendix A.8. We address three questions:

1. Does KronSAE achieve explained variance (EV) comparable to the baseline SAEs?
2. Which design choices (nonlinearity, (m, n, h)) primarily govern EV in KronSAE?
3. How do these choices affect latent properties, feature absorption, and automated interpretability?

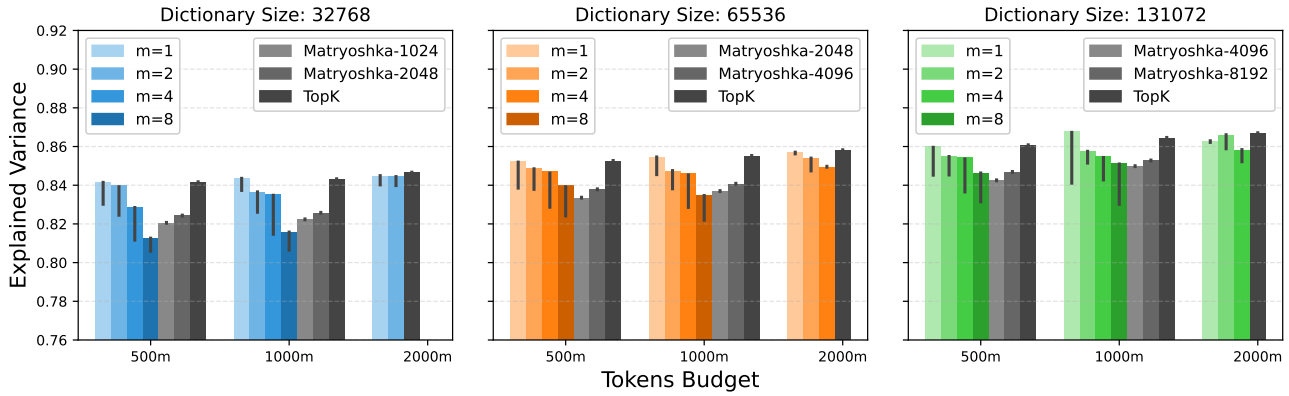

Figure 2. Maximum performance for KronSAE vs. TopK SAE vs. Matryoshka TopK SAE on Qwen-1.5B for different dictionary sizes F and token budgets. KronSAE with lower number of parameters is on-par with the baseline on EV; the gap narrows with larger dictionary size.

4.1. Ablations

Reconstruction performance. As indicated on Figure 2, KronSAE achieves EV comparable to TopK given lower number of trainable parameters and outperforms Matryoshka SAE. The performance gap narrows when increasing the dictionary size, which indicate the potential scalability of our method for large dictionaries. See also result on Gemma-2 2B model in Appendix A.3. We observe the same qualitative behavior across additional layers; these results are reported in Appendix A.5. Complementary MSE measurements are provided in Appendix A.6 and show the same overall reconstruction trends.

Decomposition hyperparameters. We systematically vary the number of heads h and the per-head base dimension m (with $n = F/(mh)$). From the Figure 3, we conclude that lower m and higher h tend to yield higher reconstruction quality in these settings. We hypothesize this reflects greater flexibility from finer-grained pre-latents; Section 5.3 provides qualitative evidence. See also Appendices A.10 and A.9.

Composition activation. To isolate the impact of our mAND operator, we compare it to two simpler interaction kernels: (i) the element-wise product of ReLUs, $\mathrm{ReLU}(u) \cdot \mathrm{ReLU}(v)$, and (ii) the raw product $u \cdot v$. As reported in Table 1, under a 125M training budget, the mAND variant achieves the highest explained variance. More description of the mAND is provided in the Appendix B. See also our experiments with JumpReLU and Gated activation functions replacing TopK sparsity in Appendix A.8.

Sparsity Analysis. To evaluate performance across different sparsity budgets ($\ell_0 = \{16, 32, 64, 128\}$), we compare multiple SAE baselines (TopK SAE, Matryoshka SAE, Switch SAE) against their KronSAE variants at dictionary size $F = 2^{16}$ trained on 500 million tokens. As shown in Figure 4, KronSAE achieves reconstruction fidelity (measured by explained variance) comparable to baseline methods across all sparsity levels. Results for smaller dictionary size $F = 2^{15}$ are presented in Appendix A.4.

Table 1. Performance of different composition activations under a budget of 125M tokens.

F	(m, n, h)	Activation	EV
32768	$(2, 4, 4096)$	$\mathrm{mAND}(u, v)$	**0.8336**
		$\mathrm{ReLU}(u) \cdot \mathrm{ReLU}(v)$	0.8267
		$u \cdot v$	0.8237
	$(4, 8, 1024)$	$\mathrm{mAND}(u, v)$	**0.8220**
		$\mathrm{ReLU}(u) \cdot \mathrm{ReLU}(v)$	0.8191
		$u \cdot v$	0.8143
65536	$(2, 4, 8192)$	$\mathrm{mAND}(u, v)$	**0.8445**
		$\mathrm{ReLU}(u) \cdot \mathrm{ReLU}(v)$	0.8328
		$u \cdot v$	0.8297
	$(4, 8, 2048)$	$\mathrm{mAND}(u, v)$	**0.8350**
		$\mathrm{ReLU}(u) \cdot \mathrm{ReLU}(v)$	0.8297
		$u \cdot v$	0.8251

4.2. Absorption

The notorious challenge in SAE interpretability is *feature absorption*, where one learned feature becomes a strict subset of another and consequently fails to activate on instances that satisfy the broader concept but not its superset representation (e.g. a "starts with L" feature is entirely subsumed by a "Lion" feature) (Chanin et al., 2024).

Figure 5 reports three absorption metrics measured via SAEBench (Karvonen et al., 2025) across sparsity levels $\ell_0 \in \{16, 32, 64, 128\}$: (1) the *mean absorption fraction*,

measuring the proportion of features that are partially absorbed; (2) the *mean full-absorption score*, quantifying complete subsumption events; and (3) the *mean number of feature splits*, indicating how often a single conceptual feature fragments into multiple activations. We use with dictionary size $F = 2^{16}$ and compare TopK SAE, Matryoshka SAE (Bussmann et al., 2025), and their KronSAE version, since they impose different hierarchical priors (see Appendix B for a discussion of how Matryoshka SAE and TopK SAE structure differs). Across all ℓ_0, KronSAE variants tend to reduce first two scores relative to the TopK SAE baseline, while maintaining a similar rate of feature splits.

We hypothesize that the observed reductions in absorption are consistent with two complementary architectural biases:

1. **AND-like behavior.** The gate enforces co-activation: a post-latent can be active only when both pre-latents are active. This can reduce full-absorption events in our metrics by limiting activation when a parent is inactive. See additional description of this mechanism in Appendix B.

2. **Head-wise Cartesian decomposition.** Dividing the latent space into h independent subspaces (each with its own $m \times n$ grid of primitive interactions) encourages correlated features to localize within a head, which can reduce cross-head absorption.

See results with dictionary size $F = 2^{15}$ in Appendix A.4. We also validate the result on Pythia models and observe similar trends, see the Appendix A.10.

5. Analysis

In this section we examine the properties of the latents in KronSAE compared to the TopK architecture.

The design of our architecture was also inspired by the observation that many features within a single SAE are correlated. KronSAE introduces structural bias that forces

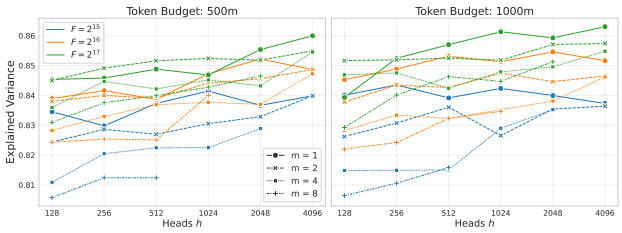

Figure 3. Dependency of EV on head count h (on the x-axis) and base dimension m under 500M and 1B token budgets in the iso-FLOPs setup. Higher h and smaller m are associated with improved reconstruction quality due to higher expressivity of pre-latents to encode semantics and larger parameter budgets.

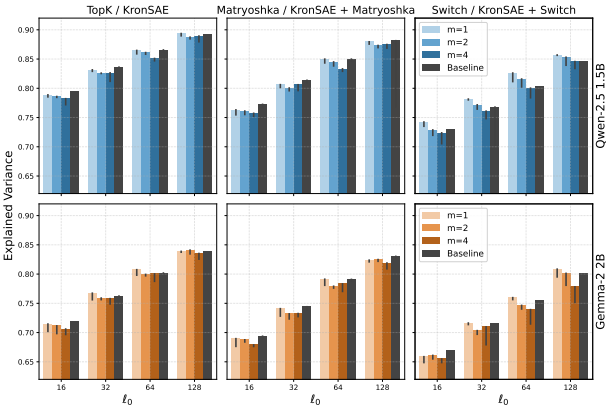

Figure 4. Maximum performance for baselines and their KronSAE modifications for different sparsity levels. KronSAE variants, despite using fewer trainable parameters, achieve reconstruction quality comparable to or better than the unmodified baselines.

the post-latents to co-occur with their pre-latents (see also Appendix B). We therefore perform two complementary analyses: (i) a synthetic toy model with controllable ground truth covariance to test recoverability of imposed correlations, and (ii) correlation measurements on language data to characterize how features tend to group under the imposed prior. These analyses do not establish ground-truth semantic hierarchies in real LLMs. As in Matryoshka SAEs, we treat the KronSAE architectural structure as an explicit inductive bias and use it to study how this bias shapes the learned dictionary. We analyze the resulting AND-gated co-activation patterns and report automated interpretability metrics, relating these results to the stable reconstruction–interpretability trade-off controlled by (m, n, h).

5.1. Toy Model of Correlation

To evaluate how well different sparse autoencoder architectures recover underlying correlation patterns, we construct a controlled experiment using a synthetic, block-structured covariance model. Input vectors $\mathbf{x} \in \mathbb{R}^F$ are sampled from a normal distribution (with $\mu = 0, \sigma = 1$). We then perform a Cholesky decomposition $S = LL^\top$ on the covariance matrix S and set $\bar{\mathbf{x}}_{\text{sparse}} = L\ \text{TopK}(\text{ReLU}(\mathbf{x}))$, so that $\bar{\mathbf{x}}_{\text{sparse}}$ exhibits the desired structure.

We train autoencoder (AE) to reconstruct $\bar{\mathbf{x}}_{\text{sparse}}$ following the (Elhage et al., 2022):

$$\hat{\mathbf{x}} = \text{ReLU}(W^\top W \cdot \bar{\mathbf{x}}_{\text{sparse}} + \mathbf{b}). \tag{5}$$

We collect hidden states of dimension $d = 64$ $(W \cdot \bar{\mathbf{x}}_{\text{sparse}})$ from AE and then train TopK SAE and our proposed KronSAE with $F = 256$ and topk $= 8$ to reconstruct it. After training, we extract the decoder weight matrices W_{dec} from each SAE, match its latents with the autoencoder latents by solving the quadratic assignment problem (Vogelstein

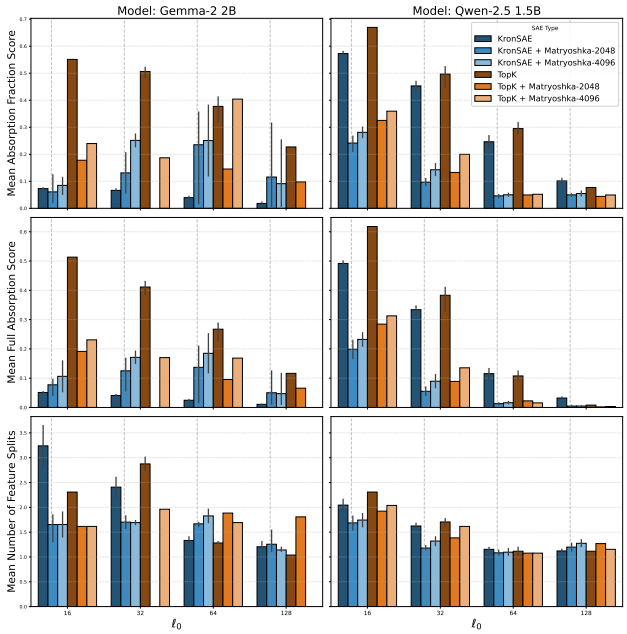

Figure 5. Feature absorption metrics on Qwen-2.5 1.5B and Gemma-2 2B. KronSAE configurations (various m, n) exhibit lower mean absorption fractions and full-absorption scores across different ℓ_0 and selected baselines.

et al., 2015) (see Appendix C for motivation and details), and compute the covariance $C_{\text{dec}} = W_{\text{dec}} W_{\text{dec}}^\top$. Result is shown in Figure 6.

