Kronecker Factorization Improves Efficiency and Interpretability of Sparse Autoencoders

Anonymous ACL submission

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.

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.

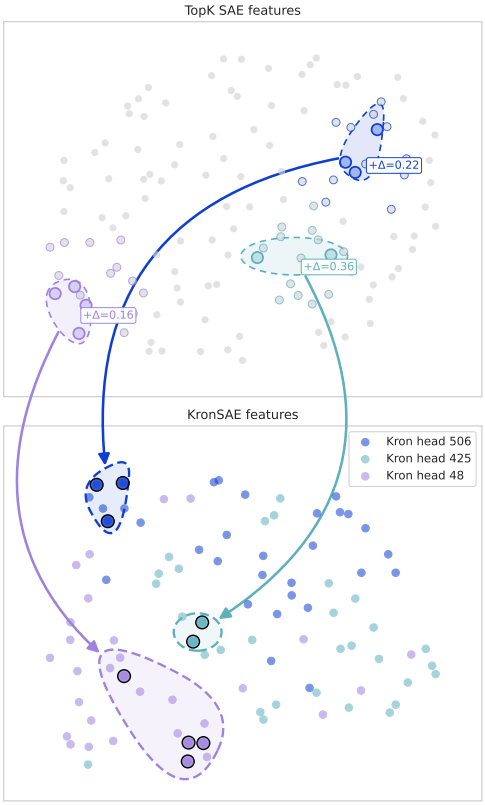

Figure 1: UMAP projection illustrating the relation between TopK SAE features and KronSAE heads. Correlated TopK features tend to map to nearby KronSAE features within the same head. Selected features correlate stronger that nearest features (Δ value between correlations).

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

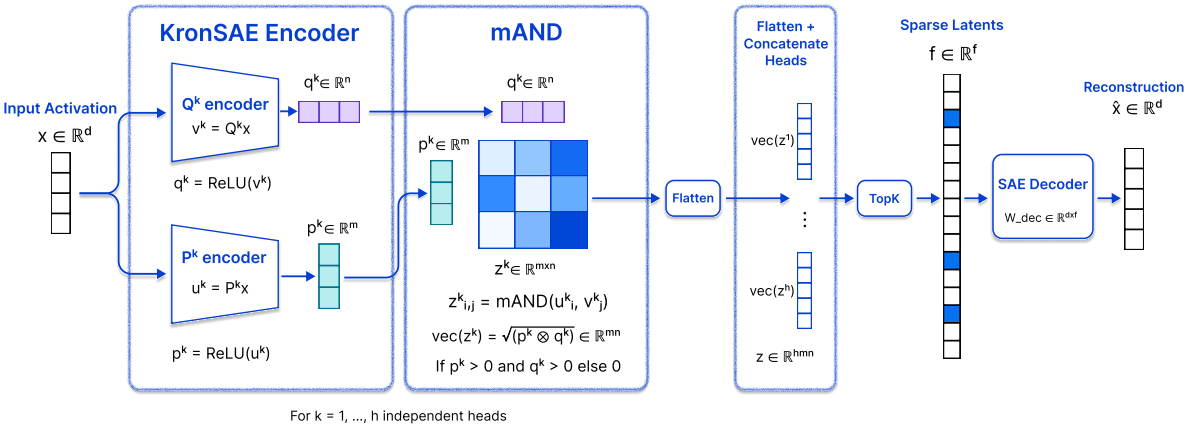

Figure 2: Overview of the proposed KronSAE method. We decompose $x \in \mathbb{R}^d$ into two vectors and then interact via mAND. Sparse latents is combination of features combined by mAND operator.

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, as 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 offer effective reconstruction trade-off induced by the hierarchical encoder – while reducing from **40%** of FLOPs we get less than 1% EV drop.

2 Related Work

Sparse Autoencoders. Early work demonstrated that SAEs can uncover human-interpretable direc-

tions 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

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

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 $m \le n \ll d$ and $F = h\,m\,n$. The *pre-latents*

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

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:

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

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,

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

Concatenating heads and applying TopK yields *post-latents* $\mathbf{f} \in \mathbb{R}^F$. Activation events follow an AND-like rule: pre-TopK support is the intersection of the supports of its two parents, and the final active support is further restricted by global TopK selection. This architectural prior on co-activations induce specific semantical 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 for numerical stability, applied before taking a square root; although it introduces small noise in exact AND rule, it is negligible due to TopK selection and vanishes at the limit case $\epsilon \to 0$.

KronSAE approach is broadly compatible with sparse decoder kernels, MoE-style routing, different loss functions and activations, as we show in the next sections. See additional discussion of method semantics in Appendix D and of computational cost in Appendix B.

