

Kronecker Factorization Improves Efficiency and Interpretability of Sparse Autoencoders

Vadim Kurochkin, Yaroslav Aksenov*,
Daniil Laptev, Daniil Gavrillov, Nikita Balagansky

T-Tech

*Correspondence: Yaroslav Aksenov, y.o.aksenov@t-tech.dev

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**, a design 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. KronSAE matches strong baselines on the EV-FLOPs frontier, improves interpretability of the latents, better captures the underlying correlated feature structure, and reduces encoder computational cost as an additional benefit. Code is available at <https://github.com/corl-team/kronsaes>.

1 Introduction

Interpreting the hidden states of large language models remains a central challenge for transparency and controllability in AI systems (Bricken et al., 2023; Heup et al., 2026). 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 (Bricken 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

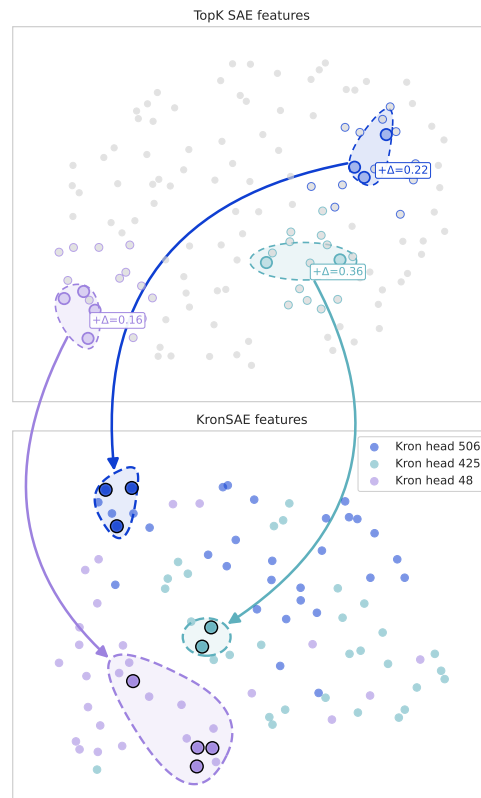


Figure 1: UMAP projection illustrating the relationship between TopK SAE features and KronSAE heads. Correlated TopK features tend to map to nearby KronSAE features within the same head. Selected features correlate more strongly than their nearest neighbors; Δ denotes the difference between correlations.

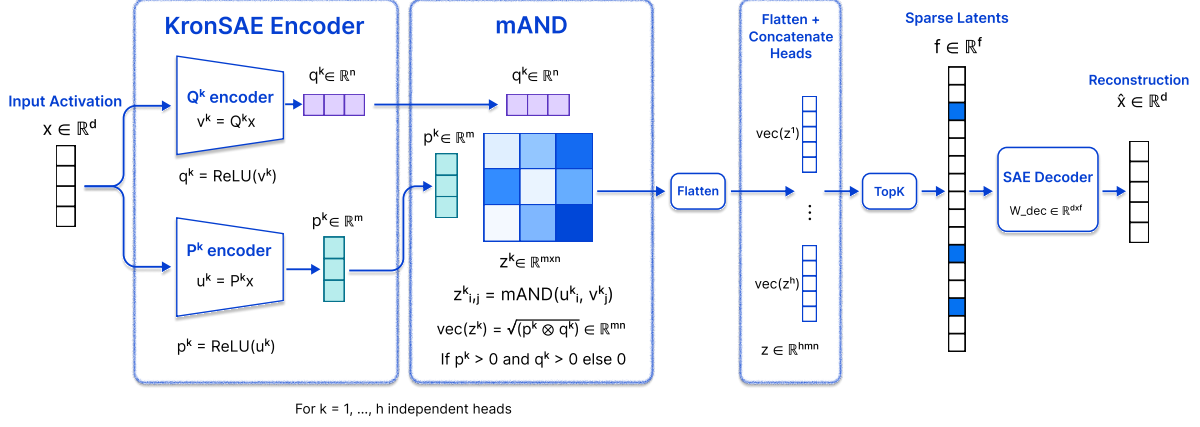


Figure 2: Overview of KronSAE. We project $x \in \mathbb{R}^d$ into head-specific base and extension pre-latents, then combine each pair with mAND to produce sparse post-latents.

(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), yielding a more structured latent space and improved interpretability results. However, a complementary direction remains underexplored: imposing hierarchy by changing the default structure of the encoding process itself. This can provide more flexible feature modeling and make structured feature interactions easier to learn.

We propose KronSAE, an encoder-side hierarchical modification to existing SAEs. KronSAE replaces a dense projection to F independent latents with projections to lower-dimensional pre-latents grouped into heads, and forms post-latent features through pairwise mAND interactions. This gives each post-latent an explicit compositional structure: it activates only when both of its parent pre-latents are active, creating an architectural prior toward co-activation and feature interaction. Figure 1 gives an example of this structure. Correlated TopK SAE features tend to map to nearby KronSAE features within the same head, suggesting that some feature relationships are better described as structured co-activation patterns

than as independent flat coordinates. This motivates an encoder architecture that makes such shared structure explicit rather than leaving it to emerge only implicitly during training.

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. Our main comparisons use an **equal token budget**, and report FLOP and parameter savings as consequences of the hierarchical encoder.

This work makes four primary contributions:

1. We identify encoder-side hierarchy as a missing design axis for sparse autoencoders, motivated by structural inductive biases in feature organization in SAEs.
2. We introduce **KronSAE**, a Kronecker-factorized SAE encoder modification equipped with the mAND interaction that forms post-latent features from reusable pre-latent factors.
3. We show that KronSAE improves SAE interpretability by decreasing feature absorption and yielding more specific latents.
4. We demonstrate an effective reconstruction–compute trade-off: reducing encoder FLOPs by more than **40%** incurs less than a 1% EV drop.

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 $\mathcal{O}(Fd)$ 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) argue that classical SAE encoders are not expressive enough and that more complex encoders can yield more faithful results without compromising interpretability.

Conditional Computation. These schemes use mixture-of-experts routing (Shazeer et al., 2017) to avoid instantiating a full dictionary per token, sending each input to a subset of expert SAEs through a lightweight gating network (Mudide et al., 2025). However, they still incur a dense encoder projection within each selected expert.

Factorizations and Logical Activation Functions. Tensor product representations (Smolensky, 1990) have been used to represent compositional structure in dense embeddings, closely resembling our idea of AND-like compositions. Recent studies have extended tensor product representations to 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 have been introduced to approximate Boolean operators smoothly (Lowe et al., 2022). Our method synthesizes these lines of work: we embed a differentiable AND-like gate into an efficient Kronecker-factorized 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

$$\begin{aligned} \mathbf{f} &= \text{TopK}(W_{\text{enc}}\mathbf{x} + \mathbf{b}_{\text{enc}}), \\ \hat{\mathbf{x}} &= W_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{dec}}, \end{aligned} \quad (1)$$

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 therefore scales as $\mathcal{O}(Fd)$ per token.

KronSAE. 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 $m \leq n \ll d$ and $F = hmn$. The *pre-latents*

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

where $\mathbf{u}^k = P^k\mathbf{x}$ and $\mathbf{v}^k = Q^k\mathbf{x}$. These pre-latents are combined through an element-wise interaction kernel independently in each head:

$$\begin{aligned} z_{i,j}^k &:= \text{mAND}(u_i^k, v_j^k) := \sqrt{u_i^k v_j^k} \\ &\text{if } u_i^k > 0 \text{ and } v_j^k > 0, \text{ otherwise } 0, \end{aligned} \quad (3)$$

where $\mathbf{z}^k \in \mathbb{R}^{m \times n}$. It is then flattened in 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 $\text{vec}(\cdot)$ uses row-major order,

$$\text{vec}(\mathbf{z}^k) = \sqrt{\text{vec}(\mathbf{p}^k(\mathbf{q}^k)^\top)} = \sqrt{\mathbf{p}^k \otimes \mathbf{q}^k}. \quad (4)$$

Concatenating heads and applying TopK yields *post-latents* $\mathbf{f} \in \mathbb{R}^F$. Activation events follow an AND-like rule: before TopK selection, a post-latent has support only where both parents are active; global TopK selection further restricts the final support. This architectural co-activation prior induces specific semantic structure, as we empirically verify in Section 6.2. The square root in Equation 3 keeps output magnitudes comparable. In practice, we add a small ϵ before taking the square root for numerical stability. Although this introduces slight noise into the exact AND rule, its effect is negligible after TopK selection and vanishes as $\epsilon \rightarrow 0$.