To quantify how closely each model's feature correlations mirror the original structure of S, we employ the RV coefficient, defined as $RV(S, C) = \text{trace}(SC)/\sqrt{\text{trace}(S^2)\,\text{trace}(C^2)}$. In our experiments, KronSAE achieves higher RV than TopK SAE in this synthetic setting, indicating that compositional encoder more faithfully reconstructs the original feature relation. See also additional experiments in Section C, where we provide further empirical intuition for how KronSAE aligns with

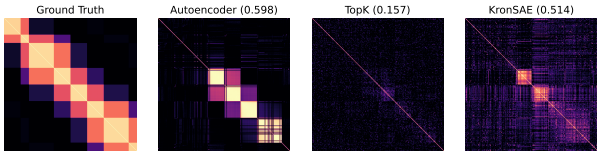

Figure 6. We generate data with covariance matrix that consists of blocks with different sizes on diagonal and off the diagonal (left panel). We then examine the decoder-weight covariance $W_{\text{dec}} \cdot W_{\text{dec}}^\top$ to assess feature-embedding correlations and compute the RV score to quantify the similarity between learned and ground truth covariance matrices. Second panel show feature embeddings for trained autoencoder $W_{\text{enc}} \cdot W_{\text{enc}}^T$. The third panel demonstrates that a TopK SAE recovers these correlation structures weakly, as indicated by a relatively low RV coefficient (0.157) even after optimal atom matching. In contrast, KronSAE (right panel) more closely reflects the original block patterns.

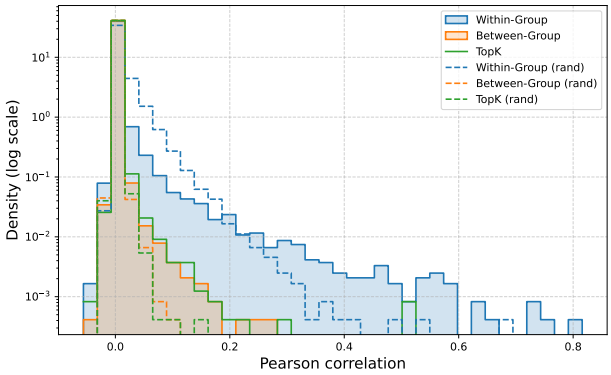

Figure 7. Correlations between features in KronSAE with $m = 4, n = 4$ within a head and with features from other heads. Our design induces higher correlations within a group, which also gets stronger after training, although SAE have also learned correlated features from different heads. TopK SAE shows less correlation structure than within-groups correlation in KronSAE.

correlations present in the data.

5.2. Correlations in SAEs Trained on Language

To examine the correlation structure of features learned in our SAE, we have calculate the correlations on 5k texts from the training dataset. For each feature we compute the mean correlation with features within its head and with all other features, and compare the randomly initialized KronSAE with $m = 4, n = 4$ with the trained one. To isolate the effect of our initialization procedure, we initialize the weights of SAE from the uniform distribution. As shown in Figure 7, correlations are indeed significantly higher within a single head and higher than for random SAE, which suggest that our choice to impose the correlated structure in SAE latents works as intended.

5.3. Analysis of Learned Features

In this section we compare KronSAE and TopK SAE in terms of interpretability and feature properties, and we analyze the properties of groups in KronSAE. For this, we choose the 14th layer of Qwen2.5-1.5B and a dictionary size of 32k features, of which the first 3072 were selected. KronSAE was chosen with $m = 4, n = 4$. We run for 24M tokens total to collect data. Our interpretation pipeline follows the common methodology: LLM interprets the activation patterns (Bills et al., 2023) and we evaluate obtained interpretations using the *detection* score and the *fuzzing* score (Paulo et al., 2024).

For each selected feature, among the standard mean activation value and frequency, we calculate two additional metrics. Low values of *token entropy* suggest that feature activates more frequently on small number of tokens, thus it is token-specific; high value of *multitoken ratio* indicates that

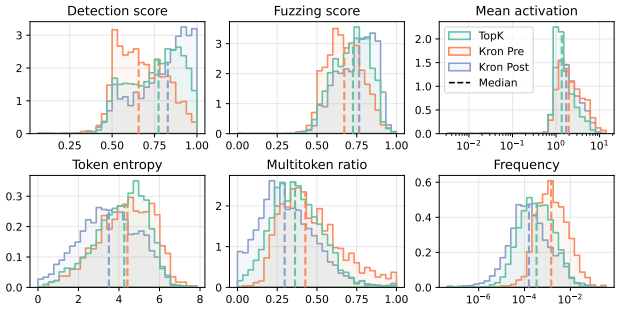

Figure 8. Distribution of properties for TopK SAE and KronSAE ($m = 4, n = 4$) with 32k dictionary size trained on Qwen2.5-1.5B. Pre and Post suffixes denote pre- and post- latents, and y-axis indicate density. Token entropy shows the entropy of the distribution of tokens on which feature has activated, and the multitoken ratio measures how often does feature activate in a single sequence. Our SAE achieves better interpretability scores by learning specialized feature groups, indicated by lower activation frequency and lower variance in activated tokens.

feature tends to activate multiple times in a single sentence. We have observed that both these metrics have notable negative correlation with the final interpretability scores and therefore they provide useful signal to assess the potential score without calculating it.

For more details on the data collection and interpretation pipeline, see Appendix D. For additional analysis of properties of learned features, additional comparison with baselines and discussion about tradeoff between reconstruction performance and interpretability, see Appendix E.

SAE properties and encoding mechanism. We observe that the features learned by KronSAE are more specific, indicated by lower values of the token entropy and multitoken ratio, and higher interpretability scores than TopK, as shown in Figure 8. Since post-latents are significantly more interpretable than corresponding pre-latents, we hypothesize the hidden mechanism for encoding and retrieval of the required semantics.

By examining activating examples and interpretations of latents, we observe that pre-latents may carry multiple distinct and identifiable modes of activation, such as composition base element 3 in head 23 shown in Table 2, and be very abstract compared to resulting post-latents. Polysemanticity of pre-latents is expected to be a consequence of reduced "working" number of encoder latents, since we decompose the full dictionary size and reduce the encoder capacity.

Thus, we hypothesize that the encoding of specific semantics in our SAE may be done via magnitude, which we validate by examining the activation examples. For the above mentioned pre-latent, the "comparison" part is encoded in the top 75% quantile, while the "spiritual" part is mostly met in the top 25% quantile, and the "geographical" part is mainly encoded in the interquartile range. We also consider but do not investigate the possibility that it may depend on the context, e.g. when the model uses the same linear direction to encode different concepts when different texts are passed to it.

Semantic retrieval and interpretable interactions. Heads usually contain a groups of semantically related pre-latents, e.g. in head 136 there are three base elements and one extension covering numbers and ordinality, two extension elements related to geographical and spatial matters, one question-related base and one growth-related extension. Interestingly, most post-latents for this head have higher interpretability score than both its parent pre-latents, which is unusual.

The retrieval happens primarily via the mechanism closely resembling the logical AND circuit, where some pre-latent works as the bearer of multiple semantics, and the corresponding pre-latent (base or extension) works as specifier. An illustrative example is shown in Table 2: we see that the base contains three detectable sub-semantics, and each extension then retrieves the particular semantics.

Other types of interaction may occur, such as appearance of completely new semantics, for example composition between base 3 and extension 1 in Table 2 where medical terminology arises and could not be interpreted as simple intersection between two pre-latents semantics. Another example is a case of head 3 where base 3 has sub-semantics related to technical instruments and extension 2 have semantics related to the posession and necessity, and their combination gives the therapy and treatment semantics which looks more like addition than intersection.

It is a frequent case that post-latent inherit semantics of only one parent, or the impact of another parent is not detectable, which usually happens if parent has a very broad interpretation and low score. However, it requires more sophisticated techniques to properly identify the fine-grained structure of interactions than just looking at the resulting latent descriptions, so we leave it to further work. Despite this, the AND-like behaviour is very common. See more examples in Appendix G.

Geometry of post-latents. Each post-latent vector has a vector representation in the residual stream represented by the corresponding column in W_{dec}, which is the approximation of overcomplete basis vectors we search for when training SAEs. Our architectural design leads to clustering of feature embeddings so that post-latents produced by same head, base or a extension elements are grouped in a tight cluster, and the geometry is dependent on hyperparameters h, m, n we choose, which is expected and may be useful for further applications such as steering. See more details in

Table 2. Interactions between composition base element 3 in head 23 and all extension elements in that head. Interaction happens in a way that closely resembles the Boolean AND operation: base pre-latent is polysemous, and the composition post-latent is the intersection, i.e. logical AND between parent pre-latents. See details in Section 5.3.

Component	Interpretation	Score
Base 3	Suffix "-like" for comparative descriptors, directional terms indicating geographical regions, and concepts related to spiritual or metaphysical dimensions	0.84
Extension elements and their compositions with base 3		
Extension 0	*Interpretation:* Comparative expressions involving "than" and "as" in contrastive or proportional relationships.	0.87
	Composition: Similarity or analogy through the suffix "-like" across diverse contexts.	0.89
Extension 1	*Interpretation:* Specific terms, names, and abbreviations that are contextually salient and uniquely identifiable.	0.66
	Composition: Medical terminology related to steroids, hormones, and their derivatives.	0.84
Extension 2	*Interpretation:* Spiritual concepts and the conjunction "as" in varied syntactic roles.	0.80
	Composition: Abstract concepts tied to spirituality, consciousness, and metaphysical essence.	0.93
Extension 3	*Interpretation:* Directional and regional descriptors indicating geographical locations or cultural contexts.	0.84
	Composition: Directional terms indicating geographical or regional divisions.	0.91

Appendix E.

6. Conclusion and Future Work

We introduce **KronSAE**, a sparse autoencoder architecture design that combines head-wise Kronecker factorization of latent space with a approximation of logical AND via mAND nonlinearity, inducing a co-activation prior on latent structure. Our approach allows to efficiently train interpretable and compositional SAE, especially in settings with limited compute budget or training data, while maintaining reconstruction fidelity and yielding more interpretable features by utilizing their correlations. Our analysis links these gains to the complementary effects of compositional latent structure and logical AND-style interactions, offering a new lens on how sparsity and factorization can synergise in representation learning.

Limitations. KronSAE introduces a clear and useful trade-off between interpretability, efficiency and reconstruction performance, and due to reduced number of trainable parameters in encoder it is expected to lag behind TopK SAE at large budgets. Our evaluation is limited to mid-sized transformer models; however, the main bottleneck there might be not the SAE itself, but the infrastracture required to handle these setups and the model inference.

Future Work. We identify three directions for extending this work: (i) *Transcoders and crosscoders.* Treat transcoders (Dunefsky et al., 2024) as implicit routers of information and investigate alternative logical gating functions (e.g. XOR or composite gates) to improve interpretability and circuit analysis; another extension is to generalize KronSAE to a crosscoder setting (Lindsey et al., 2024) and uncover interpretable, cross-level compositionality via logic operations. (ii) *Dynamic Composition.* Explore learnable tuning of both the number of attention heads and their dimensionality, enabling fine-grained decomposition into groups of correlated features at varying scales.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Balagansky, N., Maksimov, I., and Gavrilov, D. Mechanistic permutability: Match features across layers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MDvecs7EvO.

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L.,

and van der Wal, O. Pythia: A suite for analyzing large language models across training and scaling, 2023. URL https://arxiv.org/abs/2304.01373.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models, 2023. URL https://openai.com/index/language-models-can-explain-neurons-in-language-models/.

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., Nabeshima, N., Karvonen, A., and Nanda, N. Learning multi-level features with matryoshka sparse autoencoders. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=m25T5rAy43.

Chanin, D., Wilken-Smith, J., Dulka, T., Bhatnagar, H., and Bloom, J. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2024. URL https://arxiv.org/abs/2409.14507.

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

Dunefsky, J., Chlenski, P., and Nanda, N. Transcoders find interpretable LLM feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=J6zHcScAo0.

Edalati, A., Tahaei, M., Rashid, A., Nia, V. P., Clark, J. J., and Rezagholizadeh, M. Kronecker decomposition for gpt compression, 2021. URL https://arxiv.org/abs/2110.08152.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henigan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.

Elhage, N. et al. Decomposing language models with dictionary learning. *Transformer Circuits*, 2023. URL https://transformer-circuits.pub/2023/monosemantic-features.

Fel, T., Lubana, E. S., Prince, J. S., Kowal, M., Boutin, V., Papadimitriou, I., Wang, B., Wattenberg, M., Ba, D. E., and Konkle, T. Archetypal SAE: Adaptive and stable dictionary learning for concept extraction in large vision models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9v1eW8HgMU.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Heap, T. et al. Sparse autoencoders can interpret randomly initialized transformers. *arXiv preprint arXiv:2501.17727*, 2025. URL https://arxiv.org/abs/2501.17727.

Karkada, D., Korchinski, D. J., Nava, A., Wyart, M., and Bahri, Y. Symmetry in language statistics shapes the geometry of model representations, 2026. URL https://arxiv.org/abs/2602.15029.

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.

Kharlapenko, D., Shabalin, S., Conmy, A., and Nanda, N. Scaling sparse feature circuits for studying in-context learning. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025. URL https://openreview.net/forum?id=sdLwJTtKpM.

Lindsey, J., Templeton, A., Marcus, J., Conerly, T., Batson, J., and Olah, C. Sparse crosscoders for cross-layer features and model diffing, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Lowe, S. C. et al. Logical activation functions: Logit-space equivalents of probabilistic boolean operators. *arXiv preprint arXiv:2110.11940*, 2021. URL https://arxiv.org/abs/2110.11940.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

Mudide, A., Engels, J., Michaud, E. J., Tegmark, M., and de Witt, C. S. Efficient dictionary learning with switch sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=k2ZVAzVeMP.

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.

Paulo, G. and Belrose, N. Sparse autoencoders trained on the same data learn different features, 2025. URL https://arxiv.org/abs/2501.16615.