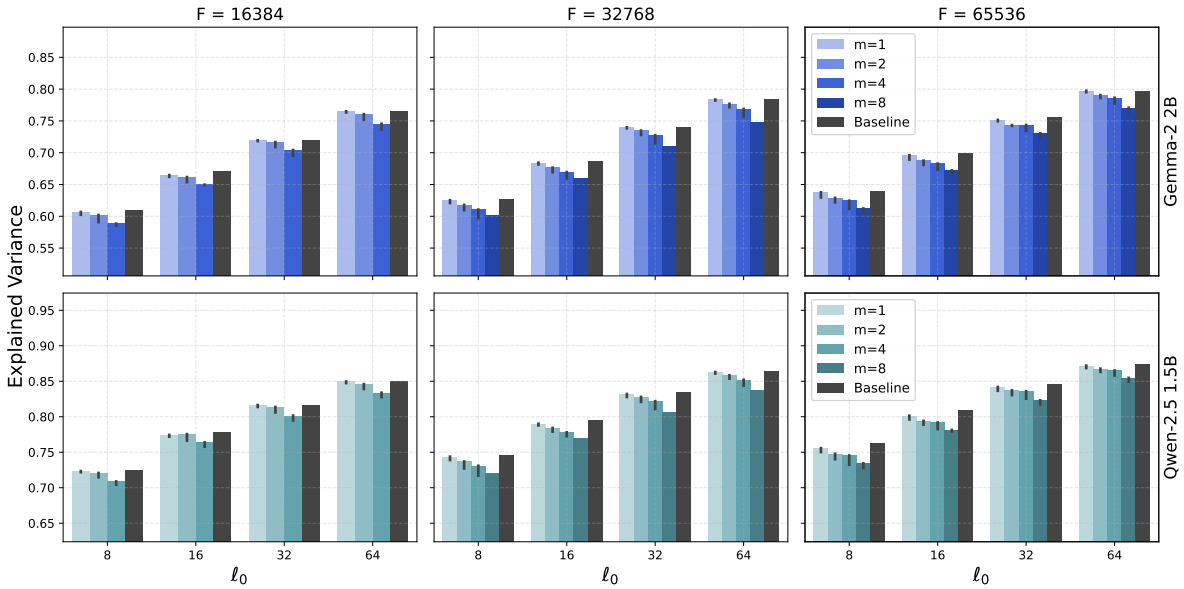

Figure 3: Maximum performance for KronSAE and TopK SAE on Qwen2.5-1.5B and Gemma-2-2B for different dictionary sizes F and 250m tokens. KronSAE with lower number of parameters is on-par with the baseline on EV; the gap narrows with larger dictionary size on layer 12 on both Qwen2.5-1.5B and Gemma2-2B.

4 Experiments

In this section we examine the choice of hyperparameters, induced trade-offs and compatibility of KronSAE with existing SAE variants.

Experimental setup. parWe train SAEs on the residual streams of Qwen-2.5-1.5B-Base (Yang et al., 2024) 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),

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

so that 1.0 is optimal, and use automated interpretability pipeline (Bills et al., 2023; Paulo et al., 2024) to measure the interpretability of learned latents (see Appendix F).

Reconstruction fidelity. We sweep over $m \in \{1, 2, 4, 8\}, h \in \{512, 1024, 2048, 4096\}, n = F/(mn)$, $\ell_0 \in \{8, 16, 32, 64\}$, $F \in \{2^{14}, 2^{15}, 2^{16}\}$. We sweep over early, middle and late layers. Here we evaluate KronSAE against of TopK SAE. Also, we use 250M tokens budget. For most setups with fixed m we report frontier performance across n and h plus dispersion over all configurations. As shown on Figure 3 on equal budget KronSAE perform on-par with less FLOPs on encoder. For layers 6 and 20 conclusions are the same as for layer 12, this showed in Appendix C. We also report MSE for all layers in Appendix C.1.

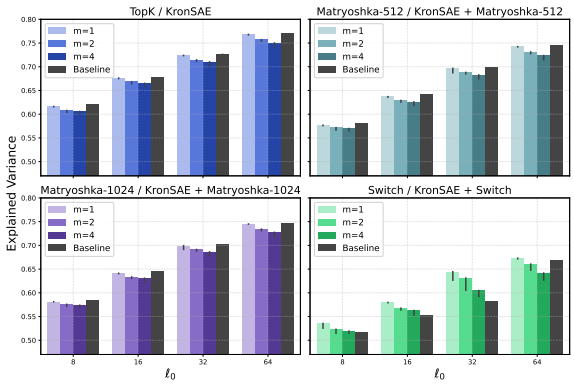

Figure 4: We apply Kron encoder mechanism to TopK, Switch and Matryoshka SAEs. Switch+Kron matches Switch baseline in most configurations and improves it in some; Matryoshka+Kron is on par across the sweep.

Compatibility with other SAEs. The method is compatible with existing SAE variants, and in this paper we test two related techniques. We select 12 layer and Gemma-2-2B model to test this setups. Matryoshka SAE (Bussmann et al., 2025) combined with our encoder improves the interpretability of latents, as shown in Figure 5, and maintains the same quality trade-off of representa-

tion, as shown in Figure 4. For Switch SAE (Mudide et al., 2025), our Kronecker encoder improves quality against the baseline (Figure 4). Matryoshka-512 and Matryoshka-1024 with selected dictionary size have 6 and 5 nested depth correspondingly. Implementation and notation details, JumpReLU and Gated SAEs tests are described in the Appendix A.