The KronSAE approach is broadly compatible with sparse decoder kernels, MoE-style routing,

and different loss functions and activations, as we show in the following sections. See Appendix D for additional discussion of the method’s semantics and Appendix B for computational costs.

4 Experiments

In this section, we examine hyperparameter choices, the resulting trade-offs, and KronSAE’s compatibility with existing SAE variants.

Experimental setup. We train SAEs on the residual streams of the Qwen2.5-1.5B-Base (Yang et al., 2024) and Gemma-2-2B (Gemma Team et al., 2024) language models. Activations are collected from FineWeb-Edu, a filtered subset of educational web pages from the FineWeb corpus (Penedo et al., 2024). We measure reconstruction quality using explained variance (EV),

$$\text{EV} = 1 - \frac{\text{Var}(\mathbf{x} - \hat{\mathbf{x}})}{\text{Var}(\mathbf{x})}, \quad (5)$$

where 1.0 is optimal, and use an automated interpretability pipeline (Bills et al., 2023; Paulo et al., 2025) to measure the interpretability of learned latents (see Appendix F).

Reconstruction fidelity. We sweep over $m \in \{1, 2, 4, 8\}$, $h \in \{512, 1024, 2048, 4096\}$, $n = F/(hm)$, $\ell_0 \in \{8, 16, 32, 64\}$, and $F \in \{2^{14}, 2^{15}, 2^{16}\}$ at early, middle, and late layers. We evaluate KronSAE against TopK SAE using a budget of 250M tokens. For most setups with fixed m , we report frontier performance across n and h , together with dispersion over all configurations. As Figure 3 shows, under an equal token budget KronSAE achieves EV comparable to TopK SAE with fewer encoder FLOPs. The conclusions for layers 6 and 20 are consistent with those for layer 12 (Appendix C). We also report MSE for all layers in Appendix C.1.

Compatibility with other SAEs. The method is compatible with existing SAE variants; we test it with two related architectures on layer 12 of Gemma-2-2B. Combining a Matryoshka SAE (Bussmann et al., 2025) with our encoder improves latent interpretability (Figure 5) while maintaining the same representation-quality trade-off (Figure 4). For the Switch SAE (Mudide et al., 2025), our Kronecker encoder improves quality relative to the baseline (Figure 4). At the selected dictionary size, Matryoshka-512 and Matryoshka-1024 have nesting depths of 6 and 5, respectively. Appendix A

provides implementation and notation details and reports tests with JumpReLU and Gated SAEs.

Interpretability trade-off. For this analysis, we select the commonly studied middle layer of both Qwen and Gemma with dictionary size $F = 16,384$, and use a larger training budget of 1B tokens. We sweep over m , n , and h to characterize the trade-off among interpretability metrics, explained variance, and KronSAE hyperparameters. As Figure 5 shows, KronSAE’s interpretability scores improve relative to TopK SAE as m increases.

Computational cost. KronSAE also reduces encoder-side computation: it projects the input to only $h(m + n)$ pre-latents instead of directly projecting to all $F = hmn$ post-latents. Figure 6 compares KronSAE configurations from the ablation sweep with the TopK SAE baseline in terms of encoder FLOPs (see Appendix B) and explained variance. The resulting Pareto curve shows that KronSAE can substantially reduce computation while preserving reconstruction quality.

5 Study of Correlation Learning

In this section, we analyze how KronSAE captures correlations between features in a real-world setting and in a controlled toy experiment.

5.1 Real correlations.

To test whether KronSAE heads capture correlated feature structure, we compare a trained KronSAE with a standard TopK SAE trained on the same model activations. We first compute Pearson correlations between TopK feature activations and, separately, between each TopK feature and each KronSAE feature. For every TopK feature i , we retain its top- r most correlated KronSAE matches, denoted $\mathcal{M}_r(i)$. We then select TopK feature pairs (i, j) whose activation correlation exceeds a threshold τ , and count a hit when any feature in $\mathcal{M}_r(i)$ and any feature in $\mathcal{M}_r(j)$ belong to the same KronSAE head. The score in Figure 7 is the fraction of correlated TopK pairs that produce such a same-head hit. This tests whether KronSAE groups correlated features from a flat TopK SAE within the same heads, and whether it does so more often than expected by chance. The chance rate is approximately $1 - (1 - 1/h)^{r^2} \approx r^2/h$, far below the observed rate. Figure 1 provides a geometric view of this behavior.

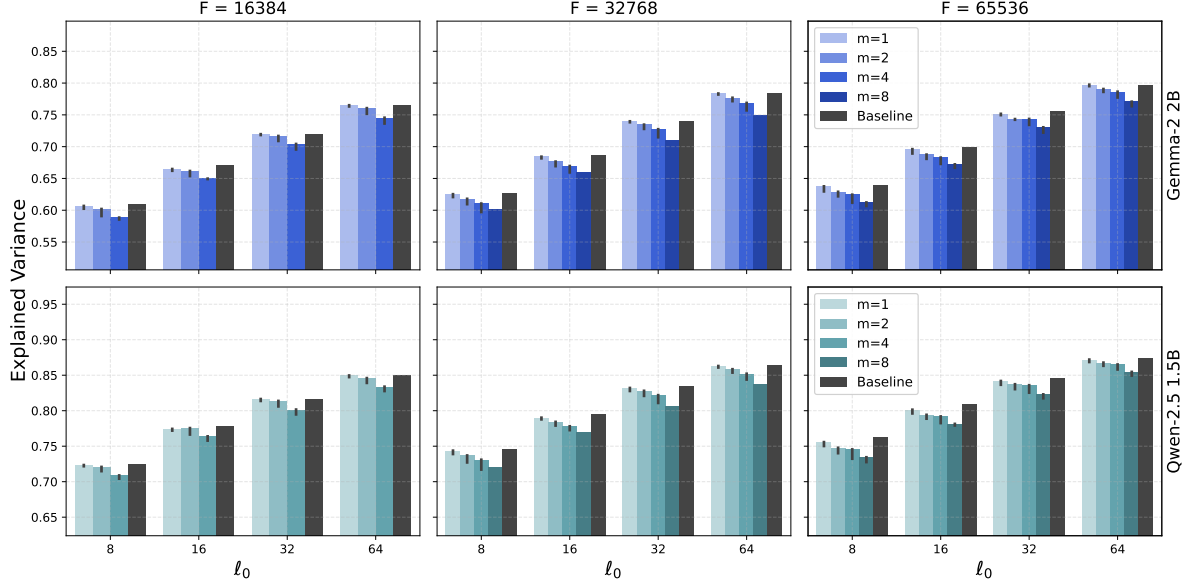


Figure 3: Maximum performance for KronSAE and TopK SAE on Qwen2.5-1.5B and Gemma-2-2B at different dictionary sizes F after training on 250M tokens. KronSAE uses fewer encoder parameters while remaining on par with the baseline in EV; the gap narrows at larger dictionary sizes on layer 12 of both models.

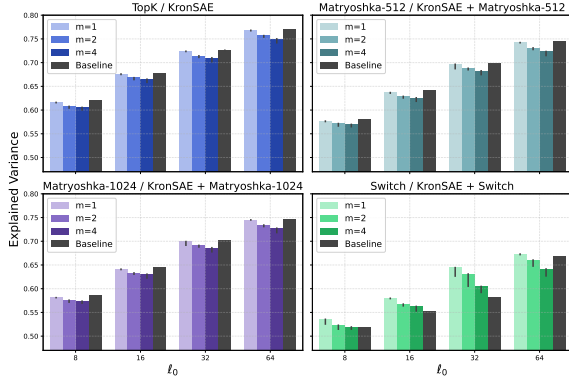


Figure 4: Applying the Kron encoder to TopK, Switch, and Matryoshka SAEs. Switch+Kron matches the Switch baseline in most configurations and improves it in some; Matryoshka+Kron remains on par across the sweep.

To test KronSAE’s co-activation inductive bias, we select a KronSAE with $m = 4$ and $n = 4$ and calculate correlations on 5K texts from the training dataset. For each feature, we compute the mean correlation with features in its head and with all other features, then compare a randomly initialized KronSAE with the trained model. To isolate the effect of our initialization procedure, we sample the weights from a uniform distribution. As Figure 8 shows, correlations are higher within a head and increase further after training, suggesting that the imposed correlation structure works as intended.