Paulo, G., Mallen, A., Juang, C., and Belrose, N. Automatically interpreting millions of features in large language models, 2024. URL https://arxiv.org/abs/2410.13928.

Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving sparse decomposition of language model activations with gated sparse autoencoders. In *NeurIPS*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/01772a8b0420baec00c4d59fe2fbace6-Abstract-Conference.html.

Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J., and Nanda, N. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders, 2024b. URL https://arxiv.org/abs/2407.14435.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1701.06538.

Smolensky, P. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artif. Intell.*, 46(1–2):159–216, November 1990. ISSN 0004-3702. doi: 10.1016/0004-3702(90)90007-M. URL https://doi.org/10.1016/0004-3702(90)90007-M.

Smolensky, P., Fernandez, R., Zhou, Z. H., Opper, M., and Gao, J. Mechanisms of symbol processing for in-context learning in transformer networks, 2024. URL https://arxiv.org/abs/2410.17498.

Soulos, P., McCoy, R. T., Linzen, T., and Smolensky, P. Discovering the compositional structure of vector representations with role learning networks. In Alishahi, A., Belinkov, Y., Chrupała, G., Hupkes, D., Pinter, Y., and Sajjad, H. (eds.), *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 238–254, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.23. URL https://aclanthology.org/2020.blackboxnlp-1.23/.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., Tsitsulin, A., Vieillard, N., Stanczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J.-B., Neyshabur, B., Bachem, O., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagic, A., Carl, A., Shen, A., Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C., Choquette-Choo, C. A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozińska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Plucińska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J. P., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., van Amersfoort, J., Gordon, J., Lipschultz, J., Newlan, J., yeong Ji, J., Mohamed, K., Badola, K., Black, K., Millican, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjoesund, L. L., Usui, L., Sifre, L., Heuermann, L., Lago, L., McNealus, L., Soares, L. B., Kilpatrick, L., Dixon, L., Martins, L., Reid, M., Singh, M., Iverson, M., Görner, M., Velloso, M., Wirth, M., Davidow, M., Miller, M., Rahtz, M., Watson, M., Risdal, M., Kazemi, M., Moynihan, M., Zhang, M., Kahng, M., Park, M., Rahman, M., Khatwani, M., Dao, N., Bardoliwalla, N., Devanathan, N., Dumai, N., Chauhan, N., Wahltinez, O., Botarda, P., Barnes, P., Barham, P., Michel, P., Jin, P., Georgiev, P., Culliton, P., Kuppala, P., Comanescu, R., Merhej, R., Jana, R., Rokni, R. A., Agarwal, R., Mullins, R., Saadat, S., Carthy, S. M., Cogan, S., Perrin, S., Arnold, S. M. R., Krause, S., Dai, S., Garg, S., Sheth, S., Ronstrom, S., Chan, S., Jordan, T., Yu, T., Eccles, T., Hennigan, T., Kocisky, T., Doshi, T., Jain, V., Yadav, V., Meshram, V., Dharmadhikari, V., Barkley, W., Wei, W., Ye, W., Han, W., Kwon, W., Xu, X., Shen,

Z., Gong, Z., Wei, Z., Cotruta, V., Kirk, P., Rao, A., Giang, M., Peran, L., Warkentin, T., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Sculley, D., Banks, J., Dragan, A., Petrov, S., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Borgeaud, S., Fiedel, N., Joulin, A., Kenealy, K., Dadashi, R., and Andreev, A. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

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., 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. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Vogelstein, J. T., Conroy, J. M., Lyzinski, V., Podrazik, L. J., Kratzer, S. G., Harley, E. T., Fishkind, D. E., Vogelstein, R. J., and Priebe, C. E. Fast approximate quadratic programming for graph matching. *PLoS ONE*, 10, 2015. URL https://api.semanticscholar.org/CorpusID:16927142.

Wang, P., Panda, R., Hennigen, L. T., Greengard, P., Karlinsky, L., Feris, R., Cox, D. D., Wang, Z., and Kim, Y. Learning to grow pretrained models for efficient transformer training. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cDYRS5iZ16f.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

A. Additional Details and Results

A.1. Experimental setup

Training details. All SAEs are optimized using AdamW with an initial learning rate of 8×10^{-4}, a cosine learning-rate schedule with a minimum LR of 1×10^{-6}, and a linear warmup for the first 10% of total training steps, auxiliary loss penalty equal to 0.03125. We use a global batch size of 8,192. We sweep over dictionary (latent) sizes of $F = 2^{15}$, $F = 2^{16}$, and $F = 2^{17}$ features. For our KronSAE variant, we further sweep the number of heads h and the per-head dimensions m and n such that $h \cdot m \cdot n$ equals the desired dictionary size. Regularization weights and auxiliary loss coefficients are kept constant throughout the runs to isolate the impact of architectural choices.

For all experiments, we spent about 330 GPU days on NVIDIA H100 80GB GPUs, including preliminary research.

SAE. For all experiments on Qwen-2.5, we train each SAE on activations from layer 14. Also for Pythia-1.4B we use layer 14 and for Gemma-2-2B we take activations from layer 12. For most of our experiments, we use sparsity level of $\ell_0 = 50$ non-zero activations per token.

Initialization. As observed by Gao et al. (2025), initializing the decoder as the transpose of the encoder ($W_{\text{dec}} = W_{\text{enc}}^\top$) provides a strong metric improvement. We adopt this strategy within KronSAE by partitioning W_{enc} into h head-wise blocks of shapes $m \times d$ and $n \times d$, denoted $\{P_i, Q_i\}_{i=1}^h$. For each head k, we define its decoded rows via additive composition:

$$C_k[i,j] \;=\; P_{k,i} \;+\; Q_{k,j}, \quad i = 1, \ldots, m,\; j = 1, \ldots, n.$$

Finally, flattening the matrices $\{C_k\}$ yields full decoder weight matrix $W_{\text{dec}} \in \mathbb{R}^{F \times d}$.

Matryoshka and Kron-based version. For Matryoshka SAE of dictionary size F we adopt the following experimental setup. We use most training settings from Training details. For Matryoshka SAE Bussmann et al. (2025) we define the parameter of dictionary group $S = [2^k, 2^k, \cdots 2^{k+i} \cdots 2^n]$, where $k < n$ and $\sum_{s \in S} s = F$. This is equivalent to nested sub-SAEs with dictionary sizes $\mathcal{M} = \{2^k, 2^k + 2^k, 2^k + 2^k + 2^{k+1}, \cdots, F\}$ and in our work we define this SAE as **Matryoshka-2$^{\mathbf{k}}$**. Training loss for Matryoshka is defined as follows:

$$\mathcal{L}(\mathbf{f}) = \sum_{m \in \mathcal{M}} ||\mathbf{x} - \mathbf{f}_{0:m} W_{0:m}^{dec} + b_{dec}||_2^2 + \alpha \mathcal{L}_{aux}, \tag{6}$$

where $\mathbf{f}$ is latent vector and $\mathcal{L}_{aux}$ is auxiliary loss used in Gao et al. (2025).

For Kron-based Matryoshka SAE we use same setup and same initialization from Initialization section. We choose m and n so that $F \mod mn = 0$, ensuring that no heads are shared between groups.

Switch SAE and Kron-based version. For the Switch SAE architecture proposed by Mudide et al. (2025), we configure all experiments with 8 experts and use identical initialization from KronSAE. In the Kron-based variant of this SAE, we distribute the heads equally across all experts, resulting in $F/(8mn)$ heads per expert. Hence we have less parameters in every expert and therefore we also reduce FLOPs for encoder.

A.2. FLOPs calculation and efficiency

We estimate the per-token floating-point operation (FLOP) cost of the SAE forward pass under the standard TopK decoding implementation: the encoder is a dense projection, while the decoder reconstructs using only the k active latents.

For a hidden dimension d, dictionary size F, and sparsity level k, the TopK SAE cost is

$$\text{FLOPs}_{\text{TopK}}(d, F, k) \;=\; dF \;+\; kd \;=\; d(F + k), \tag{7}$$

where dF corresponds to encoder matrix multiplication and kd corresponds to sparse decoding using the k selected features.

For KronSAE with h heads and factor sizes (m, n) (so $F = hmn$ post-latents), the encoder computes $(m + n)$ pre-latents per head via dense projections and then forms mn post-latents per head via the mAND interaction. This yields

$$\begin{aligned} \text{FLOPs}_{\text{KronSAE}}(d, m, n, h, k) \;&=\; dh(m+n) \;+\; mnh \;+\; kd \\ \approx\; dh(m+n) \;+\; kd \;&=\; d\big(h(m+n) + k\big), \end{aligned} \tag{8}$$

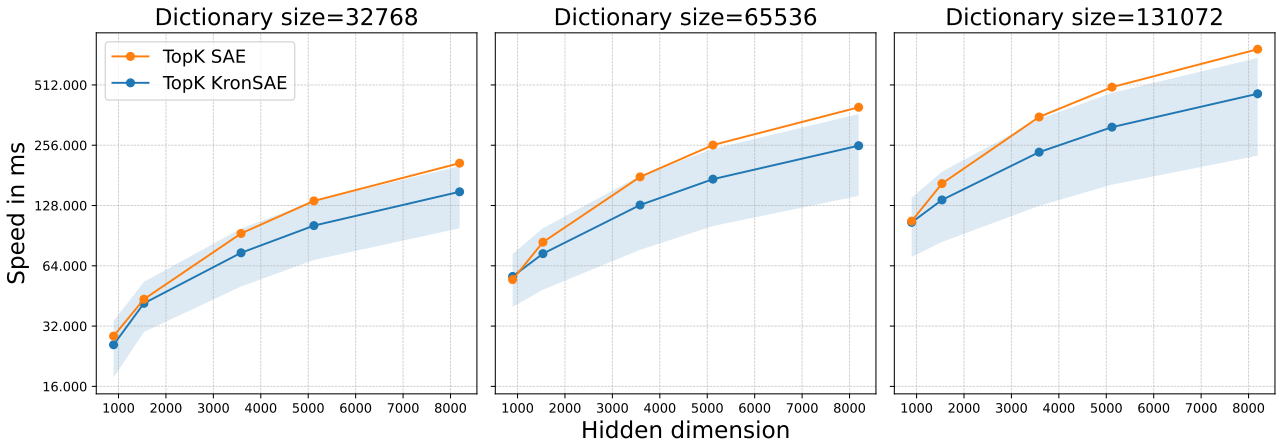

Figure 9. Speed comparision of TopK SAE with KronSAE across different hidden dimensions. We can see that KronSAE have better scaling properties than SAE with default encoder architecture.

where m, n, h accounts for the elementwise interaction over all (i, j) pairs in each head. In our typical settings, $d\,h(m+n)$ dominates m, n, h, so the approximation is accurate up to a small additive $\mathcal{O}(F)$ term.

We summarize the relative encoder+decoder compute via the ratio

$$\mathrm{FLOPs}_{\mathrm{diff}} = \frac{\mathrm{FLOPs}_{\mathrm{TopK}}(d, F, k)}{\mathrm{FLOPs}_{\mathrm{KronSAE}}(d, m, n, h, k)} \approx \frac{F + k}{h(m+n) + k}. \tag{9}$$

To corroborate the analytic scaling, we additionally benchmark the wall-clock time of a full SAE training step (forward + backward) for both methods under identical hardware and batching. Figure 9 reports how step time scales with the hidden dimension d. And 10 show an improvement in training with a lower number of FLOPs.

A.3. Scaling on Gemma-2 2B

To examine the method's generality, we conducted additional reconstruction experiments on Gemma-2 2B. As shown in Figure 11, our method achieves comparable or improved reconstruction performance across the evaluated token budgets.

A.4. Smaller dictionary size

To complement our larger-scale experiments, we further evaluate KronSAE's performance on a smaller dictionary size $F = 2^{15}$ with varying $\ell_0 = \{16, 32, 64, 128\}$ and **equal token budget**.

Sparsity. Following the experimental setup described in Section 4.1, we compare KronSAE against established baselines including TopK SAE, Matryoshka SAE, and Switch SAE . As shown in Figure 15, KronSAE consistently matches or exceeds the reconstruction performance of baseline architectures across all tested sparsity levels, while achieving these results with substantially fewer trainable parameters and FLOPs.

Feature absorption. We extend our feature absorption analysis from Section 4.2 to the smaller dictionary size, employing two distinct grouping schedules: $S_1 = [2048, 2048, 4096, 8192, 16384]$ and $S_2 = [1024, 1024, 2048, 4096, 8192, 16384]$. As shown in Figure 13 KronSAE modification demonstrates better performance, reducing feature absorption metrics.

A.5. Layerwise performance.

Additionally, we evaluate performance across different layers in Qwen-2.5-1.5B. For this experiment we fix the dictionary size to $F = 2^{15}$ and use the same hyperparameters as in the main experiments. As the Figure 14 shows, KronSAE matches the reconstruction quality (EV) of the TopK baseline across layers, suggesting that the Kronecker-factorized encoder does not degrade with depth in this setting.