Interpretability trade-off. For this analysis we select most common middle layer SAE for both Qwen and Gemma on dictionary size $F = 16384$. We use bigger token budget - 1B tokens. We sweep over different m, n, h to show tradeoff between interpretability metrics, explained variance and kronsae hyperparameters. As plots on Figure 5 shows, we have better results on KronSAE than on TopK SAE with increasing m.

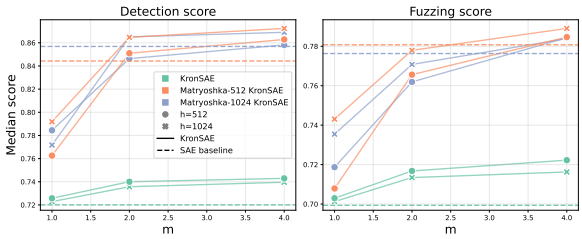

Figure 5: Detection and Fuzzing scores for Baseline SAEs and their Kron versions. For all SAEs we have improvement in scores, with inreasing m. With $m \geq 2$ all modifications match or exceed baselines.

Computational cost. KronSAE also reduces the encoder-side computational, 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 against the TopK SAE baseline in terms of encoder FLOPs (see details in Appendix B) and explained variance. The resulting Pareto curve shows that KronSAE can substantially reduce compute while preserving reconstruction quality.

5 Study of Correlation Learning

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

5.1 Real correlations.

To test whether KronSAE heads capture correlated feature structure, we compare a trained KronSAE against a standard TopK SAE trained on the same model activations. We first compute Pearson correlations between TopK feature activations, and separately compute correlations between each TopK feature and each KronSAE feature. For every TopK feature i, we keep its top-r most correlated matches from KronSAE, 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 reported score on Figure 7 is the fraction of correlated TopK pairs that produce such a same-head hit. This tests whether KronSAE groups correlated features of flat TopK SAE within same heads, and whether it does this more often than expected by chance, which is approximately $1 - (1 - 1/h)^{r^2} \approx r^2/h$ and far below observed rate. Geometric interpretation of this behaviour shown on figure 1.

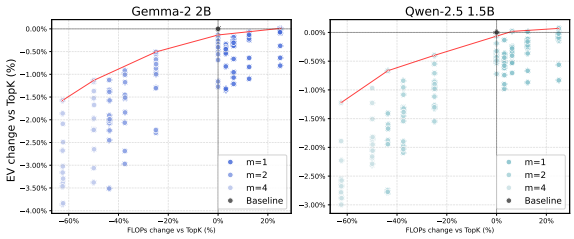

Figure 6: Comparison of TopK SAE with KronSAE in the reconstruction/compute trade-off. We can see that KronSAE offer better trade-off in term of reconstruction quality: while we have less than 1% percent drop, we get more than 40% speedup. See also Appendix B.

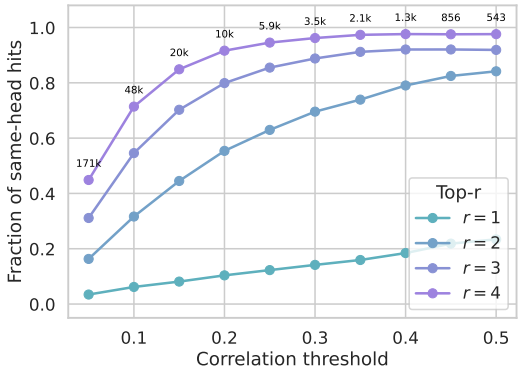

Figure 7: Fraction of correlated TopK feature pairs whose top-r KronSAE matches share at least one Kron head. We select TopK feature pairs with activation correlation above the threshold on the x-axis, map each feature to its top-r correlated KronSAE features, and count a hit if the two candidate sets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

To test co-activation inductive bias of KronSAE, we select KronSAE with $m = 4, n = 4$ and calcu-

late 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 the trained one. To isolate the effect of our initialization procedure, we initialize the weights from the uniform distribution. As shown in Figure 8, correlations are indeed higher within a single head and higher than for random SAE, suggesting that imposed correlated structure works as intended.

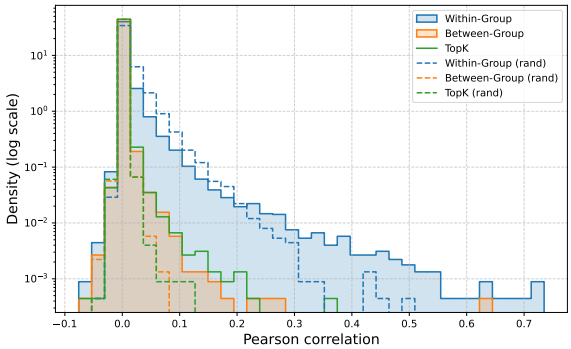

Figure 8: Correlations between features in KronSAE with $m = 4, n = 4$ within a head and with features from other heads. Our design induces higher correlations within a group, which also gets stronger after training, although SAE have also learned correlated features from different heads.