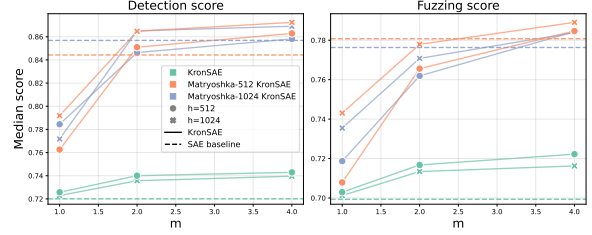


Figure 5: Detection and fuzzing scores for baseline SAEs and their Kron variants. Scores improve with increasing m ; with $m \geq 2$, all Kron variants match or exceed their baselines.

5.2 Toy model of correlations

To evaluate how well different sparse autoencoder architectures recover underlying correlation patterns, we construct a controlled sanity-check experiment using a synthetic, block-structured covariance model, where KronSAE’s inductive bias is expected to align with the ground-truth structure. We generate sparse nonnegative ground-truth feature activations, linearly mix them using a Cholesky factor L of a block-structured matrix $S = LL^T$, and train a tied-weight bottleneck autoencoder following Elhage et al. (2022). After training, we match the learned SAE decoder atoms with the AE atoms by solving a quadratic assignment problem (Vogelstein et al., 2015) (see Appendix E for motivation and details), and compute the decoder-atom similarity matrix $C_{\text{dec}} = W_{\text{dec}}^T W_{\text{dec}}$.

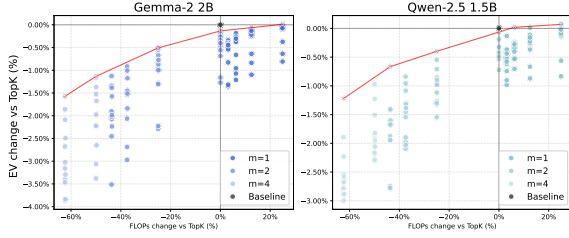


Figure 6: Reconstruction–compute trade-off between TopK SAE and KronSAE. KronSAE reduces encoder FLOPs by more than 40% while incurring less than a 1% drop in EV. See Appendix B and the full comparison in Figure 12.

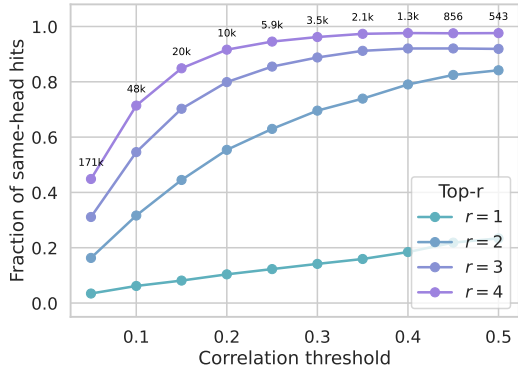


Figure 7: Fraction of correlated TopK feature pairs whose top- r KronSAE matches share at least one Kron head. We select TopK feature pairs with activation correlation above the threshold on the x -axis, map each feature to its top- r correlated KronSAE features, and count a hit if the two candidate sets contain a pair from the same Kron head. All observations are far above the chance value r^2/h .

To quantify how closely SAE atoms mirror the original structure of S , we use the RV coefficient, defined as $\text{RV}(A, B) = \text{tr}(A^T B) / \sqrt{\text{tr}(A^T A) \text{tr}(B^T B)}$. Figure 9 shows the results. KronSAE achieves a higher RV coefficient than TopK SAE in this synthetic setting, confirming that the inductive bias is expressive enough to recover block structure when it is present. Appendix E provides a detailed description and additional setups.

6 Analysis

6.1 Absorption

A persistent 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 narrower representation (e.g., a “starts with L” feature is subsumed by a “Lion”

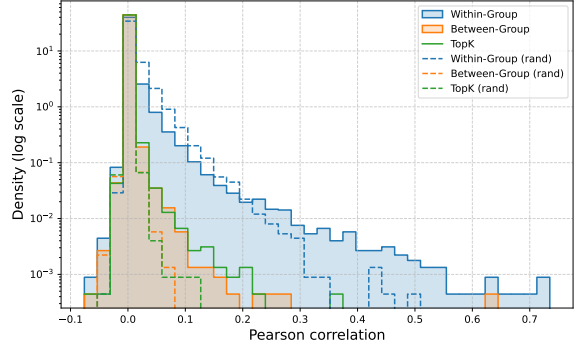


Figure 8: Correlations between features in a KronSAE with $m = 4, n = 4$, measured within a head and across heads. Our design induces higher within-head correlations, which become stronger after training, although the SAE also learns correlations across heads.

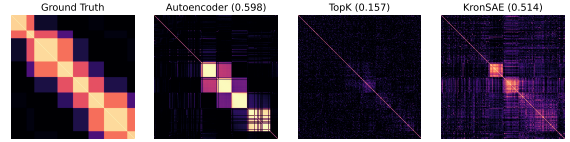


Figure 9: Synthetic block-correlation experiment. Each panel shows a feature-similarity matrix for the target block structure, the bottleneck AE representation, TopK SAE, and KronSAE. Numbers in parentheses are RV coefficients against the target after atom matching. Results show that KronSAE recovers the block structure substantially better than TopK SAE. See details in the Appendix E.

feature) (Chanin et al., 2025).

Figure 10 reports three absorption metrics measured with SAEbench (Karvonen et al., 2025) across sparsity levels $\ell_0 \in \{8, 16, 32, 64\}$: (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 dictionary size $F = 2^{14}$ and compare TopK SAE, Matryoshka SAE (Bussmann et al., 2025), and their KronSAE variants, which impose different hierarchical priors (see Appendix D). Across all ℓ_0 values, KronSAE variants tend to reduce the first two scores relative to the TopK SAE baseline while maintaining a similar rate of feature splitting.

We hypothesize that the observed reductions in absorption are consistent with two complementary architectural biases: a mechanism where a post-latent can be active only when both pre-latents are active can reduce full-absorption events by limiting

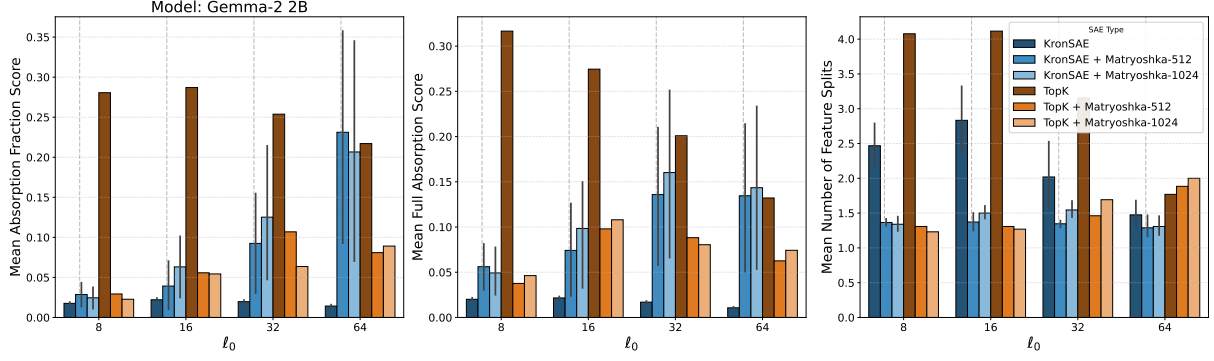


Figure 10: Feature absorption metrics on Gemma-2-2B. KronSAE configurations (various m, n) exhibit lower mean absorption fractions and full-absorption scores across different ℓ_0 values and selected baselines.

activation when a parent is inactive (see additional description in Appendix D); and dividing the latent space into h independent subspaces (each with its own base and extension factors) encourages correlated features to localize within a head (Section 5), which can reduce cross-head absorption.

6.2 Analysis of Learned Features

In this section, we compare KronSAE and TopK SAE in terms of interpretability and feature properties, and analyze the properties of KronSAE groups. We use layer 12 of Gemma-2-2B and a dictionary size of 16,384 features. The selected KronSAE has $m = 4$ and $n = 4$. We process 24M tokens to collect the analysis data. Our interpretation pipeline follows a common methodology: an LLM interprets activation patterns (Bills et al., 2023), and we evaluate the resulting interpretations using *detection* and *fuzzing* scores (Paulo et al., 2025) (see Appendix F).

For each selected feature, in addition to mean activation and frequency, we calculate two metrics. Low *token entropy* suggests that a feature activates frequently on a small number of tokens and is therefore token-specific. A high *multitoken ratio* indicates that a feature tends to activate multiple times in a single sequence. Both metrics have a notable negative correlation with the final interpretability scores and may therefore provide useful proxies without requiring full interpretation-based evaluation.

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 11. Since post-latents are significantly more

interpretable than corresponding pre-latents, we hypothesize the hidden mechanism for encoding and retrieval of the required semantics.