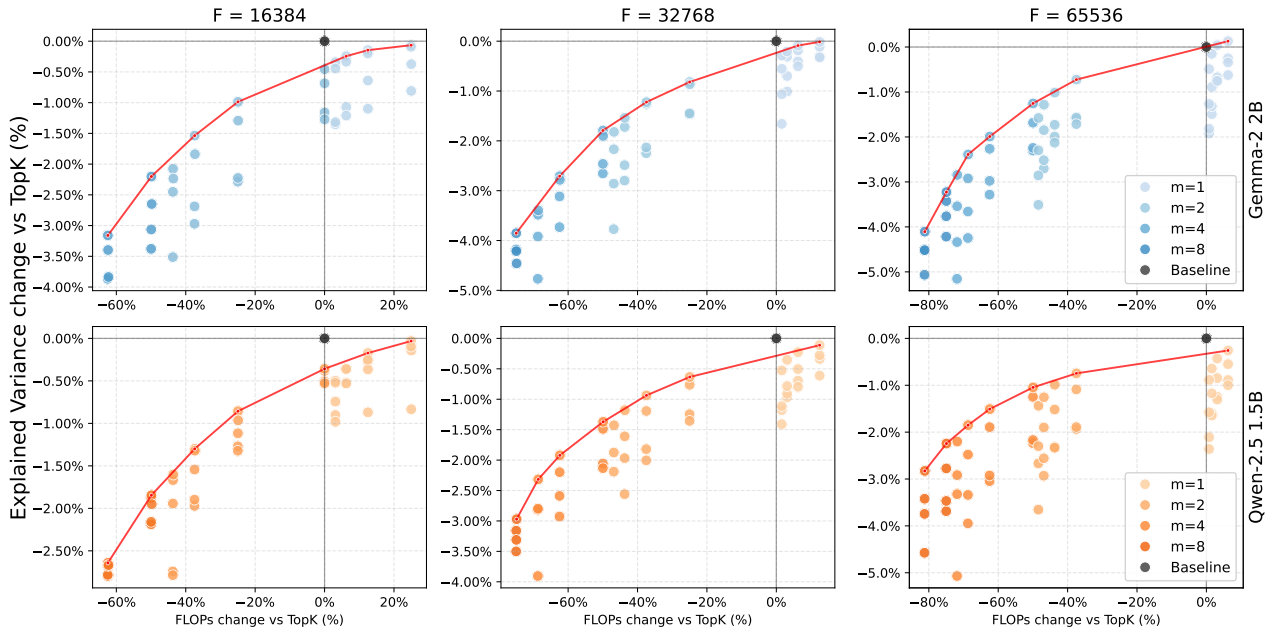

Figure 10. Paretto front of SAE performance with Kronecker encoder mechanism. With minimal EV changes we get up to -60% of reduce training number of FLOPs.

A.6. MSE performance.

We additionally report MSE results in Figure 15. The trends are consistent with the explained variance results: under the same fixed token budget, KronSAE matches the reconstruction quality of the TopK SAE baseline across the evaluated sparsity regime.

A.7. Absorption score on Gemma and Pythia

In section 4.2 we have analysed whether KronSAE achieves lower absorption score and have answered affirmatively. We also compare the results for Pythia 1.4B model and validate the improvements, as reported in the Table 3.

These results confirm that our compositional architectures improves the absorption score and feature consistency across various models from different families.

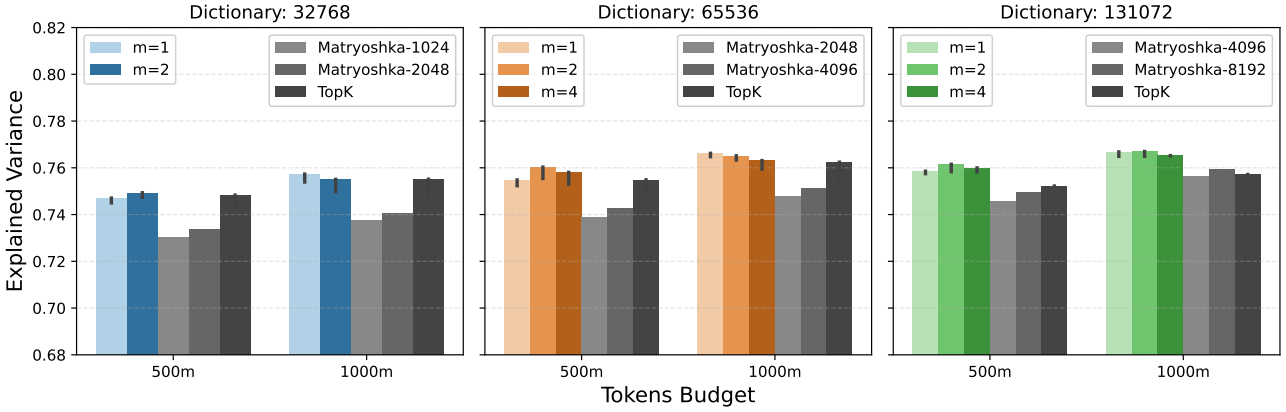

Figure 11. Performance comparision of KronSAE and TopK SAE. Across sparsity settings, KronSAE typically matches TopK's reconstruction performance and in some cases slightly outperforms it, while using fewer trainable parameters.

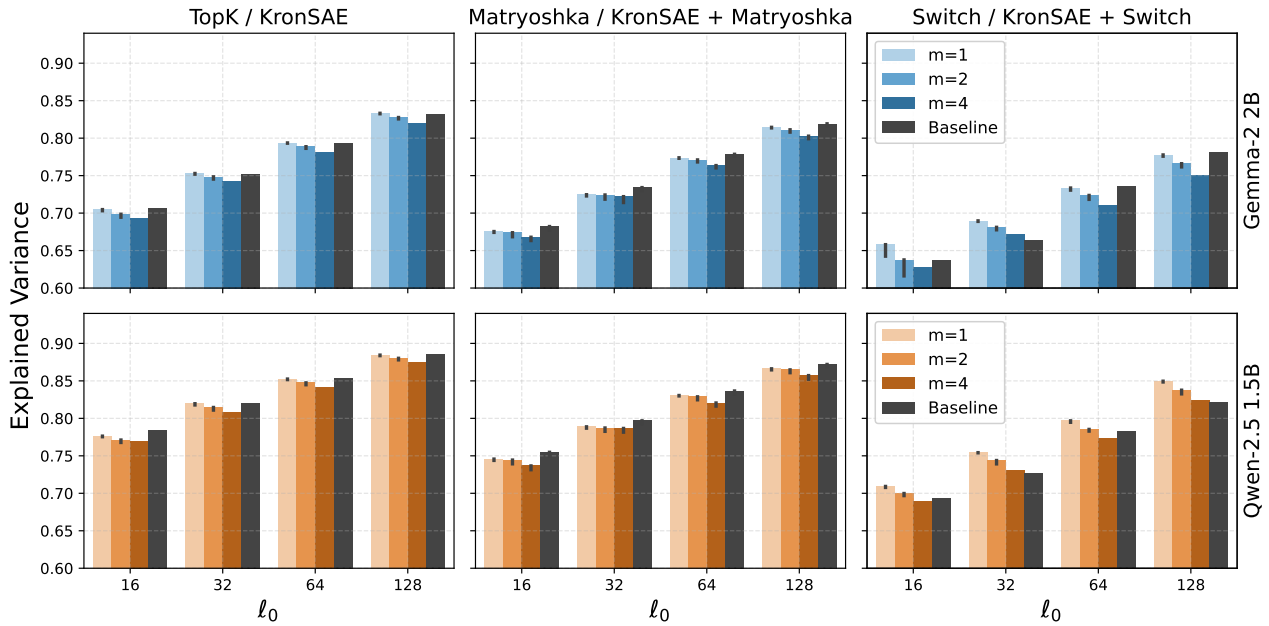

Figure 12. Performance comparision of KronSAE and TopK SAE under a fixed token budget. Across sparsity settings, KronSAE typically matches TopK's reconstruction performance and in some cases slightly outperforms it, while using much fewer trainable parameters.

Table 3. Absorption score calculated for Pythia 1.4B model. KronSAE shows lower score due to structured latent space and AND-like co-activation hierarchy between pre-latents and post-latents.

Model	SAE	$\ell_0 = 16$	$\ell_0 = 32$	$\ell_0 = 64$	$\ell_0 = 128$	$\ell_0 = 256$
Pythia	TopK	0.445	0.233	0.058	0.006	0.004
	KronSAE	0.244	0.129	0.033	0.007	0.003

A.8. Comparison with JumpReLU and Gated SAE

We provide experiments to compare KronSAE with an alternative activation mechanisms - JumpReLU and Gated SAE, and report explained variance under three sparsity levels in Table 4. Replacement of TopK with JumpReLU within KronSAE leads to a degraded performance relative to both JumpReLU SAE and KronSAE with TopK, also with degraded scaling over ℓ_0. This suggests that the architectural advantages of KronSAE interact most effectively with TopK's behaviour. For Gated SAEs, the reconstruction performance of Gated-KronSAE is on-par with baseline SAE.

A.9. Choice of m, n, h

We derive the following guidelines to train KronSAE: one should minimize the m and maximize the h to improve the reconstruction performance, and search for the most expressive configuration from feasible ones. Since m has more impact on EV, one should start from $m = 2$ in the search process, since it gives improved computational performance with on-par EV with full TopK training.

A.10. Pythia Suite

We have also conducted experiments with Pythia suite to understand the scaling over dictionary sizes F, hidden dimension d and base count m and trained SAEs on the middle layers of each model. As reported in Table 5, at larger F and fixed budget KronSAE start to outperforms TopK SAE, and with increasing d, difference between reconstruction performance of KronSAE variants became more pronounced.

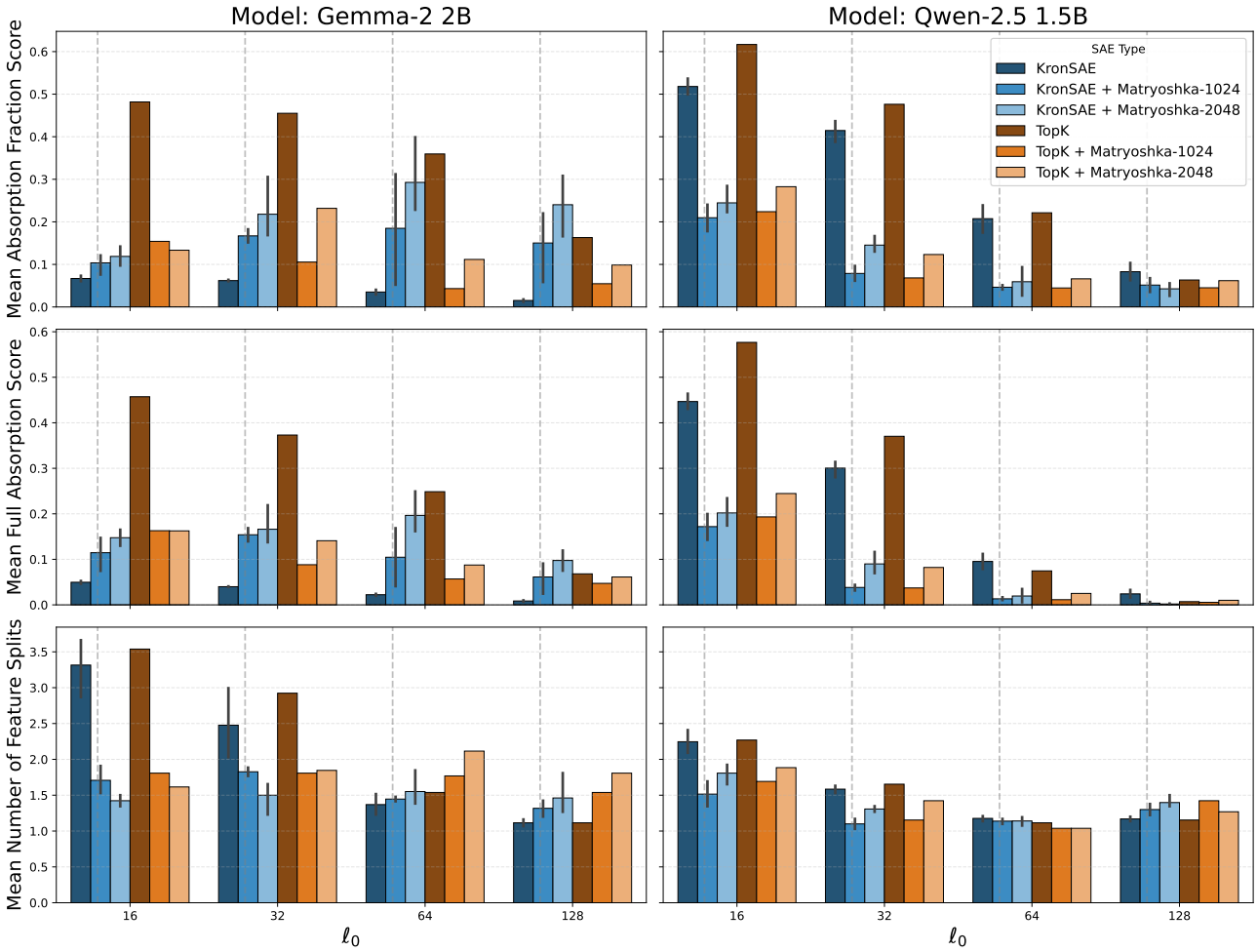

Figure 13. Feature absorption scores comparison of KronSAE vs baseline methods (TopK SAE and Matryoshka SAE) and a Kron-enhanced Matryoshka SAE variant.