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 inductive bias is expected to align with 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 SAE learned decoder atoms with the AE atoms by solving the quadratic assignment problem (Vogelstein et al., 2015) (see Appendix E for motivation and details), and compute decoder-atom similarity matrix $C_{\text{dec}} = W_{\text{dec}}^T W_{\text{dec}}$.

To quantify how closely SAE atoms mirror the original structure of S, we employ RV coefficient defined as $\mathrm{RV}(A, B) = \mathrm{tr}(A^\top B)/\sqrt{\mathrm{tr}(A^\top A)\,\mathrm{tr}(B^\top B)}$. Results are shown in Figure 9. In our experiments, KronSAE

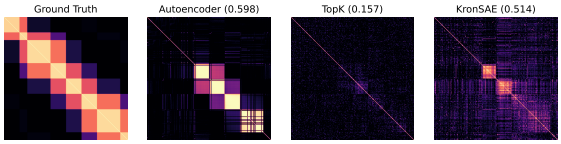

Figure 9: Synthetic block-correlation experiment. Each panel shows a feature-similarity matrix for the target block structure, the bottleneck AE representation, TopK SAE, and KronSAE. Numbers in parentheses are RV coefficients against the target after atom matching. Results show that KronSAE recovers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

achieves higher RV than TopK SAE in this synthetic setting, confirming that the inductive bias is expressive enough to recover the block structure when it is present. See detailed description and additional setups in Appendix E.

6 Analysis

6.1 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 10 reports three absorption metrics measured via 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 with dictionary size $F = 2^{14}$ and compare TopK SAE, Matryoshka SAE (Bussmann et al., 2025), and their KronSAE version, since they impose different hierarchical priors (see Appendix D 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: a mechanism where a post-latent can be active only when both pre-latents are active can reduce full-absorption events by limiting

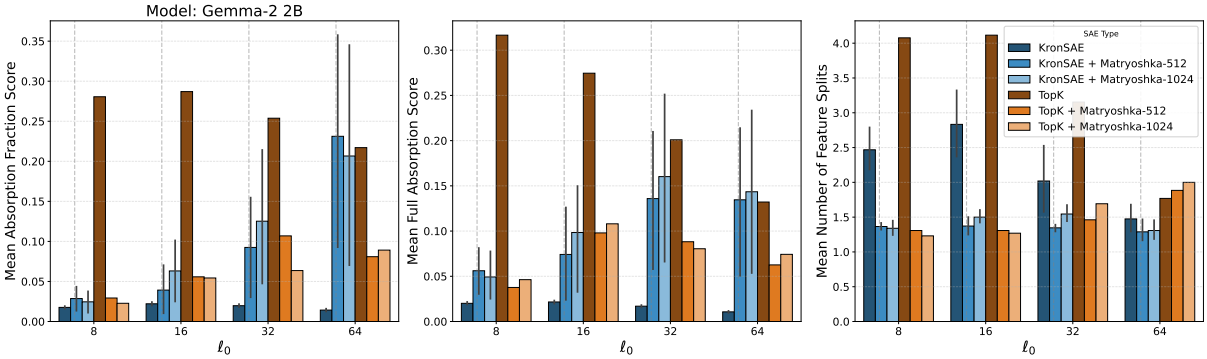

Figure 10: Feature absorption metrics on Gemma-2 2B. KronSAE configurations (various m, n) exhibit lower mean absorption fractions and full-absorption scores across different ℓ_0 and selected baselines.

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 we analyze the properties of groups in KronSAE. We choose the 12th layer of Gemma-2-2B and a dictionary size of 16k features, of which the first 16384 were selected. KronSAE was chosen with $m = 4, n = 4$. We run for 24M tokens total to collect the 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) (see details in Appendix F).

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 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.

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.

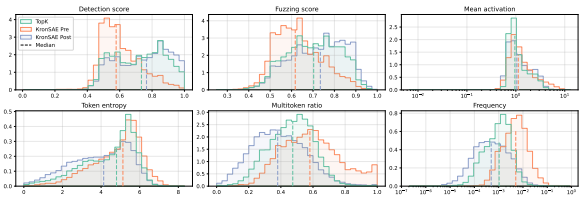

Figure 11: Distribution of properties for TopK SAE and KronSAE ($m = 4, n = 4$) with 16k dictionary size trained on Gemma-2-2B. Pre and Post suffixes denote pre- and post- latents, and y-axis indicate density. Token entropy shows the entropy of the distribution of tokens on which feature has activated, 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.

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%

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
Composition:	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
Composition:	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
Composition:	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
Composition:	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.

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 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. 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 works as specifier. An illustrative example is shown in Table 1: the base contains three detectable sub-semantics, and each extension then retrieves it.

Other types of interaction may occur, such as appearance of completely new semantics, for example composition between base 3 and extension 1 in Table 1, or a frequent case when post-latent inherit semantics of only one parent, or the impact of another parent is not detectable, which 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, so we leave it to further 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 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 compositional SAE, especially in settings with limited compute budget 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 lens on how sparsity and factorization can synergise in representation learning.