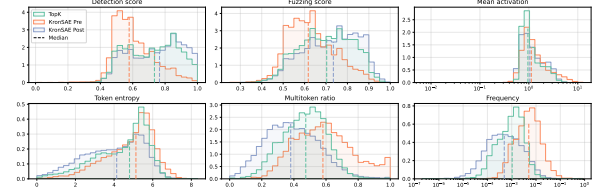


Figure 11: Property distributions for TopK SAE and KronSAE ($m = 4, n = 4$) with a 16K dictionary trained on Gemma-2-2B. The Pre and Post suffixes denote pre- and post-latents; the y -axis shows density. Token entropy is the entropy of the token distribution on which a feature activates, and the multitoken ratio measures how often a feature activates within a single sequence. KronSAE achieves better interpretability scores while learning more specialized feature groups, as indicated by lower activation frequency and lower variance among activated tokens.

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 1, 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

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	Comparative expressions involving “than” and “as” in contrastive or proportional relationships.	0.87
<i>Composition:</i>	Similarity or analogy through the suffix “-like” across diverse contexts.	0.89
Extension 1	Specific terms, names, and abbreviations that are contextually salient and uniquely identifiable.	0.66
<i>Composition:</i>	Medical terminology related to steroids, hormones, and their derivatives.	0.84
Extension 2	Spiritual concepts and the conjunction “as” in varied syntactic roles.	0.80
<i>Composition:</i>	Abstract concepts tied to spirituality, consciousness, and metaphysical essence.	0.93
Extension 3	Directional and regional descriptors indicating geographical locations or cultural contexts.	0.84
<i>Composition:</i>	Directional terms indicating geographical or regional divisions.	0.91

Table 1: 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 pre-latents. See details in Section 6.2.

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 groups of semantically related pre-latents, e.g. in head 136 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. Retrieval happens primarily through a mechanism resembling a logical AND circuit, where one pre-latent carries multiple semantics and the other acts as a specifier. Table 1 gives an illustrative example: the base contains three detectable sub-semantics, and each extension retrieves one of them.

Other types of interaction may occur, including apparently novel semantics (e.g., the composition of base 3 and extension 1 in Table 1). In other cases, a post-latent inherits detectable semantics from only one parent, particularly when the other parent has a broad interpretation and a low score. Identifying the fine-grained structure of these interactions requires more sophisticated techniques,

which we leave to future work.

See more examples in Appendix H. For details on dependencies between feature properties and feature geometry, see Appendix G.

7 Conclusion

We introduce **KronSAE**, a sparse autoencoder architecture that combines head-wise Kronecker factorization of the latent space with an approximation of logical AND through the mAND nonlinearity, inducing a co-activation prior on latent structure. Our approach enables efficient training of compositional SAEs, especially with limited compute or training data, while maintaining reconstruction fidelity and yielding more interpretable features. Our analysis links these gains to the complementary effects of compositional latent structure and logical AND-style interactions, offering a new perspective on how sparsity and factorization can work together in representation learning.

Limitations

KronSAE introduces a clear trade-off among interpretability, efficiency, and reconstruction performance. Our evaluation is limited to mid-sized transformer models. Although the overall training

and inference infrastructure is similar to that of standard SAEs, replacing the dense encoder with the proposed factorized encoder requires additional implementation work. The resulting computation is highly parallelizable and is already reasonably efficient with standard tensor operations, even without specialized kernels. We therefore view this engineering overhead as moderate relative to KronSAE’s interpretability and compute trade-offs. Finally, the toy-model setup is intended primarily as a sanity check of KronSAE’s inductive bias; real-world correlation and hierarchical structure may differ. Our real-data correlation experiment provides a more direct, practically oriented evaluation.

References

- Nikita Balagansky, Ian Maksimov, and Daniil Gavrilov. 2025. [Mechanistic permutability: Match features across layers](#). In *The Thirteenth International Conference on Learning Representations*.
- Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. [Language models can explain neurons in language models](#).
- Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. [Towards monosemanticity: Decomposing language models with dictionary learning](#). *Transformer Circuits Thread*.
- Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. 2025. [Learning multi-level features with matryoshka sparse autoencoders](#). In *Forty-second International Conference on Machine Learning*.
- David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. 2025. [A is for absorption: Studying feature splitting and absorption in sparse autoencoders](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 82318–82355. Curran Associates, Inc.
- Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2024. [Sparse autoencoders find highly interpretable features in language models](#). In *International Conference on Learning Representations (ICLR)*.
- Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Parvoti Nia, James J. Clark, and Mehdi Rezagholizadeh. 2021. [Kronecker decomposition for gpt compression](#). *Preprint*, arXiv:2110.08152.
- Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. [Toy models of superposition](#). *Transformer Circuits Thread*.
- Thomas Fel, Ekdeep Singh Lubana, Jacob S. Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba E. Ba, and Talia Konkle. 2025. [Archetypal SAE: Adaptive and stable dictionary learning for concept extraction in large vision models](#). In *Forty-second International Conference on Machine Learning*.
- Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. [Scaling and evaluating sparse autoencoders](#). In *The Thirteenth International Conference on Learning Representations*.
- Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. [Gemma 2: Improving open language models at a practical size](#). *Preprint*, arXiv:2408.00118.
- Thomas Heap, Tim Lawson, Lucy Farnik, and Laurence Aitchison. 2026. [Automated interpretability metrics do not distinguish trained and random transformers](#). In *International Conference on Learning Representations (ICLR)*.
- Dhruva Karkada, Daniel James Korchinski, Andres Nava, Matthieu Wyart, and Yasaman Bahri. 2026. [Symmetries in language statistics shape the geometry of model representations](#). In *Forty-third International Conference on Machine Learning*.
- Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Isaac Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum Stuart McDougall, Kola Ayonrinde, Demian Till, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. 2025. [SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability](#). In *Forty-second International Conference on Machine Learning*.
- Scott Lowe, Robert Earle, Jason d'Eon, Thomas Trappenberg, and Sageev Oore. 2022. [Logical activation functions: Logit-space equivalents of probabilistic boolean operators](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 29733–29747. Curran Associates, Inc.
- Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2025. [Sparse feature circuits: Discovering and editing interpretable causal graphs in language models](#). In *The Thirteenth*

International Conference on Learning Representations.

Anish Mudide, Joshua Engels, Eric J Michaud, Max Tegmark, and Christian Schroeder de Witt. 2025. [Efficient dictionary learning with switch sparse autoencoders](#). In *The Thirteenth International Conference on Learning Representations*.

Charles O’Neill, Alim Gumran, and David Klindt. 2025. [Compute optimal inference and provable amortisation gap in sparse autoencoders](#). In *Forty-second International Conference on Machine Learning*.

Gonalo Paulo and Nora Belrose. 2026. [Sparse autoencoders trained on the same data learn different features](#). In *The Fourteenth International Conference on Learning Representations*.

Gonalo Santos Paulo, Alex Troy Mallen, Caden Juang, and Nora Belrose. 2025. [Automatically interpreting millions of features in large language models](#). In *Forty-second International Conference on Machine Learning*.

Guilherme Penedo, Hynek Kydlíek, Loubna Ben al-lal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. [The fineweb datasets: Decanting the web for the finest text data at scale](#). In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. 2024a. [Improving sparse decomposition of language model activations with gated sparse autoencoders](#). In *NeurIPS*.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. 2024b. [Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders](#). *Preprint*, arXiv:2407.14435.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczy, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Jeff Dean. 2017. [Outrageously large neural networks: The sparsely-gated mixture-of-experts layer](#). In *International Conference on Learning Representations (ICLR)*.

P. Smolensky. 1990. [Tensor product variable binding and the representation of symbolic structures in connectionist systems](#). *Artif. Intell.*, 46(1–2):159–216.

Paul Smolensky, Roland Fernandez, Zhenghao Herbert Zhou, Mattia Oppen, and Jianfeng Gao. 2024. [Mechanisms of symbol processing for in-context learning in transformer networks](#). *Preprint*, arXiv:2410.17498.

Paul Soulos, R. Thomas McCoy, Tal Linzen, and Paul Smolensky. 2020. [Discovering the compositional structure of vector representations with role learning networks](#). In *Proceedings of the Third BlackboxNLP*

Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 238–254, Online. Association for Computational Linguistics.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, and 3 others. 2024. [Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet](#). *Transformer Circuits Thread*.

Joshua T. Vogelstein, John M. Conroy, Vince Lyzinski, Louis J. Podrazik, Steven G. Kratzer, Eric T. Harley, Donniell E. Fishkind, R. Jacob Vogelstein, and Carey E. Priebe. 2015. [Fast approximate quadratic programming for graph matching](#). *PLoS ONE*, 10(4):e0121002.