B. mAND as a Logical Operator and KronSAE as Logical SAE

AND-like mechanism and co-activation hierarchy. Suppose we have $\mathbf{u}^k = P^k\mathbf{x}$ and $\mathbf{v}^k = Q^k\mathbf{x}$. KronSAE can be described in two equivalent ways:

1. Applying the ReLU to latents $\mathbf{u}$, $\mathbf{v}$, then applying the Kronecker product and square root.
2. Applying the mAND kernel that creates the matrix as in equation 3 and flattening it.

These approaches are complementary. To formally understand how they induce the AND-like mechanism, consider base and extension pre-latent vectors $\mathbf{p} = \mathrm{ReLU}(\mathbf{u})$ and $\mathbf{q} = \mathrm{ReLU}(\mathbf{v})$. Each p_i is a ith base pre-latent activation, and q_j is jth extension pre-latent activation. Kronecker product $\mathbf{p} \otimes \mathbf{q}$ creates the vector $(p_1 * q_1, \ldots, p_1 * q_m, p_2 * q_1, \ldots, p_n * q_m)$ of the activations of post-latents. Then there is two situations:

1. Post-latent is active $\implies$ both pre-latents activations are positive.
2. Post-latent is inactive $\implies$ at least one of pre-latent activations is zero.

Hence post-latent is active only when both pre-latents are active. Fix some post-latent and its corresponding pre-latents, and suppose that $\mathcal{P}, \mathcal{Q}, \mathcal{F}$ are the sets of input vectors from the hidden state space (passed to SAE) on which base pre-latent,

Table 4. Activation function ablation for KronSAE. **Left**: performance of SAEs with JumpReLU and TopK activations. Since we have floating sparsity controlled via l0 penalty coefficient, we performed a sweep over various sparsity levels, fitted a parabola to the resulting data as a function of ℓ_0, and evaluated it on those sparsity levels. In contrast, TopK and KronSAE (with TopK) were trained using fixed, predefined sparsity levels. **Right**: comparison of Gated SAE architecture with its KronSAE version for dictionary $F = 2^{16}$.

Model Variant	$\ell_0 = 32$	$\ell_0 = 50$	$\ell_0 = 64$
TopK	0.809	0.837	0.852
JumpReLU	0.813	0.838	0.844
KronSAE (TopK)	0.814	0.840	0.853
KronSAE (JumpReLU)	0.790	0.817	0.828

Variant	ℓ_0	EV ↑
KronSAE (m=1, n=32)	63	0.8319
Gated (run 1)	68	0.83050
KronSAE (m=1, n=16)	63	0.83027
KronSAE (m=2, n=8)	66	0.82060
Gated (run 2)	63	0.81300

extension pre-latent and post-latent are active. Then it is true that $\mathcal{F} = \mathcal{P} \cap \mathcal{Q}$, meaning that pre-latents must be broader and polysemous to encode multiple semantics of emitted post-latents. We validate this behaviour qualitatively in section 5.3, although in the same section we describe that apparently other types of interactions also presented.

Matryoshka and Kron hierarchies. In contrast to TopK SAE, Matryoshka loss imposes different kind of hierarchy by dividing the dictionary into groups of $G_1, \ldots, G_k$ latents where each is of different level of granularity. Namely, first G_1 latents are the most broad and abstract, next G_2 latents add more fine-grained semantics, lowering the level of abstraction, and so on. This type of structure imposed by specific loss function - increasing the level of granularity must decrease the reconstruction error, and the lowest level of G_1 features must also maintain good reconstruction quality - and does not strictly demand some kind of conditional activation of features, while KronSAE, instead of imposing a parent-child abstraction levels hierarchy, imposes a two-level AND-like hierarchy via architectural co-activation prior (Kronecker product). As we have two different mechanisms of feature hierarchy (from encoding mechanism and loss function design) we can combine it, as shown in Section 4.1, 4.2 and Appendix A.4, to combine properties of both approaches.

Visual intuition. We also compare our mAND to existing $\mathrm{AND}_{\mathrm{AIL}}$ (Lowe et al., 2021). Since our objective is to drive each atom toward a distinct, monosemantic feature, we found that tightening the logical conjunction encourages sharper feature separation. Moreover, by using the geometric mean ($\sqrt{p\,q}$) rather than a simple product or minimum, mAND preserves activation magnitudes and prevents post-latent activation to be exploded when both p, q are positive. A visual comparison of mAND and $\mathrm{AND}_{\mathrm{AIL}}$ appears in Figure 16.

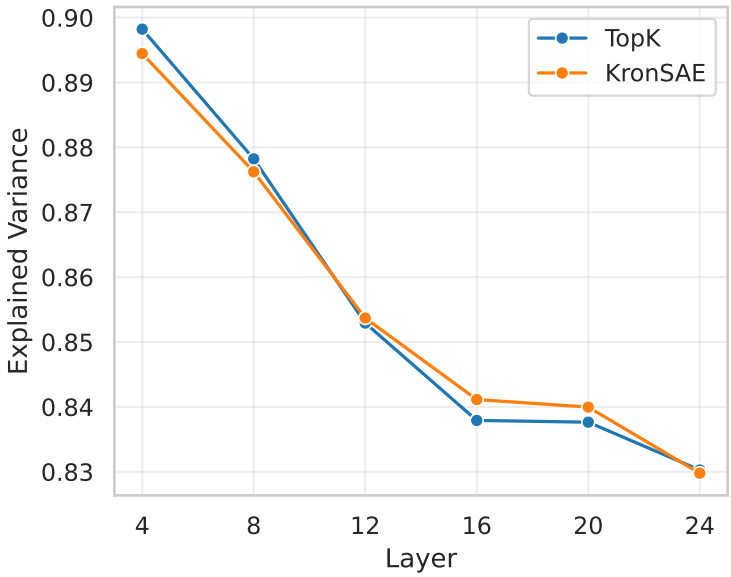

Figure 14. EV across layers of Qwen-2.5-1.5B, demonstrating that KronSAE matches TopK performance regardless of depth.

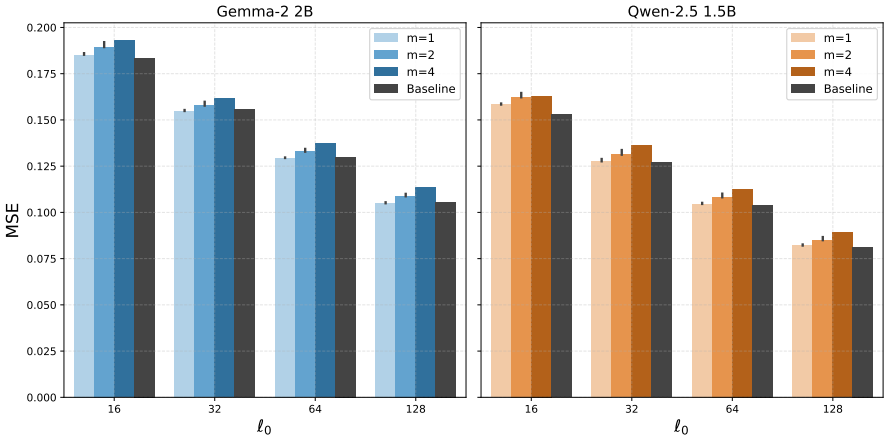

Figure 15. MSE comparision of KronSAE and TopK SAE under a fixed token budget.

Table 5. SAE performance on a Pythia suite. **Left**: results for Pythia-1.4B at 125M token budget. **Right**: SAE performance of on 70M, 160M and 410M Pythias with varying hidden dimensionality.

Dictionary	SAE	Mean	Max
32k	TopK		0.793
	KronSAE	0.783	0.793
65k	TopK		0.802
	KronSAE	0.795	0.805
131k	TopK		0.801
	KronSAE	0.800	0.810

Model	70M (d=512)		160M (d=768)		410M (d=1024)	
Dictionary	65k	256k	65k	256k	65k	256k
$m = 1$	0.893	0.892	0.856	0.855	0.834	0.835
$m = 2$	0.896	0.897	0.859	0.859	0.832	0.841
$m = 4$	0.894	0.899	0.857	0.859	0.828	0.839
$m = 8$	0.896	0.899	0.856	0.862	0.827	0.846
TopK	0.905	0.903	0.870	0.867	0.843	0.847

C. More Results on Synthetic

In this section we present the motivation behind our matching algorithm and additional results.

C.1. Feature Matching as Quadratic Problem

Suppose that we have two sets of feature embeddings from different models represented as matrices $X, Y \in \mathbb{R}^{F \times d}$, where F is the number of features and d is the dimensionality of feature embeddings. Our task is to find the optimal assignment between features from Y to features from X so that this assignment would satisfy some considerations.

The standard approach is to solve linear assignment problem - find a permutation matrix Π subject to minimizing the trace$(C^T \Pi)$, where C is a cost matrix defined as pairwise distance between features $C_{i,j} = d(X_i, Y_j)$. Standard algorithm for solving it in context of sparse dictionary learning is a Hungarian algorithm (Paulo & Belrose, 2025; Balagansky et al., 2025; Fel et al., 2025).

This linear problem only considers pairwise information between features while ignoring the global dependencies between features within X and Y sets separately, e.g. clusters should map to clusters, and the linear problem does not internalize this information. Our observation of correlations between features naturally requires to search for assignment that would take this information into account: we seek for a permutation Π that would give best *global* alignment, measured as the Frobenius norm of $X^T \Pi Y$. So the objective becomes:

$$\max_{\Pi} \|X^T \Pi Y\|_F^2 = \max_{\Pi} \text{trace}(\Pi^T X X^T \Pi Y Y^T), \tag{10}$$

where global feature structure is explicitly encoded in the matrices XX^T and YY^T.

An efficient algorithm to solve the quadratic assignment problem is Fast Approximate Quadratic Programming (FAQ) method (Vogelstein et al., 2015) that initially was designed for graphs matching: given the adjacency matrices A and B, it

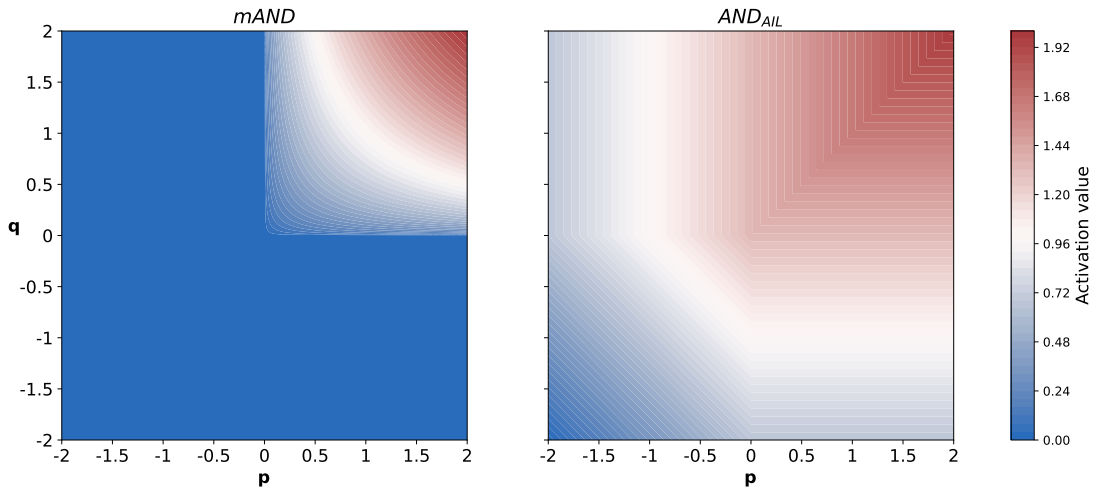

Figure 16. Comparison of the smooth mAND operator against the $\mathrm{AND}_{\mathrm{AIL}}$ (Lowe et al., 2021).

minimizes the trace$(A\Pi B^T\Pi^T)$, where Π are relaxed from permutation matrices to the set of doubly stochastic matrices (Birkhoff polytope). In our case we define $A = XX^T$ and $B = YY^T$, and since we do not want to minimize the cost but rather maximize the similarity, we solve for the reversed objective:

$$\max_{\Pi}(XX^T\Pi(YY)^T\Pi^T) = \max_{\Pi}\mathrm{trace}(\Pi^T XX^T\Pi YY^T), \tag{11}$$

which is the same as equation 10, and this formulation preserves global dependencies because the contribution of assigning Y_j to X_i depends on all other assignments through the cross-terms in the quadratic form. Listing 1 shows the implementation of this matching procedure.