8 Limitations.

KronSAE introduces a clear trade-off between interpretability, efficiency, and reconstruction performance. Our evaluation is also limited to mid-sized transformer models. While the overall training and inference infrastructure is largely similar to that of standard SAEs, replacing the ordinary dense encoder layer with the proposed factorized encoder requires some additional implementation work. However, the resulting computation is highly parallelizable and is already reasonably efficient using standard tensor operations, even without specialized kernels. We therefore view this engineering overhead as moderate relative to the interpretability and compute trade-offs offered by KronSAE. Additionally, toy model setup is intended mainly as sanity check for KronSAE inductive bias, since real-world corelation and hierarchical structure may differ; the real-world data experiment on correlations is more direct and practically oriented.

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 Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, and 1 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2.

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, and Joseph Bloom. 2024. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *Preprint*, arXiv:2409.14507.

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 Partovi 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 Henigan, 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*.

Nelson Elhage and 1 others. 2023. Decomposing language models with dictionary learning. *Transformer Circuits*.

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*.

Thomas Heap and 1 others. 2025. Sparse autoencoders can interpret randomly initialized transformers. *arXiv preprint arXiv:2501.17727*.

Dhruva Karkada, Daniel J. Korchinski, Andres Nava, Matthieu Wyart, and Yasaman Bahri. 2026. Symmetry in language statistics shapes the geometry of model representations. *Preprint*, arXiv:2602.15029.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. 2025. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. *Preprint*, arXiv:2503.09532.

Scott C. Lowe and 1 others. 2021. Logical activation functions: Logit-space equivalents of probabilistic boolean operators. *arXiv preprint arXiv:2110.11940*.

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*.

Gonçalo Paulo and Nora Belrose. 2025. Sparse autoencoders trained on the same data learn different features. *Preprint*, arXiv:2501.16615.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. 2024. Automatically interpreting millions of features in large language models. *Preprint*, arXiv:2410.13928.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, 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. *Preprint*, arXiv:2406.17557.

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 Maziarz, 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 Opper, 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.

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.

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.

Peihao Wang, Rameswar Panda, Lucas Torroba Hennigen, Philip Greengard, Leonid Karlinsky, Rogerio 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 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 $F = 2^{14}$, $F = 2^{15}$, and $F = 2^{16}$ 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.

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 $\{P_k, Q_k\}_{k=1}^{h}$. For each head k, we define its decoded rows via additive composition:

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

with $i = 1, \ldots, m,\ j = 1, \ldots, n$. 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$^{\mathbf{k}}$**. Reconstruction loss for Matryoshka is defined as follows:

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

where $\mathbf{f}$ is latent vector; in training, an auxiliary loss $\mathcal{L}_{aux}$ is used as 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.

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 2. For Gated SAEs, the reconstruction performance of Gated-KronSAE is on-par with baseline SAE.

B Computational cost

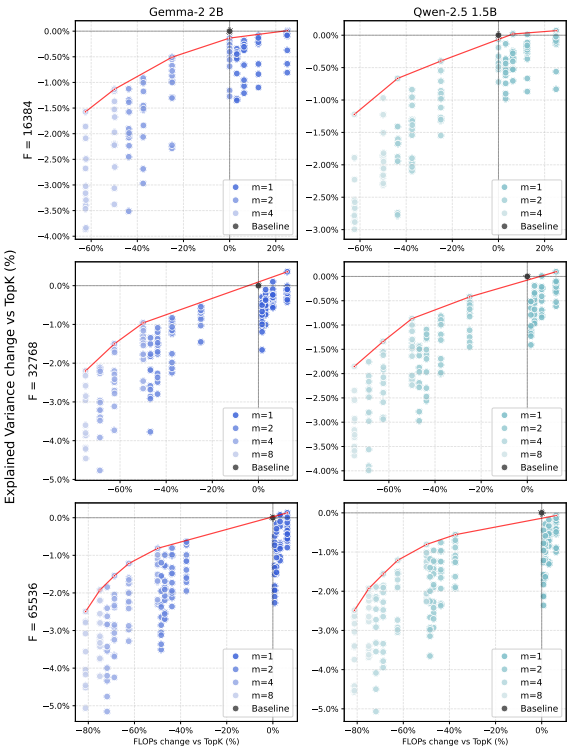

Figure 12: Trade-off between FLOPs and explained variance. In most settings, KronSAE substantially reduces compute with only a small change in EV.

We estimate the per-token floating-point operation (FLOP) cost of the SAE forward pass under

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 ↑
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. 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. **Bottom**: comparison of Gated SAE architecture with its KronSAE version for dictionary $F = 2^{16}$.

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

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

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

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

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.

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.

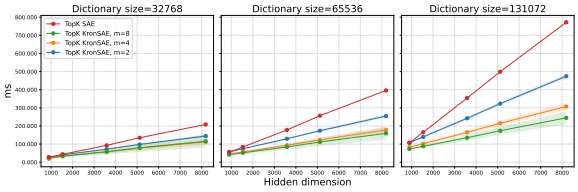

Figure 13: Speed comparison of TopK SAE with KronSAE across different hidden dimensions. We can see that KronSAE have better scaling properties than SAE with default encoder architecture.