Peihao Wang, Rameswar Panda, Lucas Torroba Hen-nigen, Philip Greengard, Leonid Karlinsky, Roge-rio Feris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. 2023. [Learning to grow pretrained models for efficient transformer training](#). In *The Eleventh International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. [Qwen2.5 technical report](#). *arXiv preprint arXiv:2412.15115*.

A Experimental Setup Details

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 learning rate of 1×10^{-6} , a linear warmup over the first 10% of training steps, and an auxiliary-loss coefficient of 0.03125. We use a global batch size of 8,192. We sweep over dictionary sizes $F \in \{2^{14}, 2^{15}, 2^{16}\}$. For KronSAE, we also sweep the number of heads h and per-head dimensions m and n , subject to $hmn = F$. Regularization weights and auxiliary-loss coefficients are held constant across runs to isolate the effect of architecture.

Across all experiments, including preliminary research, we used approximately 330 GPU-days on NVIDIA H100 80GB GPUs.

Initialization. As observed by Gao et al. (2025), initializing the decoder as the transpose of the encoder ($W_{\text{dec}} = W_{\text{enc}}^\top$) substantially improves performance. We adapt this strategy to KronSAE by partitioning W_{enc} into h head-wise pairs of blocks with shapes $m \times d$ and $n \times d$, denoted $\{P_k, Q_k\}_{k=1}^h$. For each head k , we define decoder rows through additive composition:

$$C_k[i, j] = P_{k,i} + Q_{k,j},$$

with $i = 1, \dots, m$ and $j = 1, \dots, n$. Flattening the matrices $\{C_k\}$ yields the full decoder weight matrix $W_{\text{dec}} \in \mathbb{R}^{F \times d}$ in the row-vector convention used by our implementation.

Matryoshka and Kron-based version. For a Matryoshka SAE (Bussmann et al., 2025) with dictionary size F , we use the training settings above and define dictionary groups $S = [2^k, 2^k, \dots, 2^{k+i}, \dots, 2^n]$, where $k < n$ and $\sum_{s \in S} s = F$. This is equivalent to nested subSAEs with dictionary sizes $\mathcal{M} = \{2^k, 2^k + 2^k, 2^k + 2^k + 2^{k+1}, \dots, F\}$. We refer to this model as **Matryoshka-2^k**. Its reconstruction loss is

$$\mathcal{L}(\mathbf{f}) = \sum_{m \in \mathcal{M}} \left\| \mathbf{x} - \left(\mathbf{f}_{0:m} W_{0:m}^{\text{dec}} + \mathbf{b}_{\text{dec}} \right) \right\|_2^2, \quad (6)$$

where $\mathbf{f}$ is the latent vector. During training, we also use the auxiliary loss $\mathcal{L}_{\text{aux}}$ from Gao et al. (2025).

For the Kron-based Matryoshka SAE, we use the same setup and initialization. We choose m and n so that $F \bmod (mn) = 0$, 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 use 8 experts in all experiments and the same initialization as KronSAE. In the Kron-based variant, we distribute the heads equally across experts, yielding $F/(8mn)$ heads per expert. This reduces both the number of parameters in each expert and encoder FLOPs.

Comparison with JumpReLU and Gated SAE. We compare KronSAE with alternative activation mechanisms, including JumpReLU and Gated SAE, and report explained variance at three sparsity levels in Table 2. For Gated SAEs, the reconstruction performance of Gated-KronSAE is on par with that of the baseline.

B Computational Cost

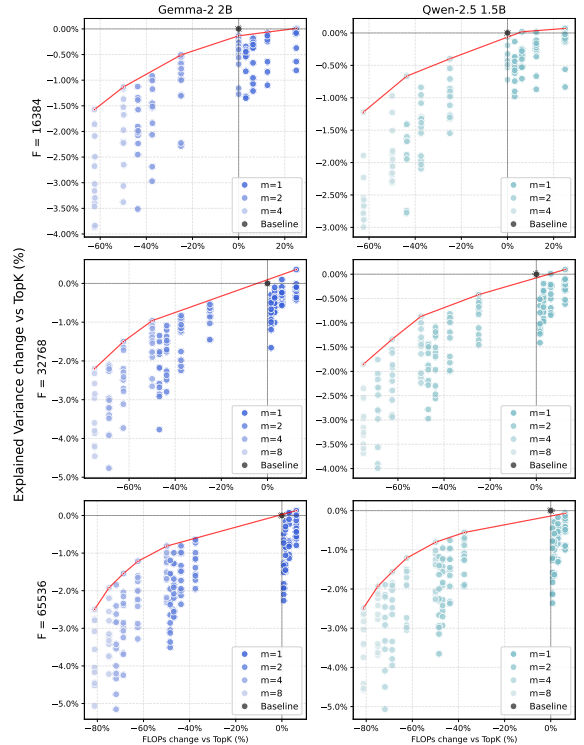


Figure 12: Trade-off between FLOPs and explained variance. In most settings, KronSAE substantially reduces compute with only a small change in EV.

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

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

SAE	$\ell_0 = 32$	$\ell_0 = 50$	$\ell_0 = 64$
TopK	0.809	0.837	0.852
JumpReLU	0.813	0.838	0.844
Kron+TopK	0.814	0.840	0.853
Kron+JumpReLU	0.790	0.817	0.828

SAE	ℓ_0	EV $\uparrow$
Kron+Gated ($m=1, n=32$)	63	0.8319
Gated (run 1)	68	0.83050
Kron+Gated ($m=1, n=16$)	63	0.83027
Kron+Gated ($m=2, n=8$)	66	0.82060
Gated (run 2)	63	0.81300

Table 2: Activation-function ablation for KronSAE. **Top**: performance of SAEs with JumpReLU and TopK activations. Because JumpReLU has variable sparsity controlled by an ℓ_0 penalty coefficient, we sweep over sparsity levels, fit a quadratic curve to the resulting EV values as a function of ℓ_0 , and evaluate the curve at the reported levels. TopK and KronSAE with TopK are instead trained at fixed, predefined sparsity levels. **Bottom**: comparison of the Gated SAE architecture with its KronSAE variant for dictionary size $F = 2^{16}$.

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

$$\begin{aligned} \text{FLOPs}_{\text{Kron}} &= dh(m + n) + mn h + kd \approx \\ &\approx dh(m + n) + kd = d(h(m + n) + k), \end{aligned} \quad (8)$$

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

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 13 reports how step time scales with the hidden dimension d .

C Performance on Different Layers

Figures 14 and 15 show that KronSAE is on par with TopK SAE when $m = 1$. Other configurations incur only small EV changes while substantially reducing the number of encoder parameters.

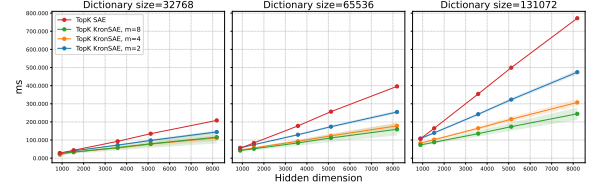


Figure 13: Training-step time for TopK SAE and KronSAE across hidden dimensions. KronSAE scales more favorably than an SAE with the standard encoder architecture.

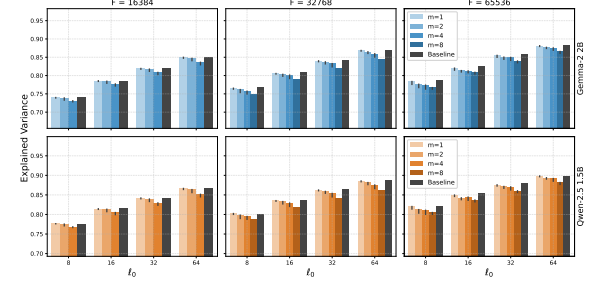


Figure 14: Maximum performance for KronSAE and TopK SAE on Qwen2.5-1.5B and Gemma-2-2B at different dictionary sizes F under an equal token budget. KronSAE uses fewer encoder parameters while remaining on par with the baseline in EV; the gap narrows at larger dictionary sizes on **layer 6** of both models.

C.1 MSE Performance

We also report MSE results for layers 6, 12, and 20 in Figure 23. 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.

D mAND as a Logical Operator and KronSAE as a Logical SAE

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. Apply ReLU to latents $\mathbf{u}$ and $\mathbf{v}$, then apply the Kronecker product and square root.
2. Apply the mAND kernel to create the matrix in Equation 3, then flatten it.