Listing 1. Implementation of FAQ algorithm for quadratic feature assignment problem.

```
def feature_matching(A, B, max_iter):
    F, d = A.shape[0]
    G_A, G_B = A @ A.T, B @ B.T
    P = np.ones((F, F)) / F # Barycenter initialization

    # Frank-Wolfe iterations
    for _ in range(max_iter):
        grad = 2 * G_A @ P @ G_B  # Gradient for maximization
        r, c = linear_sum_assignment(-grad)  # Solve LAP
        Q = np.zeros_like(P)
        Q[r, c] = 1

        # Compute optimal step size
        D = Q - P
        b, a = np.trace(grad.T @ D), np.trace(G_A @ D @ G_B @ D.T)

        if abs(a) < 1e-12:
            alpha = 1.0 if b > 0 else 0.0
        elif a < 0:  # Concave case
            alpha = np.clip(-b/(2*a), 0, 1)
        else:        # Convex case
            alpha = 1.0 if b > 0 else 0.0

        P_new = P + alpha * D
        P = P_new

    # Project to permutation matrix
    r, c = linear_sum_assignment(-P)
    P_final = np.zeros((F, F))
    P_final[r, c] = 1

    return P_final, P_final @ B
```

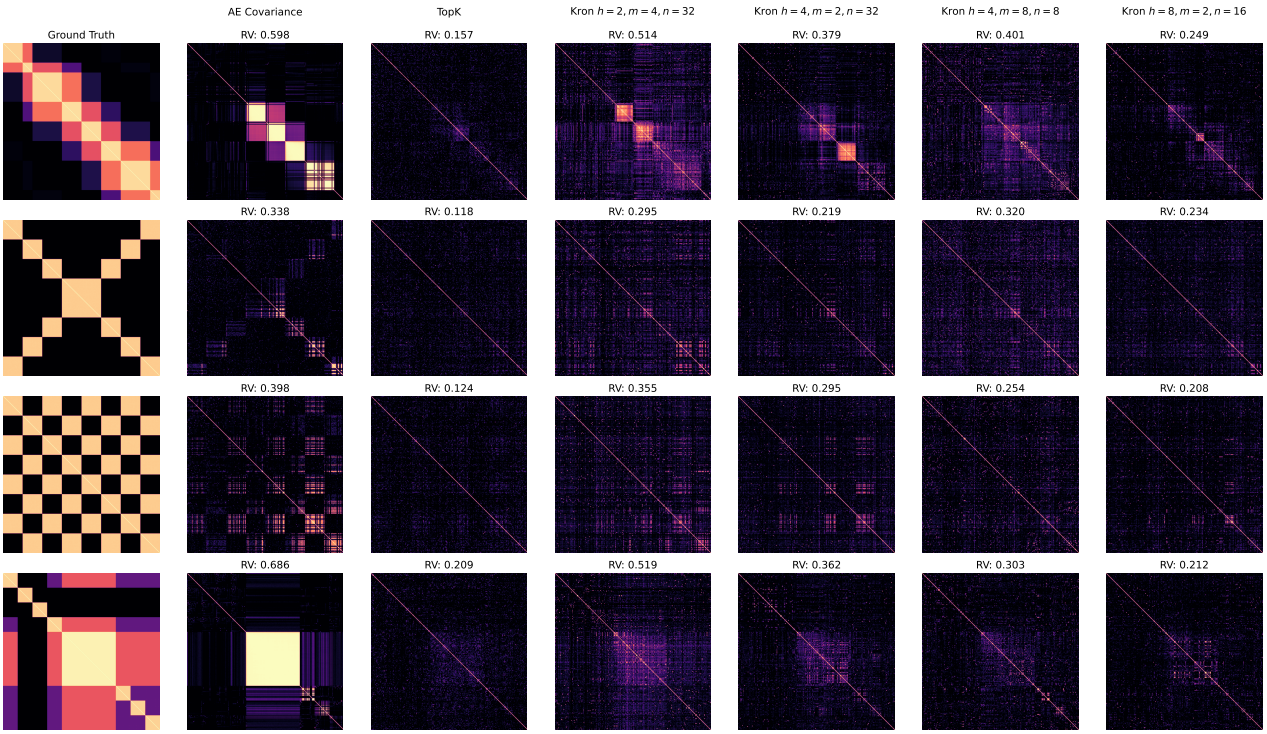

Figure 17. Examples of patterns learned in autoencoder TopK and KronSAE variants after we apply the improved matching scheme. KronSAE have learned patterns that more closely resemble the underlying ground truth structure, and with increasing number of heads (more fine-grained structure) it approaches the TopK SAE.

C.2. Additional Results

As shown in Figure 17, KronSAE maintains correlation structure that is heavier than in TopK and is better aligned with ground truth covariance matrix. Table 6 indicate that structure learned by KronSAE is more diverse as indicated by standard deviation of the corresponding values, and is also more aligned with the ground truth covariance according to higher RV scores and lower effective rank difference Δ between properties of ground truth and learned matrix.

D. Feature Analysis Methodology

We analyze learned features using an established pipeline (Bills et al., 2023; Paulo et al., 2024) consisting of three stages: (1) statistical property collection, (2) automatic activation pattern interpretation, and (3) interpretation evaluation.

Table 6. Results with improved matching scheme, computed for 8 different covariance setups. Higher RV coefficients between ground truth and learned matrices and lower effective rank difference Δ between properties of these matrices indicate that KronSAE is more variable across different setups and is better aligned with ground truth correlation structures.

Model	RV coefficient	Effective rank	Mean correlation	Rank Δ
AE	0.59 ± 0.17	63.3 ± 0.3	0.13 ± 0.04	47.9
TopK	0.12 ± 0.02	58.5 ± 1.2	0.08 ± 0.01	43.2
Kron $h = 2, m = 4, n = 32$	0.22 ± 0.07	53.8 ± 3.4	0.08 ± 0.01	38.5
Kron $h = 4, m = 2, n = 32$	0.3 ± 0.12	46.0 ± 7.8	0.10 ± 0.03	30.7
Kron $h = 4, m = 8, n = 8$	0.23 ± 0.08	48.8 ± 5.0	0.09 ± 0.02	33.5
Kron $h = 8, m = 2, n = 16$	0.17 ± 0.04	55.9 ± 1.7	0.07 ± 0.01	40.6

D.1. Data collection

Our collection process uses a fixed-size buffer $B = 384$ per feature, continuing until processing a predetermined maximum token count T_{max}. The procedure operates as follows:

Initial processing batches generate large activation packs of 1M examples, where each example comprises 256-token text segments. When encountering feature activations, we add them to the buffer, applying random downsampling to maintain size B when exceeding capacity. This approach enables processing arbitrary token volumes while handling rare features that may require extensive sampling.

During collection, we compute online statistics including activation minimums, maximums, means, and frequencies. Post-processing yields two key metrics: token entropy and multitoken ratio. The token entropy is calculated as:

$$\text{token entropy} = -\sum_{i=0}^{n} p_i \cdot \log(p_i), \quad p_i = \frac{\text{activations of token i}}{\text{total amount of activations}}, \tag{12}$$

where n represents unique activated tokens. The multitoken ratio is:

$$\text{multitoken ratio} = \frac{1}{b}\sum_{i=0}^{b} \frac{\text{number of activations in sequence } i}{\text{total tokens in sequence } i}, \tag{13}$$

with $b < B$ denoting collected context examples per feature.

We then segment examples using a 31-token context window (15 tokens before/after each activation), potentially creating overlapping but non-duplicated examples. Features with high multitoken ratio may have number of examples significantly exceeding B.

A separate negative examples buffer captures non-activating contexts. Future enhancements could employ predictive modeling (e.g., using frequent active tokens) to strategically populate this buffer with expected-but-inactive contexts, potentially improving interpretation quality.

D.2. Feature interpretations

For each feature, we generate interpretations by sampling 16 random activation examples above the median activation quantile and presenting them to Qwen3 14B (Yang et al., 2025) (AWQ-quantized with reasoning enabled). The model produces concise descriptions of the activation patterns. Empirical observations suggest reasoning mode improves interpretation quality, though we lack quantitative measurements. This aligns with findings in (Paulo et al., 2024), which compared standard responses with Chain-of-Thought outputs, making model reasoning an interesting direction for future research.

The interpretation process uses the system prompt presented in a Figure 18. User prompts include all special characters verbatim, as some features activate specifically on these characters. A representative (slightly abbreviated) user prompt example is presented on Figure 19.

D.3. Evaluation pipeline

We evaluate interpretations using balanced sets of up to 64 positive (activation quantile ¿ 0.5) and 64 negative examples, employing the same model without reasoning to reduce computational costs. When insufficient examples exist, we maintain class balance by equalizing positive and negative counts. The evaluation uses modified system prompts from (Paulo et al., 2024), with added emphasis on returning Python lists matching the input example count exactly. We discard entire batches if responses are unparseable or contain fewer labels than the number of provided examples. We calculate two scores.

Detection Score: After shuffling positive/negative examples, we present up to 8 unformatted text examples per batch to the model. The model predicts activations (1/0) for each example, generating up to 128 true/predicted label pairs, and calculate:

$$\text{score} = \frac{1}{2}\left(\frac{\text{correctly predicted positives}}{\text{total positives}} + \frac{\text{correctly predicted negatives}}{\text{total negatives}}\right). \tag{14}$$

Fuzzing Score: We ≪highlight≫ activated tokens on sampled examples, from which 50% are correctly labeled positive examples, 25% are mislabeled positive examples, and 25% are randomly labeled negative examples. We present batches of up to 8 examples and the model identifies correct/incorrect labeling, with scoring following Equation 14.

```
You are a meticulous AI researcher conducting an important investigation
into patterns found in language.  Your task is to analyze text and provide
an explanation that thoroughly encapsulates possible patterns found in it.

Guidelines:

You will be given a list of text examples on which special words are
selected and between delimiters like <<this>>.  If a sequence of consecutive
tokens all are important, the entire sequence of tokens will be contained
between delimiters <<just like this>>.  How important each token is for the
behavior is listed after each example in parentheses.

- Your explanation should be a concise STANDALONE PHRASE that describes
observed patterns.
- Focus on the essence of what patterns, concepts and contexts are present
in the examples.
- Do NOT mention the texts, examples, activations or the feature itself in
your explanation.
- Do NOT write "these texts", "feature detects", "the patterns suggest",
"activates" or something like that.
- Do not write what the feature does, e.g.  instead of "detects heart
diseases in medical reports" write "heart diseases in medical reports".
- Write explanation in the last line exactly after the [EXPLANATION]:
```

Figure 18. System prompt for feature interpretations.

E. Additional Feature Analysis Results

Feature property correlations. Our analysis reveals significant correlations between feature properties and interpretability scores (Figure 20). Notably, token entropy and mean activation show substantial correlations with interpretability scores, suggesting their potential as proxies for assessing feature quality without running the full interpretation pipeline. These findings are based on analysis of the first 3072 features from 32k TopK and KronSAE (m=4, n=4) trained on 24M tokens, warranting further validation with larger-scale studies.

Pre-latent to post-latent relationships. We investigate how post-latent properties correlate with various combinations of pre-latent properties, including individual values, means, products, and the mAND operation (product followed by square root). Figure 21 demonstrates that post-latent multitoken ratio, token entropy, and frequency show stronger correlations with pre-latent products or mAND values than with individual pre-latent properties or their means.

Basis geometry. As noted in Section 5.3, latent embeddings primarily exhibit clustering within their originating groups (head, base, extension). With the support of observations reported in Sections 4.1 and 5.3, we find that models with more heads achieve better reconstruction while producing more diverse basis vectors. This suggests that fine-grained architectures yield more expressive representations, although they may also exhibit undesired challenging behavior like feature splitting (Bricken et al., 2023) or absorption (Chanin et al., 2024).

Figure 22 visualizes this structure through UMAP projections (n_neighbors=15, min_dist=0.05, metric='cosine') of decoder weights from the first 8 heads of 32k SAEs with varying m,n configurations. The plots reveal distinct clustering patterns: for $m < n$ we observe tight base-wise clustering with weaker grouping by extension, and for $m \geq n$ extension-wise clustering is stronger.

This asymmetry suggests that pre-latent capacity requirements directly manifest in the embedding geometry - components with lower polysemanticity (extensions when m ¡ n) exhibit greater geometric diversity. We expect symmetric behavior for reciprocal configurations (e.g., m=4,n=8 vs. m=8,n=4), merely swapping the roles of bases and extensions.

```
Examples of activations:

Text:  '  Leno<<,>> a San Francisco Democrat<<, said in a statement.>>'
Activations:  '  said (22.74), statement (27.84), in (27.54)'

Text:  '  city spokesman Tyler Gamble<< said in an>> email.'
Activations:  '  said (2.92), in (12.81), an (14.91)'

Text:  '  towpath at Brentford Lock.  <<Speaking>> on BBC London 94'
Activations:  'Speaking (3.48)'

Text:  '  Michelle, a quadriplegic,<< told>> DrBicuspid.com'
Activations:  '  told (4.05)'

Text:  '  CEO Yves Carcelle<< said in a statement>>.'
Activations:  '  said (19.64), in (29.09), statement (29.39)'
```

Figure 19. Example of prompt passed to LLM. This feature received the interpretation "*Structural elements in discourse, including speech attribution, prepositional phrases, and formal contextual markers*" with a detection and fuzzing scores of 0.84 and 0.76.

Interpretability across sparsity regimes. We compare KronSAE and its Matryoshka variant with TopK, Matryoshka and Switch SAE baselines across different sparsity regimes for Gemma 2 2B and Qwen 2.5 1.5B. For each SAE we follow the same pipeline as for results in Section 5.3, but use only 18 million tokens for examples collection. For KronSAE and TopK we use first 4096 features, and for other models we sample 4096 features randomly. Scores for $F = 2^{16}$ and $F = 2^{15}$ are presented in Figures 23 and 24.

Results show that TopK and KronSAE are very stable across both models and sparsity regimes, while for other models their architectural and training design significantly affect the interpretability.