C Performance on Different Layers

As showed on Figure 14 and Figure 15 that KronSAE perform on-par with TopK SAE on $m = 1$ setup, while on other setup we have minimum EV change, while reduce much number of encoder parameters.

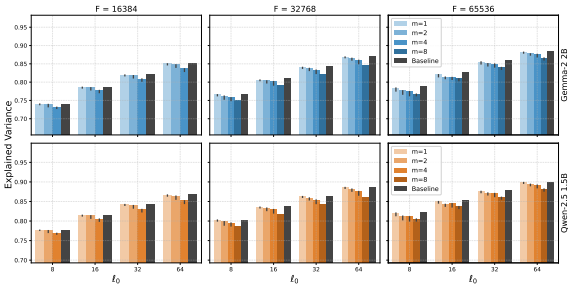

Figure 14: Maximum performance for KronSAE vs.TopK SAE vs. on Qwen-1.5B and Gemma 2 2B for different dictionary sizes F and budgets in equal token setting. KronSAE with lower number of parameters is on-par with the baseline on EV; the gap narrows with larger dictionary size on **layer 6** on both Qwen2.5-1.5B and Gemma2-2B.

C.1 MSE performance.

We also report MSE results for layers 6, 12 and 20 on the 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 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. 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.

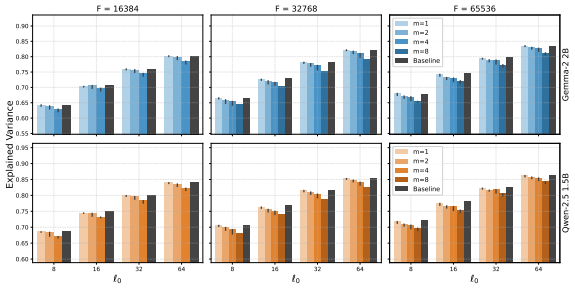

Figure 15: Maximum performance for KronSAE vs.TopK SAE vs. on Qwen-1.5B and Gemma 2 2B for different dictionary sizes F. KronSAE with lower number of parameters is on-par with the baseline on EV; the gap narrows with larger dictionary size on **layer 20** on both Qwen2.5-1.5B and Gemma2-2B.

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, 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:

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, extension pre-latent and post-latent are active. Then it is true that $\mathcal{F} \subseteq \mathcal{P} \cap \mathcal{Q}$, meaning that pre-latents must be broader and polysemous to encode multiple semantics of emitted post-latents. We validate this behaviour qualitatively in section 6.2, although in the same section we describe that apparently other types of interactions also presented (which arise probably due to magnitude exploitation).

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 Sections 4, 6.1, 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.

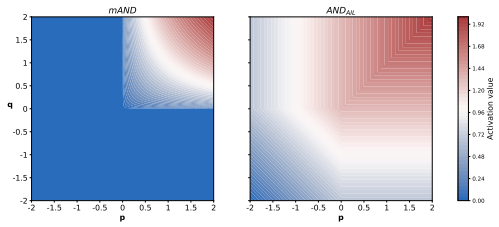

Figure 16: Comparison of the smooth mAND operator against the $\mathrm{AND}_{\mathrm{AIL}}$ (Lowe et al., 2021).

E Synthetic Correlations Experiment

In this section we describe the setup and metrics for toy model studies 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 $\bar{\mathbf{x}}_{\text{sparse}} = L\ \mathrm{TopK}(\mathrm{ReLU}(\mathbf{x}))$, 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):

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

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, and match its latents with the autoencoder latents by solving the quadratic assignment problem (Vogelstein et al., 2015) (described in the next section) and compute the covariance $C_{\text{dec}} = W_{\text{dec}} W_{\text{dec}}^{\top}$.

E.2 Quadratic matching algorithm

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

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 and 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:

$$\begin{aligned} &\max_{\Pi} \|X^T \Pi Y\|_F^2 = \\ &= \max_{\Pi} \operatorname{trace}(\Pi^T X X^T \Pi Y Y^T), \end{aligned} \tag{10}$$

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 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:

$$\begin{aligned} &\max_{\Pi} (X X^T \Pi (Y Y)^T \Pi^T) = \\ &= \max_{\Pi} \operatorname{trace}(\Pi^T X X^T \Pi Y Y^T), \end{aligned} \tag{11}$$

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
        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 employ the RV coefficient, defined as $RV(S, C) = \operatorname{trace}(SC) / \sqrt{\operatorname{trace}(S^2) \operatorname{trace}(C^2)}$. 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 3 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.

