fmxcoders: Factorized Masked Crosscoders for Cross-Layer Feature Discovery

Andreas D. Demou[1] **Panagiotis Koromilas**[1,2] **James Oldfield**[3]
Yannis Panagakis[2,4] **Mihalis A. Nicolaou**[5,1]

[1]The Cyprus Institute [2]University of Athens [3]University of Oxford
[4]Archimedes AI/Athena Research Center [5]University of Cyprus

github.com/AndreasDemou/fmxcoders

Abstract

Many features in pretrained Transformers span multiple layers: they emerge through stages of inference, persist in the residual stream, or are built jointly by parallel MLPs. Crosscoders (namely, sparse dictionaries trained jointly across layers) aim to recover these cross-layer features in a single shared latent space. We show that standard crosscoders largely fail at this purpose. Although their decoder weight norms spread evenly across layers, a functional coherence metric we introduce reveals that each latent's activation is effectively driven by only one or two layers on average. While functionally coherent latents act as human-interpretable concept detectors (e.g., US states and cities), the layer-localized latents that crosscoders predominantly learn collapse onto surface-level patterns such as digit detectors. We trace this failure to two structural limitations: unconstrained cross-layer parameterization and unregularized cross-layer dependence. We address both by introducing `fmxcoders`, which (i) replace the encoder and decoder with low-rank tensor factorizations that draw every latent's per-layer weights from a shared cross-layer basis, and (ii) apply stochastic layer masking, a denoising regularizer along the layer axis that penalizes latents whose contribution collapses when a single layer is masked. Across GPT2-Small, Pythia-410M, Pythia-1.4B, and Gemma2-2B, `fmxcoders` lift mean probing F1 by 10–30 points, surpassing per-layer SAE baselines that standard crosscoders fail to reach, reduce reconstruction MSE by 25–50%, and roughly double mean functional coherence. An LLM-as-a-judge evaluation further shows that `fmxcoders` recover 2.6–14$\times$ more semantic latents than standard crosscoders across all four base LLMs. Finally, steering experiments evidence the causal effect of the discovered features, both in generic and targeted contexts.

1 Introduction

The principal aim of mechanistic interpretability is the decomposition of network activations into human-interpretable features and the identification of circuits that connect these features [1, 2]. Focusing on the former, the dominant method for feature detection is sparse dictionary learning, which trains an encoder to map activations into a sparse code and a decoder to reconstruct them [3–9]. Each coordinate of the sparse code is referred to as a *latent*, and corresponds to a learned dictionary direction whose activation is interpreted as the presence of a feature in the input.

A deep network gives rise to two kinds of features that dictionary learning aims to recover: *local* features, tied to a specific layer or a narrow band of adjacent layers, and *global* features, built jointly by several layers or persisting across many. Both are well documented in the literature. On the local side, knowledge neurons store specific factual associations at identifiable layers [10]. At a

Preprint.