Interpretability tradeoff. As shown in Section 4.1 KronSAE introduces tradeoff between computational efficiency, explained variance and interpretability. We can force more features to be correlated in same head by increasing the m and n, and this will improve computational efficiency, but at the cost of reconstruction performance; however, improving the explained variance (with small m and fine-grained structure of groups) goes with the cost of slight reduction in the interpretability. For absorption, lower scores are better, and we report them alongside interpretability to make the trade-off explicit. These tradeoffs are expected and presented across variety of SAEs (Karvonen et al., 2025).

To evaluate how KronSAE behave under different m, n we compute autointerpretability scores following the same setup as described in Appendix D for Qwen-2.5 1.5B and Gemma-2 2B models with sparsity $\ell_0 \in \{16, 32, 64, 128\}$. As Figure 25 shows, setup $m = 1$ and $h = 4096$ is less interpretable than $m = 2$ and $m = 4$ with the same number of heads despite having stronger reconstruction performance.

F. KronSAE in Terms of Tensor Diagram

The proposed encoder architecture can be visualized as a tensor diagram (Figure 26). Notably, this formulation draws a connection to quantum mechanics, where $|\mathbf{f}\rangle$ represents the (unnormalized) state of two disentangled qubits described by $|\boldsymbol{p}\rangle$ and $|\boldsymbol{q}\rangle$.

If we were to sum the outputs of the encoder's heads instead of concatenating them, $|\mathbf{f}\rangle$ would correspond to a separable quantum state. This scenario can be expressed via the Schmidt decomposition:

$$|\mathbf{f}\rangle = \sum_h |\boldsymbol{p}_h\rangle \otimes_K |\boldsymbol{q}_h\rangle \,,$$

where $\otimes_K$ denotes the Kronecker product. However, preliminary experiments revealed that this alternative design results in poorer performance compared to the concatenation-based approach.

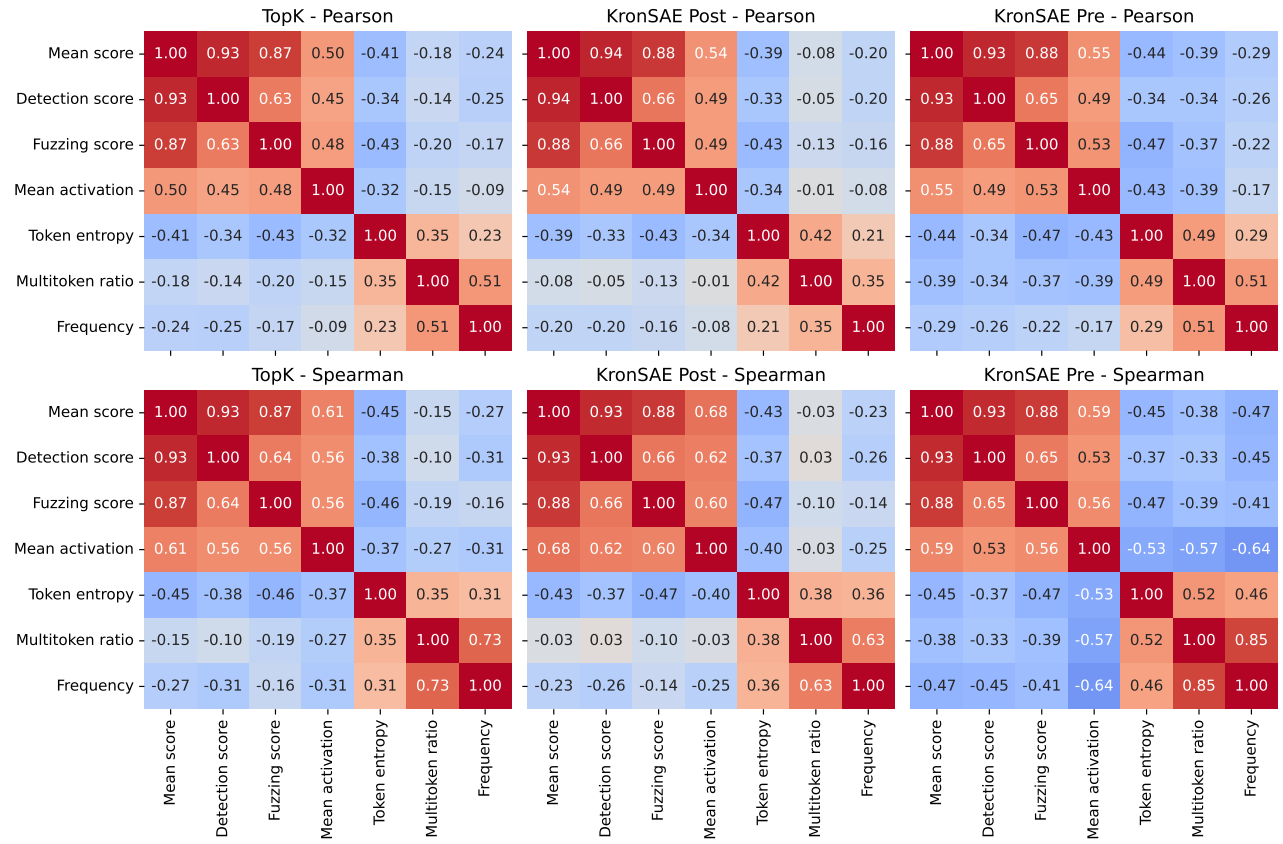

Figure 20. Correlation coefficients (Pearson and Spearman) between properties of TopK and KronSAE latents. Token entropy emerges as a strong predictor of interpretability scores, while higher mean activation and lower frequency also indicate more interpretable features.

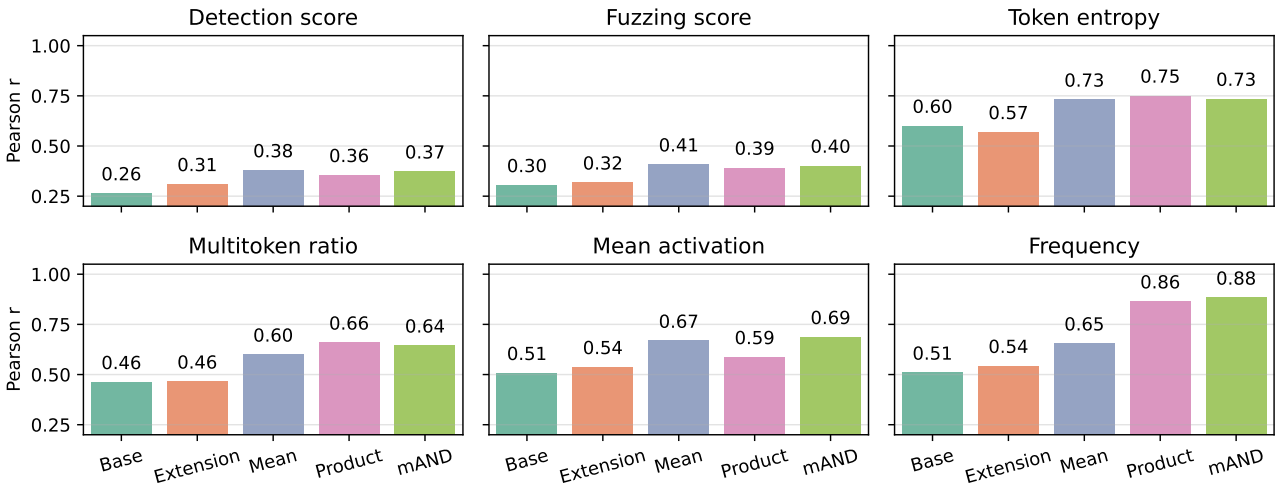

Figure 21. Correlation patterns between properties of post-latents and pre-latents.

G. Analysis of Compositional Structure

Head 3. For this head we have selected all base elements and extension 2, shown in Table 7. Extension element 2 shows moderate interpretability with clear AND-like interactions: with base 1 (semantic inheritance through shared pre-latent semantics) and base 2 (retaining only instrument-related semantics). Notable interactions occur with base 0 (acquiring

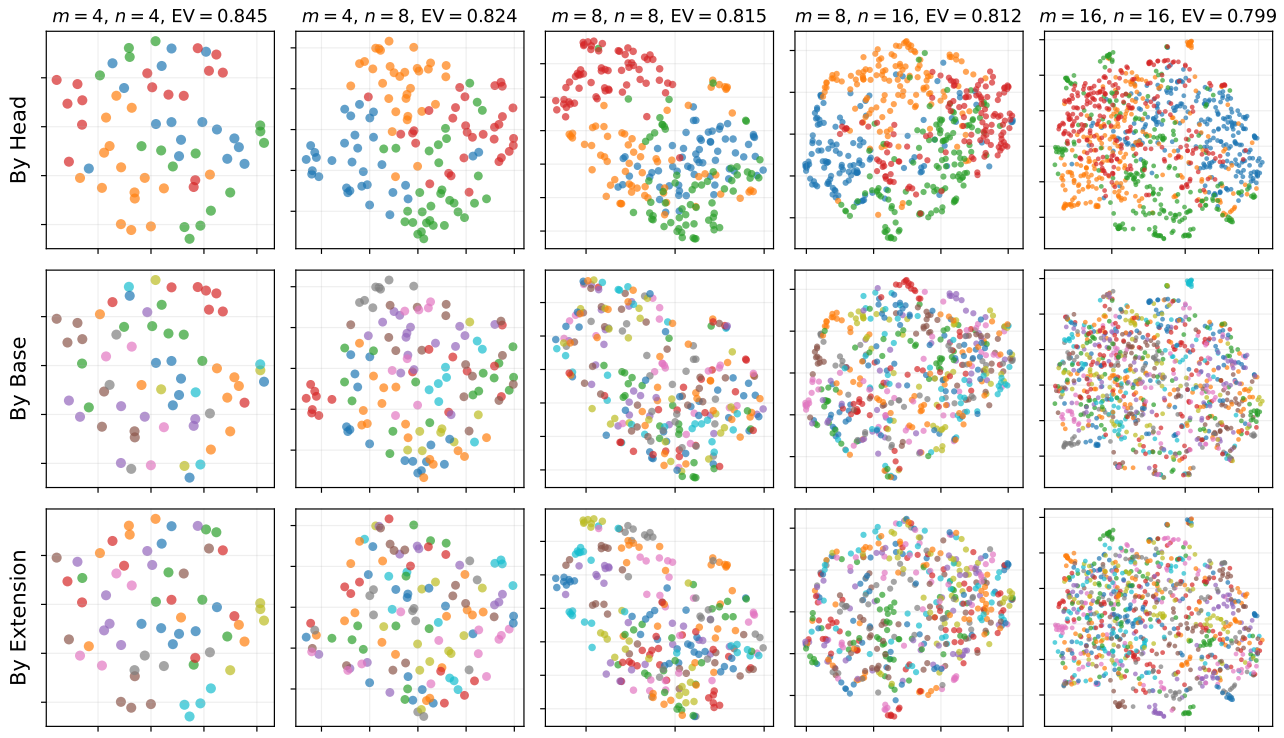

Figure 22. UMAP visualization of post-latent clustering patterns by head, base, and extension group membership. We observe tight clusters by base for $m < n$ and by extension for $m \geq n$.

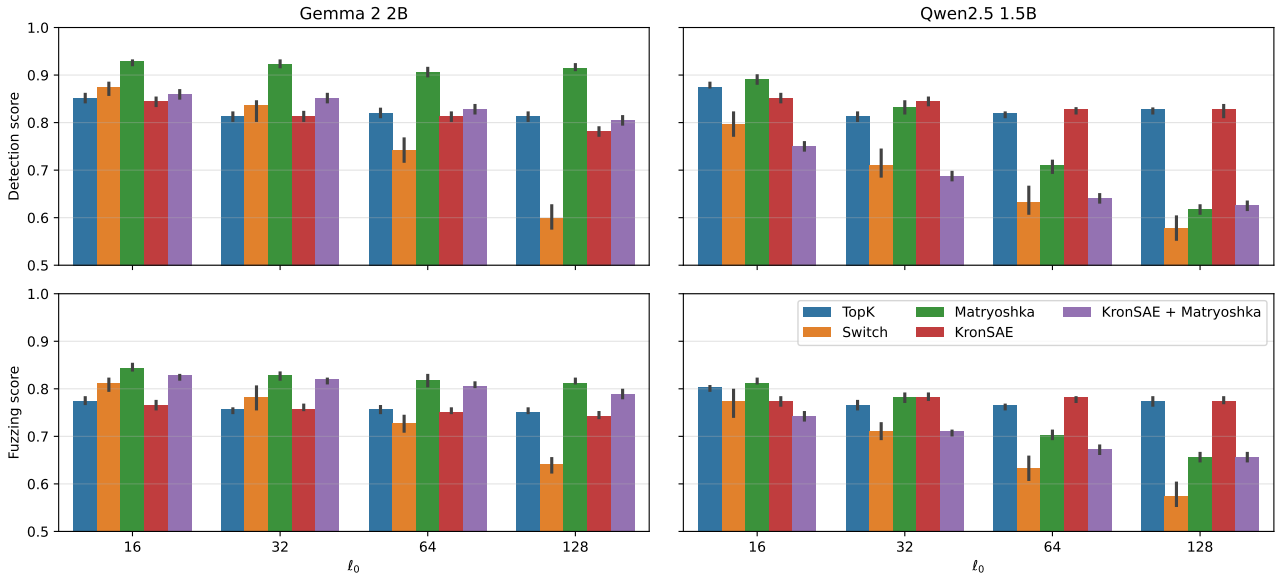

Figure 23. Interpretability scores for $F = 2^{16} = 65536$. KronSAE shows good consistency across different sparsity regimes, in some cases outperforming the TopK baseline.

medical semantics while preserving metric/number aspects) and base 3 (combining instrument semantics with necessity to yield therapy/treatment concepts). The high interpretability scores suggest potential additional encoding mechanisms beyond simple intersection, possibly related to activation magnitude, though dataset or interpretation artifacts cannot be ruled out without further validation.