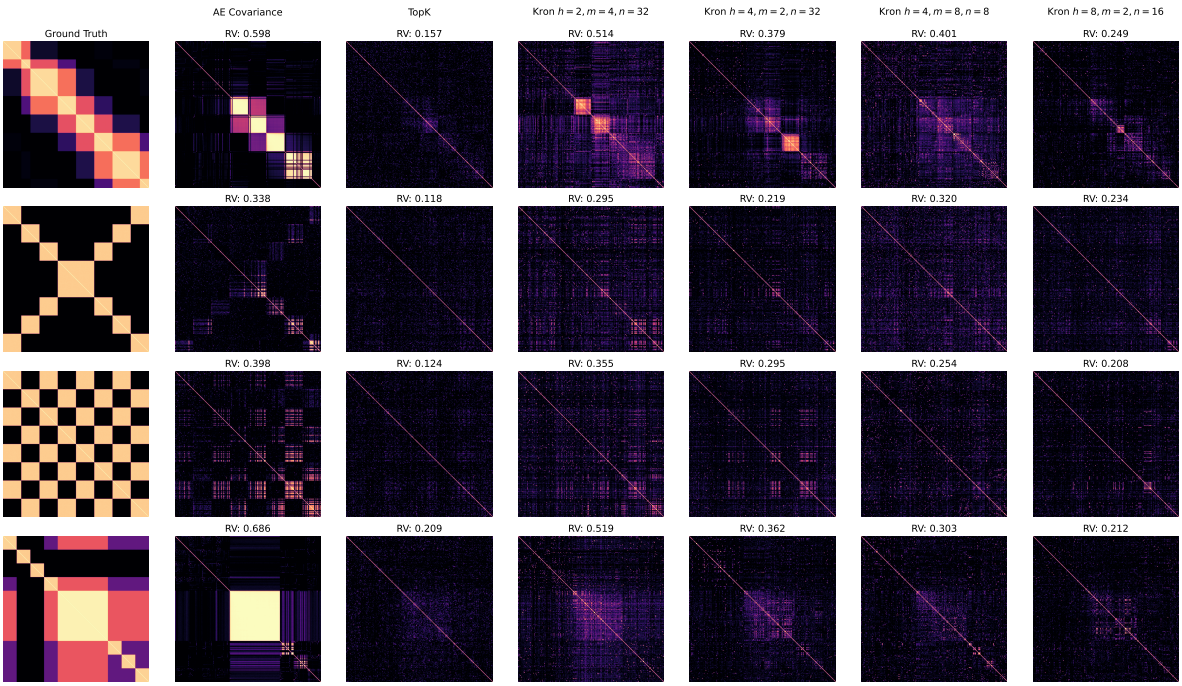

Figure 17: Examples of patterns learned in autoencoder TopK and KronSAE variants after we apply the improved matching scheme. KronSAE have learned patterns that more closely resemble the underlying ground truth structure, and with increasing number of heads (more fine-grained structure) it approaches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

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.

F 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.

F.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 computed per sequence:

$$\text{mt ratio} = \frac{1}{b} \sum_{i=0}^{b} \frac{\text{№ of activations}}{\text{total tokens}}, \tag{13}$$

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.

F.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.

```
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.

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., 2024), with added em-

```
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.

phasis 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: 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:

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

Fuzzing: 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 up to 8 examples in batch and the model identifies correctness of the labeling, with scoring as in Equation 14.

G 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.

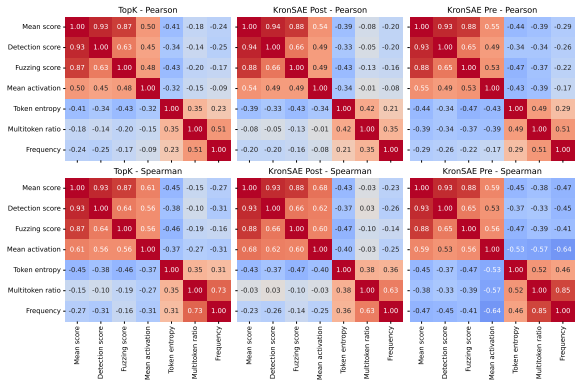

Figure 20: Correlation coefficients (Pearson and Spearman) between properties of TopK and KronSAE latents. Token entropy emerges as a strong predictor of interpretability scores, while higher mean activation and lower frequency also indicate more interpretable features.

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.

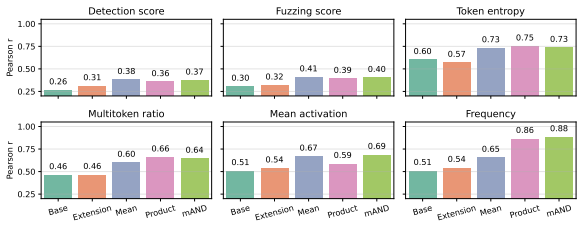

Figure 21: Correlation patterns between properties of post-latents and pre-latents.

Basis geometry. 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.

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

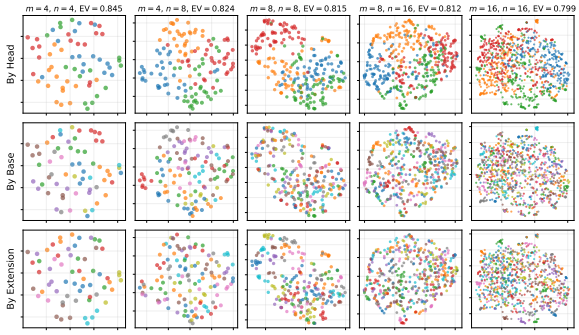

Figure 22: UMAP visualization of post-latent clustering patterns by head, base, and extension group membership. We observe tight clusters by base for $m < n$ and by extension for $m \geq n$.