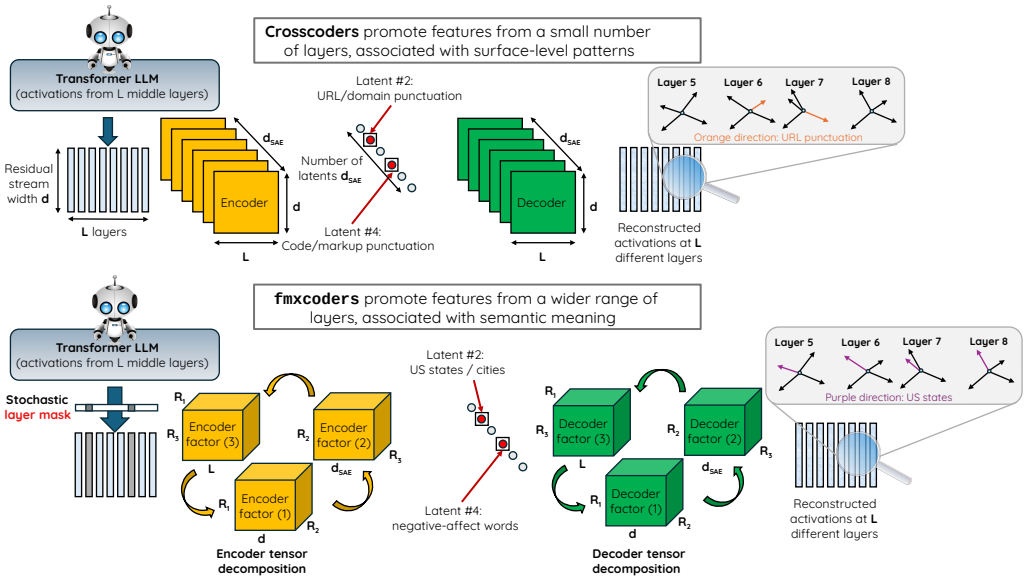

Figure 1: Standard crosscoders (**top**) parameterize the encoder and decoder as L independent per-layer matrices and recover latents whose functional dependence collapses onto a few layers, capturing surface-level patterns. `fmxcoders` (**bottom**) replace these with low-rank tensor factorizations that share a cross-layer basis across all latents, and add stochastic layer masking as a denoising regularizer along the layer axis, producing latents that span more layers and act as semantic concept detectors. Insets show per-layer decoder directions for one example latent.

coarser scale, transformer inference appears to unfold in universal stages (detokenization, feature engineering, prediction ensembling, and residual sharpening) each tied to a range of layers [11]. On the global side, the residual-stream view holds that transformer layers incrementally update a shared hidden state across the network [12]. Universal neurons recur with similar activation patterns across different LLMs [13]. Per-layer sparse autoencoder (SAE) analyses reveal features that activate with near-identical directions across many consecutive layers [14, 15], as well as features built by multiple MLPs acting in parallel on the residual stream, forming *cross-layer superposition* [16].

SAEs are able to properly recover local features [3, 4] by tying a dictionary to a single layer. Crosscoders [16, 17] extend this to attain global features by learning a single dictionary whose latent neurons read from and write to multiple layers at once. Although crosscoder weight norms spread roughly evenly across layers [16, 18], we show that the functional dependence of each latent in fact collapses onto one or two on average. To measure this gap, we introduce a *coherence* function, defined as the ratio of the sum of per-layer dependence to its maximum per-layer dependence (see Section 4.2), bounded between 1 (all dependence on one layer) and L (equal across all L layers). Two choices of defining this per-layer dependence give two metrics: *norm coherence*, which uses the decoder weight norm at each layer (a generalization of typical crosscoder diagnostics [16, 18]), and *functional coherence*, which zeros all activations of a specific layer at the crosscoder's input and measures the resulting change in the latent's activation. We validate that, as already known in the literature [16, 18], norm coherence is high (> 7), but this is misleading since functional coherence is low (~ 2) and thus crosscoder latents model information from just two layers on average.

This divergence reflects a structural failure of the standard crosscoder to learn cross-layer features which we trace to two concrete limitations:

Limitation 1 (L1): Unconstrained cross-layer structure. The encoder and decoder are parameterized as L independent per-layer matrices with no structural link across layers or across the latent population. The reconstruction loss (Equation (2)) decomposes additively across layers and, as a result, the objective is invariant to reallocations of a latent's weight mass that preserve per-layer reconstructions.

Limitation 2 (L2): Unregularized cross-layer dependence. Even with a constrained structure, the reconstruction objective places no penalty on how a latent's activation depends on the different layers, so a latent can still collapse its functional dependence onto a single dominant layer at no cost.

We address these limitations by introducing `fmxcoders`, with two complementary modifications. First, `fmxcoders` constrain the cross-layer structure by stacking the per-layer decoder/encoder matrices into a 3D tensor and factorizing it [19], directly addressing **L1**. The layer axis of the input passes through a shared low-dimensional space rather than L independent matrices, so each latent's engagement across layers is drawn from a structured family shared across all latents. Second, `fmxcoders` regularize the cross-layer dependence with *stochastic layer masking*, directly addressing **L2**. During training, the activations from specific layers are randomly masked with a probability p, encouraging redundant codes that distribute features across channels, similar to coding theory [20, 21]. In this setting, latents supported by a single layer pay a reconstruction cost growing with p, while latents distributed across layers do not, so the optimization pressure favors cross-layer support.

Our contributions are summarized below:

C1. By introducing a functional coherence metric we show that **standard crosscoders concentrate functional dependence on two layers on average**, despite spreading their weight norms uniformly. This pathology is invisible to the norm-based diagnostics used in prior works.

C2. We identify two architectural limitations as the sources of the pathology described in **C1**, **unconstrained cross-layer structure** (**L1**) and **unregularized cross-layer dependence** (**L2**).