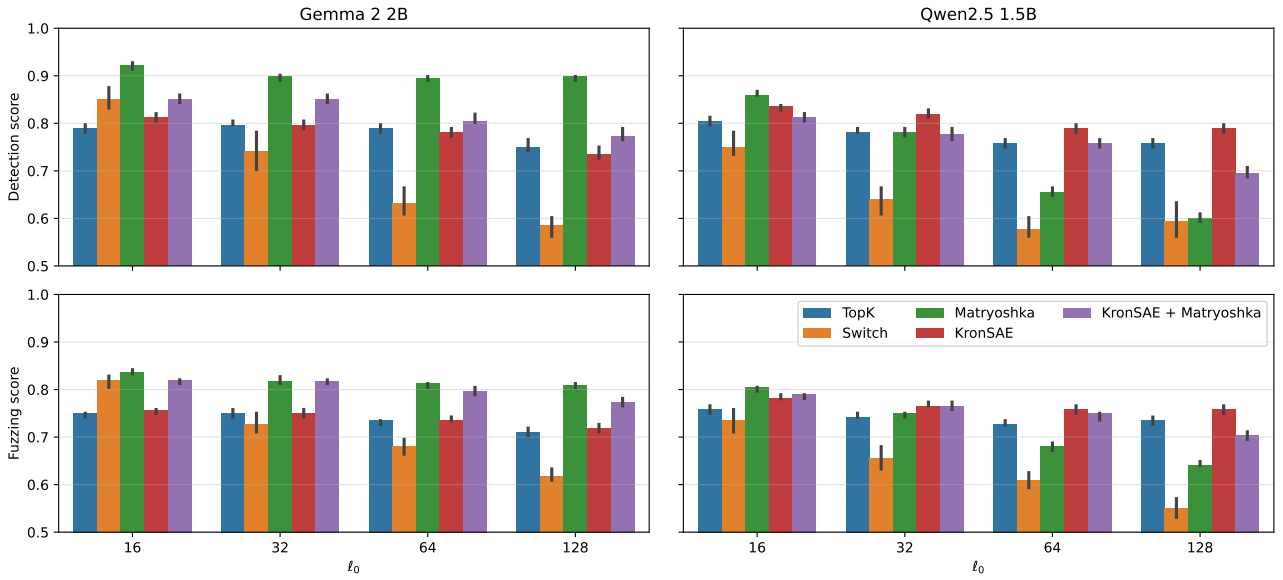

Figure 24. Interpretability scores for $F = 2^{15} = 32768$. KronSAE shows good consistency across different sparsity regimes, in some cases outperforming the TopK baseline. The result is the same as for 65k dictionary size at the Figure 23, but with slightly lower scores.

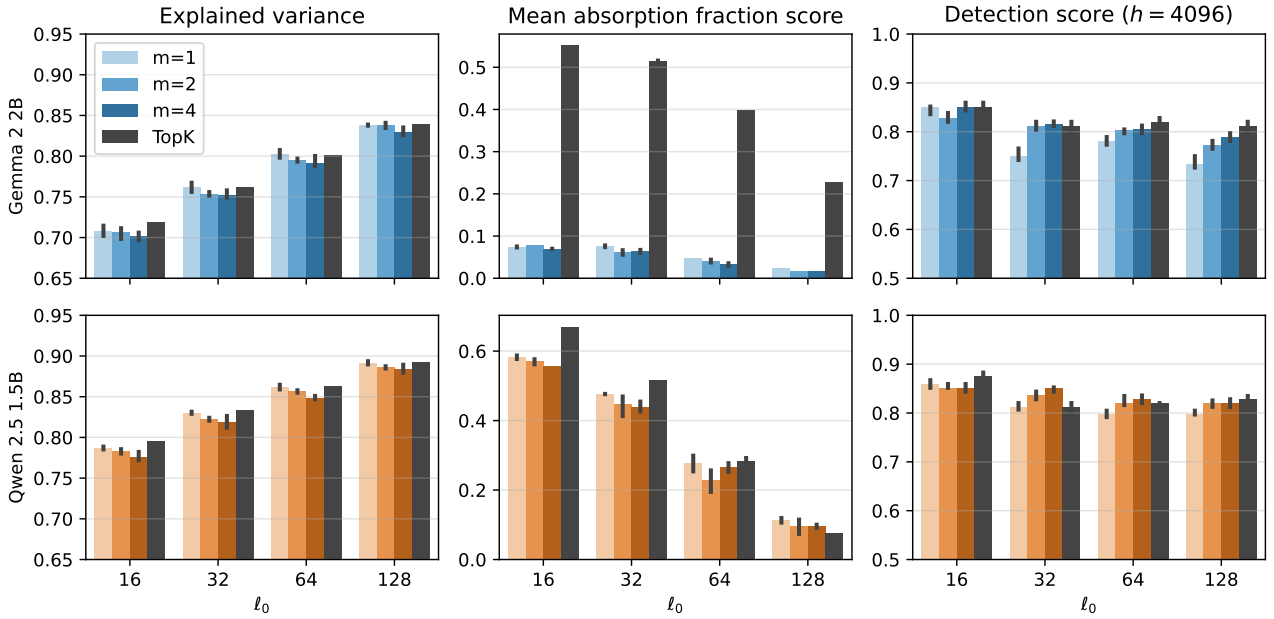

Figure 25. EV, absorption score and interpretability for $F = 2^{16}$. For constant h, decreasing m leads to lower interepretability scores and absorption scores, but increase EV - this is expected due to increasing polysemanticity of pre-latents and entanglement of the features. See Section 5.3 for a explanation of the retrieval mechanism.

Head 136. This head exhibits higher interpretability in post-latents than pre-latents. Key observations from the Table 8 include: extension 2 with base 0 narrows semantics to Illinois (likely inheriting geographical subsemantics), while interactions with bases 2-3 demonstrate complexity beyond simple intersection, often introducing additional semantics requiring deeper investigation.

Head 177. Latents presented in Table 9 emonstrates more consistent AND-like behavior than Heads 3 and 136, closely matching the interaction pattern shown in Figure 2.

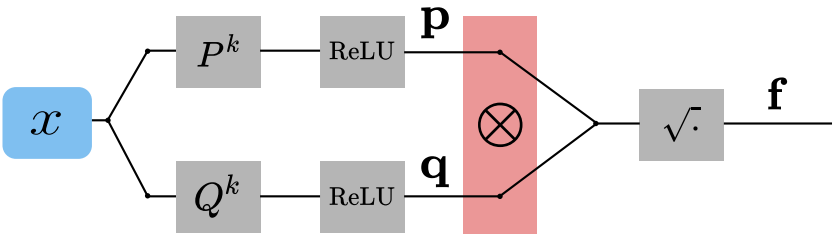

Figure 26. For a single head, the KronSAE encoder architecture separates the input $\boldsymbol{x}$ into two distinct components, $\boldsymbol{p}$ and $\boldsymbol{q}$, via matrix multiplications P^k and Q^k accordingly, followed by ReLU activation. These components are then combined via the Kronecker product $\boldsymbol{p} \otimes \boldsymbol{q}$ and square root operation $\sqrt{\cdot}$, resulting in an output vector $\mathbf{f}$

.

Table 7. Interactions between extension 2 in head 3 and all base elements in that head.

Component	Interpretation	Score
Extension 2	Scientific instruments, acronyms, and critical numerical values in technical and astronomical contexts	0.71
Base elements and their compositions with extension 2		
Base 0	*Interpretation:* Punctuation marks and line breaks serving as structural separators in text.	0.66
	Composition: Health-related metrics focusing on survival rates, life expectancy, and longevity.	0.88
Base 1	*Interpretation:* Numerical values, both in digit form and as spelled-out numbers, often accompanied by punctuation like decimals or commas, in contexts of measurements, statistics, or quantitative expressions.	0.80
	Composition: Numerical digits and decimal points within quantitative values.	0.86
Base 2	*Interpretation:* Nuanced actions and adverbial emphasis in descriptive contexts.	0.71
	Composition: Astronomical instruments and their components, such as space telescopes and their acronyms, in scientific and observational contexts.	0.90
Base 3	*Interpretation:* Forms of the verb ”to have” indicating possession, necessity, or occurrence in diverse contexts.	0.91
	Composition: Antiretroviral therapy components, viral infection terms, and medical treatment terminology.	0.87

Table 8. Interactions between extension 2 in head 136 and all base elements in that head.

Component	Interpretation	Score
Extension 2	Hierarchical scopes, geographic references, and spatial dispersal terms	0.78
Base elements and their compositions with extension 2		
Base 0	*Interpretation:* Numerical decimal digits in quantitative expressions and proper nouns.	0.79
	Composition: The state of Illinois in diverse contexts with high significance.	0.95
Base 1	*Interpretation:* The number three and its various representations, including digits, Roman numerals, and related linguistic forms.	0.84
	Composition: Geographic place names and their linguistic variations in textual contexts.	0.91
Base 2	*Interpretation:* Ordinal suffixes and temporal markers in historical or chronological contexts.	0.87
	Composition: Terms indicating layers, degrees, or contexts of existence or operation across scientific, organizational, and conceptual domains.	0.82
Base 3	*Interpretation:* Question formats and topic introductions with specific terms like ”What”, ”is”, ”of”, ”the”, ”Types”, ”About” in structured text segments.	0.77
	Composition: Spatial spread and occurrence of species or phenomena across environments.	0.87

Table 9. Interactions between extension 1 in head 177 and all base elements in that head.

Component	Interpretation	Score
Extension 1	Geographical mapping terminology and institutional names, phrases involving spatial representation and academic/organizational contexts	0.90
Base elements and their compositions with extension 1		
Base 0	*Interpretation:* Proper nouns, abbreviations, and specific named entities.	0.64
	Composition: Geographical or spatial references using the term ”map”.	0.93
Base 1	*Interpretation:* Emphasis on terms indicating feasibility and organizations.	0.80
	Composition: Specific organizations and societal contexts.	0.89
Base 2	*Interpretation:* Institutional names and academic organizations, particularly those containing ”Institute” or its abbreviations, often paired with prepositions like ”of” or ”for” to denote specialization or affiliation.	0.89
	Composition: Institutional names containing ”Institute” as a core term, often followed by prepositions or additional descriptors.	0.92
Base 3	*Interpretation:* Closure and termination processes, initiating actions.	0.79
	Composition: Initiating or establishing a state, direction, or foundation through action.	0.85

H. KronSAE Simplified Implementation

```python
class KronSAE(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.config = config
        _t = torch.nn.init.normal_(
                torch.empty(
                    self.config.act_size,
                    self.config.h * (self.config.m + self.config.n)
                )
            ) / math.sqrt(self.config.dict_size * 2.0)
        self.W_enc = nn.Parameter(_t)
        self.b_enc = nn.Parameter(
            torch.zeros(self.config.h * (self.config.m + self.config.n))
        )
        W_dec_v0 = einops.rearrange( # Initialize decoder weights
            _t.t().clone(), "(h mn) d -> h mn d",
            h=self.config.h, mn=self.config.m + self.config.n
        )[:, :self.config.m]
        W_dec_v1 = einops.rearrange(
            _t.t().clone(), "(h mn) d -> h mn d",
            h=self.config.h, mn=self.config.m + self.config.n
        )[:, self.config.m:]
        self.W_dec = nn.Parameter(einops.rearrange(
            W_dec_v0[..., None, :] + W_dec_v1[..., None, :, :],
            "h m n d -> (h m n) d"
        ))
        self.W_dec.data[:] = (
            self.W_dec.data / self.W_dec.data.norm(dim=-1, keepdim=True)
        )
        self.b_dec = nn.Parameter(torch.zeros(self.config.act_size))

    def encode(self, x: torch.Tensor) -> torch.Tensor:
        B, D = x.shape
        acts = F.relu(
            x @ self.W_enc + self.b_enc
        ).view(B, self.h, self.m + self.n)
        all_scores = torch.sqrt(
            acts[..., :self.config.m, None] * \
            acts[..., self.config.m:, None, :] + 1e-5
        ).view(B, -1)
        scores, indices = all_scores.topk(
            self.config.k, dim=-1, sorted=False
        )
        acts_topk = torch.zeros(
            (B, self.config.dict_size)
        ).scatter(-1, indices, scores)
        return acts_topk

    def forward(self, x):
        acts_topk = self.encode(x)
        x_rec = acts_topk @ self.W_dec + self.b_dec
        output = self.get_loss_dict(x, x_rec)
        return output

    def get_loss_dict(self, x, x_rec):
        loss = (x_rec - x.pow(2).mean()
        pt_l2 = (x_rec - x).pow(2).sum(-1).squeeze()
        var = (x - x.mean(0)).pow(2).sum(-1).squeeze()
        ev = (1 - pt_l2 / var).mean()
        return loss, ev
```