These descriptions are equivalent. To understand how they induce the AND-like mechanism, consider base and extension pre-latent vectors $\mathbf{p} = \text{ReLU}(\mathbf{u})$ and $\mathbf{q} = \text{ReLU}(\mathbf{v})$. Here, p_i is the i th base activation and q_j is the j th extension activation. The Kronecker product $\mathbf{p} \otimes \mathbf{q}$ creates the post-latent activation vector $(p_1 q_1, \dots, p_1 q_n, p_2 q_1, \dots, p_m q_n)$. Therefore:

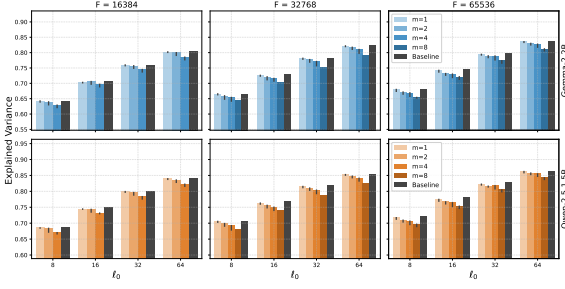


Figure 15: Maximum performance for KronSAE and TopK SAE on Qwen2.5-1.5B and Gemma-2-2B at different dictionary sizes F . KronSAE uses fewer encoder parameters while remaining on par with the baseline in EV; the gap narrows at larger dictionary sizes on **layer 20** of both models.

1. A post-latent is active $\implies$ both parent pre-latent activations are positive.
2. A post-latent is inactive $\implies$ at least one parent pre-latent activation is zero.

Hence, a post-latent is active only when both parent pre-latents are active. Fix a post-latent and its parents, and let $\mathcal{P}$, $\mathcal{Q}$, and $\mathcal{F}$ be the sets of hidden-state inputs on which the base pre-latent, extension pre-latent, and post-latent are active, respectively. Then $\mathcal{F} \subseteq \mathcal{P} \cap \mathcal{Q}$. Thus, pre-latents can be broader and more polysemous than their emitted post-latents. We validate this behavior qualitatively in Section 6.2, where we also observe interactions that may exploit activation magnitude and do not appear to be simple intersections.

Matryoshka and Kron hierarchies. In contrast to TopK SAE, the Matryoshka loss imposes a hierarchy by dividing the dictionary into groups of $G_1, \dots, G_k$ latents at different levels of granularity. The first G_1 latents are broad and abstract; each subsequent group adds finer-grained semantics. The loss requires progressively larger prefixes to reduce reconstruction error while the smallest prefix remains useful, but it does not require conditional feature activation. KronSAE instead imposes a two-level AND-like hierarchy through an architectural co-activation prior. Because these mechanisms act through the loss and encoder architecture, respectively, they can be combined, as shown in Sections 4 and 6.1.

Visual intuition. We also compare mAND with AND_{AIL} (Lowe et al., 2022). Because our objective is to drive each atom toward a distinct, monosemantic feature, tightening the logical conjunction

can encourage sharper feature separation. Moreover, using the geometric mean ($\sqrt{pq}$) instead of a simple product or minimum moderates the scale of the post-latent activation when both p and q are positive. Figure 16 provides a visual comparison.

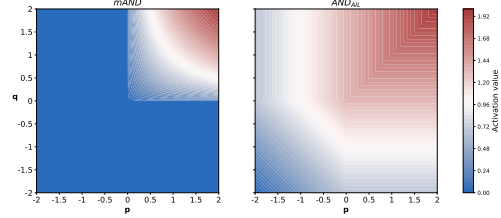


Figure 16: Comparison of the smooth mAND operator with AND_{AIL} (Lowe et al., 2022).

E Synthetic Correlations Experiment

This section describes the setup and metrics for the toy-model study of correlation recovery.

E.1 Correlated Toy 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.

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

$$\hat{\mathbf{x}} = \text{ReLU}(W^\top W \cdot \bar{\mathbf{x}}_{\text{sparse}} + \mathbf{b}). \quad (9)$$

We collect $d = 64$ dimensional hidden states ($W\bar{\mathbf{x}}_{\text{sparse}}$) from the AE, then train a TopK SAE and KronSAE with $F = 256$ and $K = 8$ to reconstruct them. After training, we extract each SAE’s decoder weight matrix W_{dec} , match its latents to the AE latents by solving the quadratic assignment problem (Vogelstein et al., 2015) described below, and compute the decoder-atom similarity matrix $C_{\text{dec}} = W_{\text{dec}}W_{\text{dec}}^\top$.

E.2 Quadratic Matching Algorithm

We next motivate our matching algorithm and present additional results.

Suppose two sets of feature embeddings from different models are represented by matrices $X, Y \in \mathbb{R}^{F \times d}$, where F is the number of features and d is the embedding dimension. Our task is to find an assignment from features in Y to features in X that best preserves their global structure.

The standard approach solves a linear assignment problem: find a permutation matrix Π that minimizes $\text{tr}(C^\top \Pi)$, where $C_{i,j} = d(X_i, Y_j)$ is a pairwise feature-distance matrix. The Hungarian algorithm is commonly used for this purpose in sparse dictionary learning (Paulo and Belrose, 2026; Balagansky et al., 2025; Fel et al., 2025).

This linear problem considers pairwise feature information but ignores dependencies within the X and Y sets; for example, clusters should map to clusters. We therefore seek a permutation Π that maximizes *global* alignment, measured by the squared Frobenius norm of $X^\top \Pi Y$:

$$\begin{aligned} \max_{\Pi} \|X^\top \Pi Y\|_F^2 &= \\ &= \max_{\Pi} \text{tr}(\Pi^\top X X^\top \Pi Y Y^\top), \end{aligned} \quad (10)$$

where global feature structure is explicitly encoded in the matrices $X X^\top$ and $Y Y^\top$.

We solve this quadratic assignment problem using the Fast Approximate Quadratic assignment (FAQ) method (Vogelstein et al., 2015), originally developed for graph matching. Given adjacency matrices A and B , FAQ relaxes Π from the set of permutation matrices to the set of doubly stochastic matrices (the Birkhoff polytope) and optimizes the corresponding trace objective. In our case, $A = X X^\top$ and $B = Y Y^\top$, and we maximize similarity:

$$\begin{aligned} \max_{\Pi} \text{tr}(X X^\top \Pi Y Y^\top \Pi^\top) &= \\ &= \max_{\Pi} \text{tr}(\Pi^\top X X^\top \Pi Y Y^\top), \end{aligned} \quad (11)$$

which is equivalent to Equation 10. 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 matching procedure.

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

```
def feature_matching(A, B, max_iter):
    F, d = A.shape
    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
        a = np.trace(G_A @ D @ G_B @ D.T)
        b = np.trace(grad.T @ D)
```

```
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
```

E.3 Results

To quantify how closely each model’s feature correlations mirror the original structure of S , we use the RV coefficient, $\text{RV}(S, C) = \text{tr}(SC) / \sqrt{\text{tr}(S^2) \text{tr}(C^2)}$. As Figure 17 shows, KronSAE retains a stronger correlation structure than TopK SAE and aligns better with the ground-truth covariance matrix. Table 3 shows that KronSAE’s learned structure varies more across covariance setups, as reflected by the reported standard deviations, and aligns better with ground truth according to higher RV scores and a smaller effective-rank difference Δ .

F Feature Analysis Methodology

We analyze learned features using an established pipeline (Bills et al., 2023; Paulo et al., 2025) consisting of three stages: (1) statistical property collection, (2) automatic activation pattern interpretation, and (3) interpretation evaluation.

F.1 Data collection

Our collection process uses a fixed-size buffer of $B = 384$ examples per feature and continues until reaching a predetermined maximum token count T_{max} . The procedure is as follows.