C3. To address these limitations we propose `fmxcoders`, a crosscoder variant combining two complementary modifications: (i) **low-rank tensor decomposition** of the encoder and decoder weights, which couple per-layer parameters through a shared cross-layer basis, and (ii) **stochastic layer masking**, which acts as a regularizer along the layer axis and pressures latents to spread their support across multiple layers.

C4. Across GPT2-Small, Pythia-410M, Pythia-1.4B, and Gemma2-2B, the combined method **raises probing F1** above the per-layer SAE baseline that the standard crosscoder fails to reach by large margins, **reduces reconstruction MSE** by 25–50%, and roughly **doubles average functional coherence**.

C5. A qualitative inspection together with an LLM-as-a-judge evaluation shows that `fmxcoders` **recover** 2.6**–**$14\times$ **more semantic latents** than the standard crosscoders across all LLMs, with high-coherence latents acting as concept detectors (US states, negative-affect words, etc.) and low-coherence latents collapsing onto surface-level patterns (digits, punctuation, etc.). The discovered features exhibit a causal effect on the model output in corresponding steering experiments.

2 Related Work

Sparse dictionary learning for interpretability. Sparse autoencoders have become the standard tool for decomposing transformer activations into monosemantic features [3–5], with variants such as TopK [6], JumpReLU [7], and BatchTopK [8] refining the sparsification operation. SAEs learn a feature dictionary at a single location in the transformer, and training multiple SAEs across different transformer layers forms the basis of circuit discovery [22–24].

Multi-layer and cross-layer methods. Several approaches extend dictionary learning across layers. Multi-layer SAEs [14] and feature-matching procedures [15] align per-layer dictionaries post hoc, while crosscoders [16] train a single shared dictionary over all layers and provide the basis for our work. They have been used to study feature evolution within a single LLM [17, 25, 26], and to track features across base-finetune LLM pairs or training checkpoints [27–29]. The crosscoder encoders and decoders remain unconstrained, which [18] flagged in the model-axis setting as producing artifactual latents. We identify the layer-axis analogue of this failure, show it persists despite high dictionary norms across the layers, and address it by applying tensor factorization and layer masking.

Tensor factorization in deep networks. Tensor decompositions including Canonical Polyadic [30] (CP) and Tensor Ring [31] (TR) have a long history of compressing weight tensors and inducing structured priors in deep models [19]. In interpretability research, tensor decompositions have been applied on the mixture-of-experts paradigm to provide scalable expert specialization [32] and faithful approximation to multilayer perceptron (MLP) components [33], as well as to model quadratic and

cubic feature interactions in polynomial SAEs [9]. In the present study, tensor decompositions are repurposed to provide a shared representation space across the latent population and across layers, addressing the structural limitation identified in standard crosscoders.

Denoising and structured masking. Denoising autoencoders [34, 35] corrupt inputs and require reconstruction of clean targets to learn robust and distributed representations; an argument that was formalized via coding theory [20]. The result is redundancy in information encoding so that masking some still leaves recoverable signal. The principle has scaled to masked language modeling [36], masked image modeling [37], and feature-level dropout [21], all of which corrupt at the input or feature level. In our work, we apply the same principles along the layer axis where the corruption pattern correlates with the failure mode being targeted, namely layer-specialized latents.

3 Factorized Masked Crosscoders

3.1 Preliminaries

Notation. Vectors and matrices are denoted by lowercase and uppercase bold letters respectively, $\boldsymbol{u}$, $\boldsymbol{U}$ and tensors are represented by uppercase bold upright letters $\mathbf{U}$. An element (scalar) of a matrix or tensor is accessed by subscript, *e.g.* $\mathbf{U}_{i,j,k}$. Mode-n fibers (the generalization of rows and columns from matrices to tensors) are obtained by fixing all indices except the one that corresponds to the nth mode, *e.g.* a mode-1 fiber of $\mathbf{U}$ is $\mathbf{U}_{:,j,k}$. Similarly, tensor slices are obtained by fixing a single index, *e.g.* $\mathbf{U}_{i,:,:}$. The i-th column of $\boldsymbol{M}$ is $\boldsymbol{m}_i$, and $\boldsymbol{M}_{:,1:r}$ denotes its first r columns. For depth stacking, *i.e.* combining matrices as slices of a tensor along a new dimension, we use $[\boldsymbol{X}, \boldsymbol{Y}] \in \mathbb{R}^{d_1 \times d_2 \times 2}$ for $X, Y \in \mathbb{R}^{d_1 \times d_2}$. The symbol $\circ$ denotes the vector outer product, so $\boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$ is the rank-one tensor with entries $a_i b_j c_k$. $\mathbb{R}^d$ and $\mathbb{R}^{d_{\text{sae}}}$ denote the activation and sparse-code spaces, with $d_{\text{sae}} \gg d$.