This asymmetry suggests that pre-latent capacity requirements directly 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.

Interpretability tradeoff. As shown in Section 4 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).

H Analysis of Compositional Structure

Head 3. For this head we have selected all base elements and extension 2, shown in Table 4. 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 medical semantics while preserving metric/number aspects) and base 3 (combining instrument semantics with necessity to yield therapy/treatment concepts). 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. 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.

Head 136. This head exhibits higher interpretability in post-latents than pre-latents. Key observations from the Table 5 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 6 emonstrates more consistent AND-like behavior than Heads 3 and 136, closely matching the interaction pattern shown in Figure 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
Composition:	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
Composition:	Numerical digits and decimal points within quantitative values.	0.86
Base 2	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	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 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
Composition:	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
Composition:	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
Composition:	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
Composition:	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.

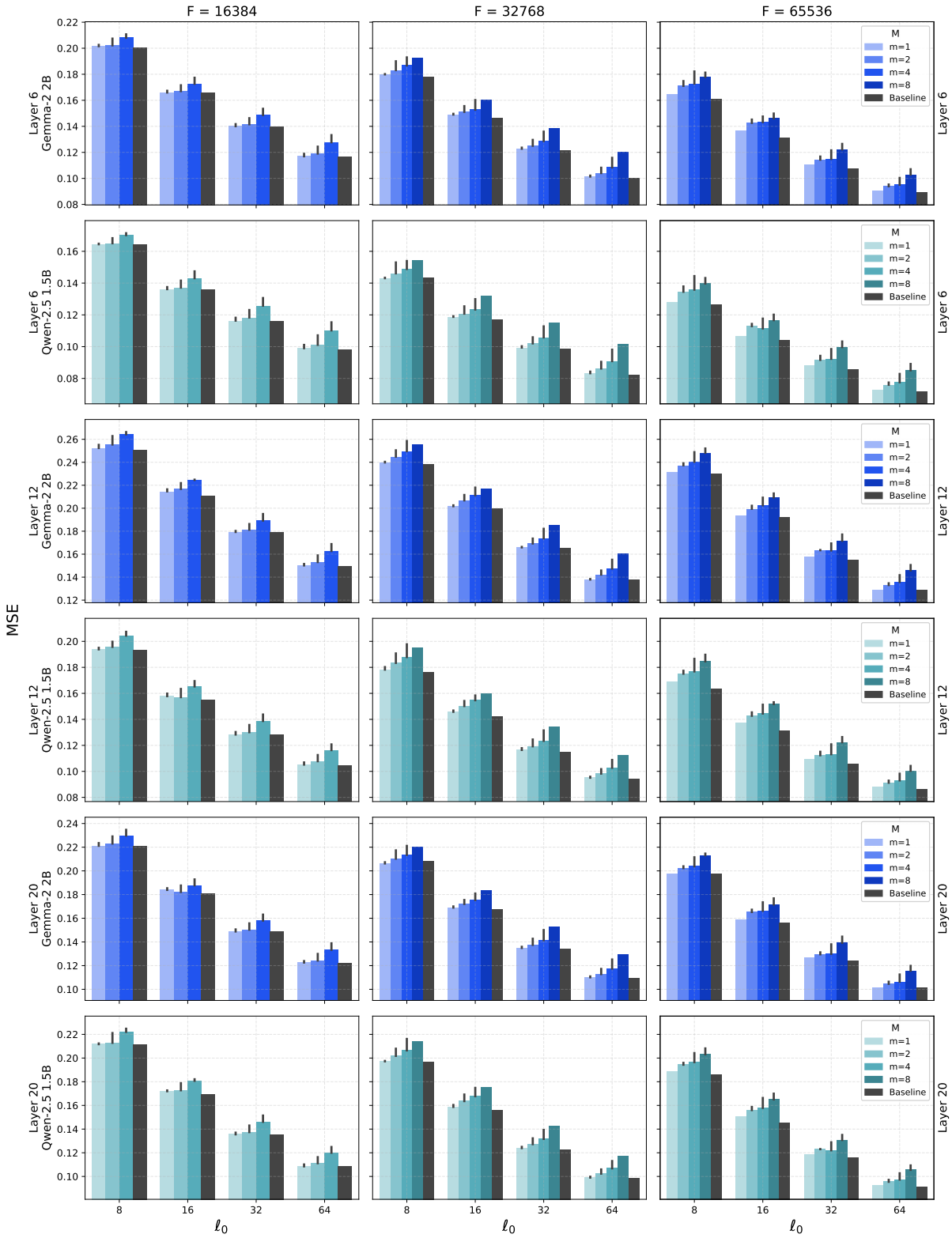

Figure 23: MSE-based comparison of KronSAE and TopK SAE under a fixed token budget.

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
Composition:	Geographical or spatial references using the term "map".	0.93
Base 1	Emphasis on terms indicating feasibility and organizations.	0.80
Composition:	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
Composition:	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
Composition:	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

```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()
        var = (x - x.mean(0)).pow(2).mean()
        ev = 1 - loss / var
        return loss, ev
```