Initial processing batches generate large activation packs of 1M examples, each comprising a 256-token text segment. When a feature activates, we add the example to its buffer and randomly down-sample whenever the buffer exceeds size B . This approach supports arbitrary token volumes while accommodating rare features that 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

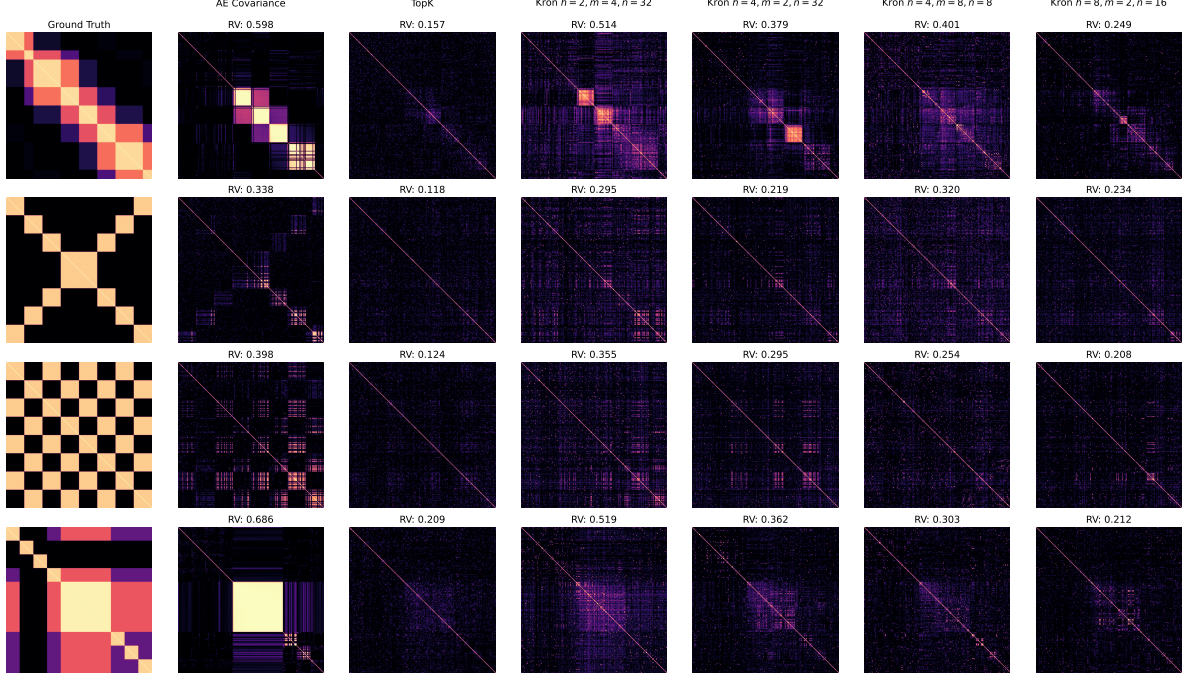


Figure 17: Patterns learned by the autoencoder, TopK SAE, and KronSAE variants after quadratic matching. KronSAE learns patterns that more closely resemble the ground-truth structure; as the number of heads increases and groups become more fine-grained, its behavior approaches that of TopK SAE.

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

Table 3: 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.

token entropy is calculated as:

$$\text{token entropy} = - \sum_{i=1}^n p_i \log p_i, \quad (12)$$

$$p_i = \frac{\text{activations on token } i}{\text{total activations}},$$

where n is the number of unique activated tokens. The multitoken ratio is computed per sequence:

$$\text{mt ratio} = \frac{1}{b} \sum_{i=1}^b \frac{\text{activations in example } i}{\text{tokens in example } i}, \quad (13)$$

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

We then segment examples using a 31-token context window (15 tokens before and after each activation), potentially creating overlapping but non-duplicated examples. Features with a high multi-token ratio may yield significantly more than B context windows.

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.

F.2 Feature interpretations

For each feature, we generate interpretations by sampling 16 activation examples above the median activation and presenting them to Qwen3-14B (Yang et al., 2025) (AWQ-quantized, with reasoning enabled). The model produces concise descriptions of the activation patterns. Our qualitative observations suggest that reasoning mode improves interpretation quality, although we do not quantify this effect. This is consistent with Paulo et al. (2025), who compare standard responses with chain-of-thought outputs, and motivates further study of reasoning during feature interpretation.

The interpretation process uses the system prompt in Figure 18. User prompts preserve all special characters because some features activate specifically on them. Figure 19 shows a representative, slightly abbreviated user prompt.

```
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 in which selected words appear between delimiters like «this». If a sequence of consecutive tokens is important, the entire sequence will appear between delimiters «just like this». The importance of each token is listed after each example in parentheses.
- Your explanation should be a concise STANDALONE PHRASE that describes observed patterns.
- Focus on the essence of the patterns, concepts, and contexts 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 similar phrases.
- Do not write what the feature does; e.g., instead of "detects heart diseases in medical reports," write "heart diseases in medical reports."
- Write the explanation on the last line, exactly after [EXPLANATION]:
```

Figure 18: System prompt for feature interpretations.

F.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., 2025), 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.

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

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

$$\text{score} = \frac{1}{2} \left(\frac{\text{TP}}{\text{total pos.}} + \frac{\text{TN}}{\text{total negs.}} \right). \quad (14)$$

Fuzzing: We «highlight» activated tokens in sampled examples, of which 50% are correctly labeled positives, 25% are mislabeled positives, and 25% are randomly labeled negatives. We present up to 8 examples per batch, and the model evaluates whether each label is correct. Scoring follows Equation 14.

G Additional Feature Analysis Results

Feature property correlations. Our analysis reveals substantial correlations between feature properties and interpretability scores (Figure 20). In particular, token entropy and mean activation may serve as proxies for feature quality without running the full interpretation pipeline. These findings are based on the first 3,072 features from 32K TopK and KronSAE models ($m = 4, n = 4$) trained on 24M tokens, and warrant validation at larger scale.

Pre-latent-to-post-latent relationships. We investigate how post-latent properties correlate with combinations of pre-latent properties, including individual values, means, products, and the mAND operation (product followed by square root). Figure 21 shows that post-latent multitoken ratio, token entropy, and frequency correlate more strongly

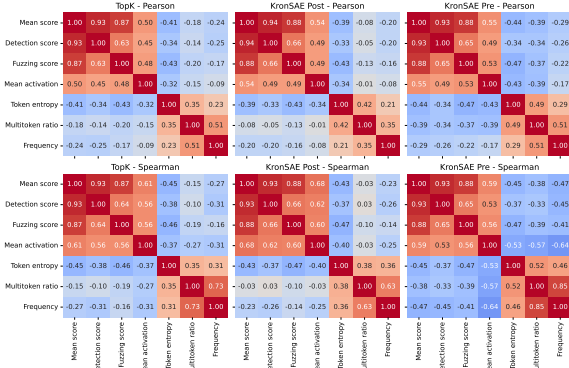


Figure 20: Correlation coefficients (Pearson and Spearman) between properties of TopK and KronSAE latents. Token entropy emerges as a strong predictor of interpretability scores, while higher mean activation and lower frequency also indicate more interpretable features.

with pre-latent products or mAND values than with individual pre-latent properties or their means.

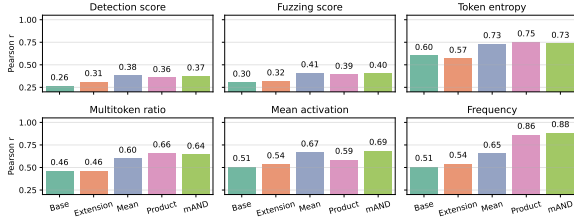


Figure 21: Correlation patterns between properties of post-latents and pre-latents.

Basis geometry. Each post-latent has a residual-stream representation given by the corresponding decoder atom, one of the overcomplete basis vectors learned during SAE training. Our architecture clusters feature embeddings: post-latents produced by the same head, base, or extension form tight groups. This geometry depends on the chosen hyperparameters h , m , and n , and may be useful for downstream applications such as steering.

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., 2025).

Figure 22 visualizes this structure through UMAP projections ($n_neighbors=15$, $min_dist=0.05$, and cosine distance) of decoder weights from the first 8 heads of 32K SAEs

with varying m, n configurations. For $m < n$, we observe tight base-wise clustering with weaker grouping by extension; for $m \geq n$, extension-wise clustering is stronger.

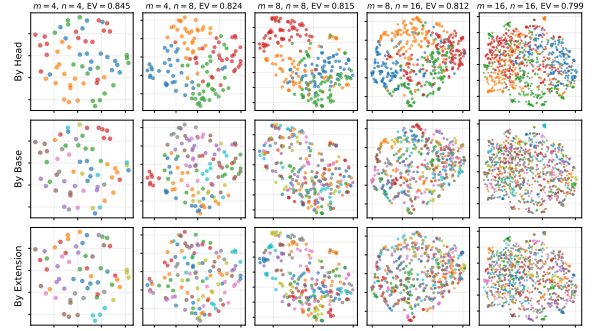


Figure 22: UMAP visualization of post-latent clustering patterns by head, base, and extension group membership. We observe tight clusters by base for $m < n$ and by extension for $m \geq n$.

This asymmetry suggests that pre-latent capacity requirements directly affect the embedding geometry: components with lower polysemanticity (extensions when $m < n$) exhibit greater geometric diversity. We expect symmetric behavior for reciprocal configurations (e.g., $m = 4, n = 8$ versus $m = 8, n = 4$), with the roles of bases and extensions exchanged.