Sparse Autoencoders. SAEs build on overcomplete dictionary learning [38, 6] to decompose neural activations into a sparse set of latent features. Given activations $\boldsymbol{x} \in \mathbb{R}^d$ from an intermediate layer of a pretrained network, an SAE learns a sparse code $\boldsymbol{z} \in \mathbb{R}^{d_{\text{sae}}}$ via $\hat{\boldsymbol{x}} = \boldsymbol{b}^{\text{dec}} + \boldsymbol{D}\boldsymbol{z}$, $\boldsymbol{z} = \mathcal{S}\big(\text{ReLU}(\boldsymbol{E}^\top \boldsymbol{x} + \boldsymbol{b}^{\text{enc}})\big)$, where $\boldsymbol{E} \in \mathbb{R}^{d \times d_{\text{sae}}}$ is a linear encoder, $\boldsymbol{D} \in \mathbb{R}^{d \times d_{\text{sae}}}$ is the decoder dictionary, $\boldsymbol{b}^{\text{enc}} \in \mathbb{R}^{d_{\text{sae}}}$ and $\boldsymbol{b}^{\text{dec}} \in \mathbb{R}^d$ are the encoder and decoder bias terms, $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ are the reconstructed activations, and $\mathcal{S}$ is a sparsification operator such as Top$K$ [6] or BatchTop$K$ [8]. Feature i activation depends on the extent to which $\boldsymbol{x}$ aligns with the encoder direction $\boldsymbol{e}_i$ and, in the linear regime, contributes by $z_i \boldsymbol{d}_i$ to the reconstruction.

Crosscoders. A crosscoder [16] applies sparse dictionary learning to jointly encode activations from *multiple* layers into a shared sparse latent space and use it to reconstruct all layers. For a given input token, let $\boldsymbol{x}_\ell \in \mathbb{R}^d$ denote the activation at layer $\ell \in [1, L]$, where L is the number of transformer layers considered. The crosscoder assigns encoders $\boldsymbol{E}_\ell \in \mathbb{R}^{d \times d_{\text{sae}}}$ and decoders $\boldsymbol{D}_\ell \in \mathbb{R}^{d \times d_{\text{sae}}}$ to each layer to reconstruct the activations of each layer by computing,

$$\boldsymbol{z} = \mathcal{S}\left(\text{ReLU}\left(\sum_{\ell \in [1,L]} \boldsymbol{E}_\ell^\top \boldsymbol{x}_\ell + \boldsymbol{b}^{\text{enc}}\right)\right), \qquad \hat{\boldsymbol{x}}_\ell = \boldsymbol{D}_\ell \, \boldsymbol{z} + \boldsymbol{b}_\ell^{\text{dec}}, \tag{1}$$

where $\boldsymbol{b}^{\text{enc}} \in \mathbb{R}^{d_{\text{sae}}}$ is a pre-activation bias shared across layers, $\boldsymbol{b}_\ell^{\text{dec}} \in \mathbb{R}^d$ is a per-layer decoder bias, $\mathcal{S}$ is a sparsification operator (e.g. Top$K$ or BatchTop$K$), and $\hat{\boldsymbol{x}}_\ell$ is the reconstruction at layer ℓ produced by the encoder–decoder map. The crosscoder is trained to minimize the expected mean squared reconstruction error across all layers,

$$\mathcal{L}_{\text{recon}} \;=\; \frac{1}{L} \sum_{\ell \in [1,L]} \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}} \Big[\big\| \boldsymbol{x}_\ell - \hat{\boldsymbol{x}}_\ell(\boldsymbol{x}) \big\|_2^2 \Big], \tag{2}$$

where $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L)$ is the tuple of per-layer activations for a token, and $\mathcal{D}$ is the joint distribution over such tuples induced by passing tokens from the training corpus through the base LLM. In practice the expectation is approximated by the empirical mean over a minibatch of tokens.