Interpretability trade-off. As Section 4 shows, KronSAE trades off computational efficiency, explained variance, and interpretability. Increasing m and n places more correlated features in each head and improves computational efficiency, but can reduce reconstruction performance. Conversely, smaller m and finer-grained groups improve explained variance at the cost of a slight reduction in interpretability. Lower absorption scores are better; we report them alongside interpretability to make this trade-off explicit. Similar trade-offs occur across a range of SAEs (Karvonen et al., 2025).

H Analysis of Compositional Structure

Head 3. For this head, we select all base elements and extension 2, as shown in Table 4. Extension 2 has moderate interpretability and exhibits clear AND-like interactions with base 1 (semantic inheritance through shared pre-latent semantics) and base 2 (retaining only instrument-related semantics). Other interactions occur with base 0 (acquiring medical semantics while preserving metric/number aspects) and base 3 (combining necessity-related semantics with technical or medical concepts to

yield therapy/treatment concepts). The latter resembles semantic addition more than intersection. The high interpretability scores suggest encoding mechanisms beyond simple intersection, possibly involving activation magnitude, although dataset or interpretation artifacts cannot be ruled out without further validation.

Head 136. This head exhibits higher interpretability in post-latents than in pre-latents. Table 5 shows that combining extension 2 with base 0 narrows the semantics to Illinois, likely through a geographical sub-semantic. Interactions with bases 2 and 3 are more complex than simple intersection and introduce additional semantics that require further investigation.

Head 177. The latents in Table 6 demonstrate more consistent AND-like behavior than those in heads 3 and 136, closely matching the interaction pattern in Table 1.

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	Punctuation marks and line breaks serving as structural separators in text.	0.66
<i>Composition:</i>	Health-related metrics focusing on survival rates, life expectancy, and longevity.	0.88
Base 1	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
<i>Composition:</i>	Numerical digits and decimal points within quantitative values.	0.86
Base 2	Nuanced actions and adverbial emphasis in descriptive contexts.	0.71
<i>Composition:</i>	Astronomical instruments and their components, such as space telescopes and their acronyms, in scientific and observational contexts.	0.90
Base 3	Forms of the verb "to have" indicating possession, necessity, or occurrence in diverse contexts.	0.91
<i>Composition:</i>	Antiretroviral therapy components, viral infection terms, and medical treatment terminology.	0.87

Table 4: Interactions between extension 2 in head 3 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	Numerical decimal digits in quantitative expressions and proper nouns.	0.79
<i>Composition:</i>	The state of Illinois in diverse contexts with high significance.	0.95
Base 1	The number three and its various representations, including digits, Roman numerals, and related linguistic forms.	0.84
<i>Composition:</i>	Geographic place names and their linguistic variations in textual contexts.	0.91
Base 2	Ordinal suffixes and temporal markers in historical or chronological contexts.	0.87
<i>Composition:</i>	Terms indicating layers, degrees, or contexts of existence or operation across scientific, organizational, and conceptual domains.	0.82
Base 3	Question formats and topic introductions with specific terms like "What", "is", "of", "the", "Types", "About" in structured text segments.	0.77
<i>Composition:</i>	Spatial spread and occurrence of species or phenomena across environments.	0.87

Table 5: Interactions between extension 2 in head 136 and all base elements in that head.

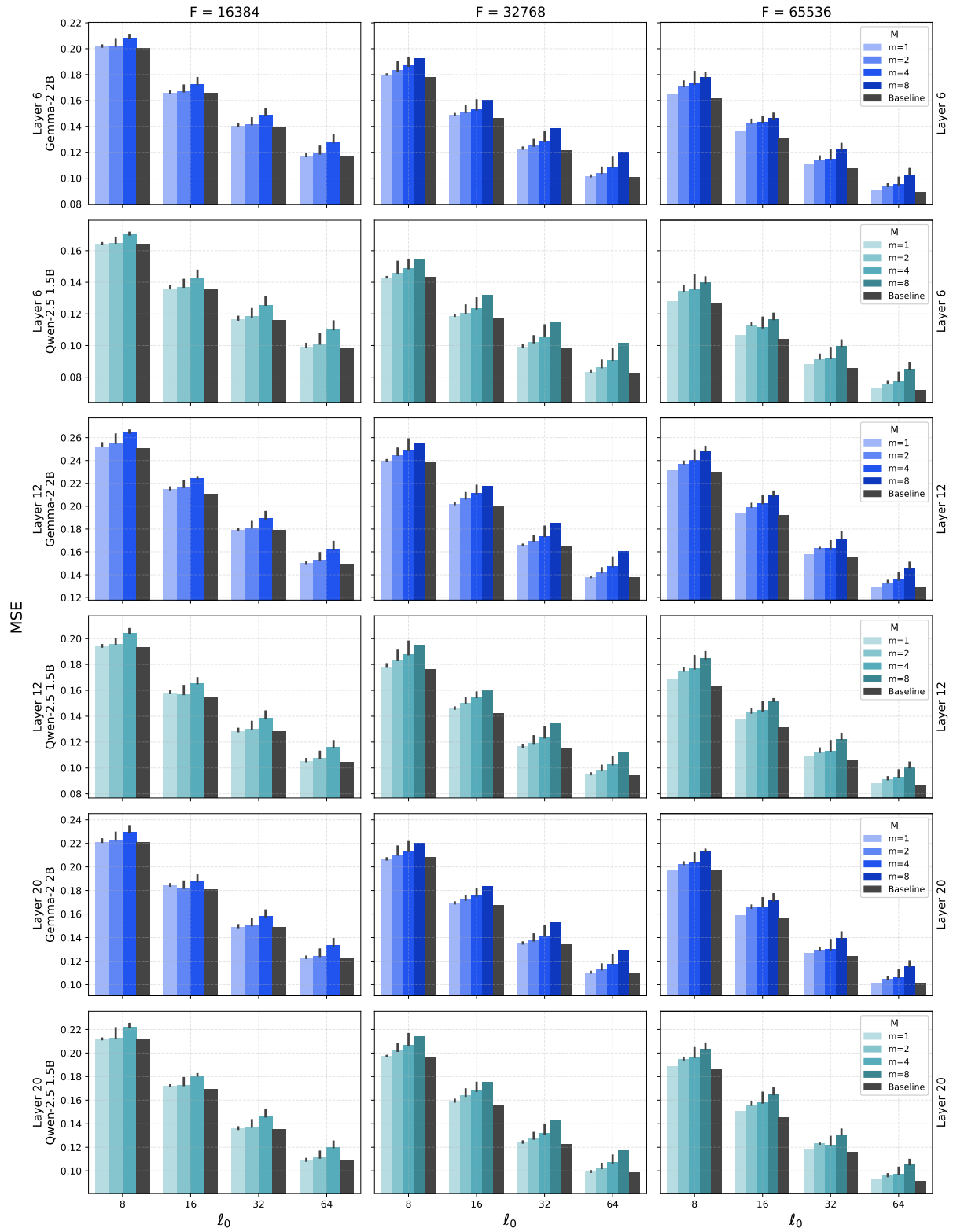


Figure 23: MSE-based comparison of KronSAE and TopK SAE under a fixed token budget.

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	Proper nouns, abbreviations, and specific named entities.	0.64
<i>Composition:</i>	Geographical or spatial references using the term "map".	0.93
Base 1	Emphasis on terms indicating feasibility and organizations.	0.80
<i>Composition:</i>	Specific organizations and societal contexts.	0.89
Base 2	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
<i>Composition:</i>	Institutional names containing "Institute" as a core term, often followed by prepositions or additional descriptors.	0.92
Base 3	Closure and termination processes, initiating actions.	0.79
<i>Composition:</i>	Initiating or establishing a state, direction, or foundation through action.	0.85

Table 6: Interactions between extension 1 in head 177 and all base elements in that head.

I KronSAE Simplified Implementation

```
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, _ = x.shape
        acts = F.relu(
            x @ self.W_enc + self.b_enc
        ).view(
            B, self.config.h, self.config.m + self.config.n
        )
        base = acts[... , :self.config.m]
        extension = acts[... , self.config.m:]
        all_scores = torch.sqrt(
            base[... , :, None] * extension[... , None, :] + 1e-5
        ).view(B, -1)
        scores, indices = all_scores.topk(
            self.config.k, dim=-1, sorted=False
        )
        acts_topk = torch.zeros_like(all_scores).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()
        var = (x - x.mean(0)).pow(2).mean()
        ev = 1 - loss / var
        return loss, ev
```