3.2 Tensor Factorization of Crosscoder Weights

Depth-stacking the per-layer encoders expresses the total encoder as a tensor $\mathbf{E} = [\boldsymbol{E}_1, \ldots, \boldsymbol{E}_L] \in \mathbb{R}^{d \times d_{\text{sae}} \times L}$, and the decoder as $\mathbf{D} = [\boldsymbol{D}_1, \ldots, \boldsymbol{D}_L] \in \mathbb{R}^{d \times d_{\text{sae}} \times L}$. In this setting, the mode-1 fiber $\mathbf{E}_{:,i,\ell} \in \mathbb{R}^d$ is the encoder direction for feature i at layer ℓ, and $\mathbf{D}_{:,i,\ell}$ its decoder atom.

This tensor view makes the standard crosscoder's structural assumption explicit: the L different frontal slices $\mathbf{E}_{:,:,\ell}$ are parameterized independently, with no mechanism coupling a feature's behavior across layers. This independence is precisely the unconstrained cross-layer structure flagged as **L1** in Section 1, and it manifests in two ways. First, *representational fragmentation*: together with a reconstruction loss (Equation (2)) that scores each layer separately, nothing pushes latent i to represent the same feature at layer ℓ and layer ℓ', so the shared latent space does not yield global features but only places layer-local features in a common coordinate system. Second, *parameter redundancy*: at $\mathcal{O}(d_{\text{sae}} \cdot d \cdot L)$ parameters, the crosscoder costs as much as L stacked SAEs and pays to re-discover cross-layer features at every layer.

We observe that both issues admit a shared fix. Inducing structure across layers resolves the fragmentation, and constraining how the three modes interact eliminates the redundancy. This choice has both empirical and theoretical precedent. Empirically, structural intervention has been shown to resolve a different crosscoder pathology, artifactual latents along the model axis in model-diffing settings [18] between a base and a fine-tuned model. While the underlying inductive bias of low-rank coupling is shared, the layer-axis failure we identify is distinct: it concerns functional dependence collapsing within a single model with many layers rather than spurious latents arising between two models, and it requires complementary regularization (Section 3.3) that does not apply in the model-axis case. Theoretically, low-rank tensor factorizations are a standard inductive bias for multi-task settings with shared latent structure across tasks [39, 40]: independent per-task parameterization fails to exploit this structure even when it is present, while factorizing along the task axis biases learning toward recovering the shared representation. In our setting, each layer must reconstruct its own activations from a representation shared with the other layers, making this inductive bias directly applicable.

Concretely, we replace $\mathbf{E}$ and $\mathbf{D}$ with low-rank tensor decompositions that jointly factorize all three modes, forcing the input through a low-dimensional bottleneck. The main results in this study are presented using the TR decomposition because it is suited for tensors with large differences between dimensions. We also report results using the CP decomposition in Appendix B.

Tensor Ring Decomposition. The Tensor Ring (TR) decomposition [31] represents each element of the tensor as the trace of a product of matrix slices drawn from three factor tensors:

$$\mathbf{E}_{j,i,\ell} \;=\; \operatorname{Tr}\big(\mathbf{G}^{(1)}_{:,j,:}\, \mathbf{G}^{(2)}_{:,i,:}\, \mathbf{G}^{(3)}_{:,\ell,:}\big), \tag{3}$$

where $\mathbf{G}^{(1)} \in \mathbb{R}^{R_1 \times d \times R_2}$, $\mathbf{G}^{(2)} \in \mathbb{R}^{R_2 \times d_{\text{sae}} \times R_3}$, $\mathbf{G}^{(3)} \in \mathbb{R}^{R_3 \times L \times R_1}$.

TR couples the three modes through three independent dimensions, the decomposition ranks (R_1, R_2, R_3), one per mode boundary in the ring. Each mode contributes a matrix and these matrices compose multiplicatively: the contribution of layer ℓ to $\mathbf{E}_{j,i,\ell}$ is also affected by the "feature" slice $\mathbf{G}^{(2)}_{:,i,:} \in \mathbb{R}^{R_2 \times R_3}$, so the same layer pattern can act differently depending on which feature it represents. The inductive bias is therefore *context-dependent coupling*: features are no longer confined to a single shared basis of layer patterns but participate in subspaces that vary with feature identity, and decomposition ranks (R_1, R_2, R_3) control expressiveness separately at each mode boundary. Tensor factorization restricts the cross-layer parameterizations of $\mathbf{E}$ and $\mathbf{D}$ (addressing **L1**) without biasing the optimizer toward solutions that spread features across layers (**L2**).

3.3 Stochastic Layer Masking

Within the constrained encoder/decoder parametrizations, a latent can still concentrate all its functional dependence at a single layer, as nothing in the reconstruction objective of Equation (2) penalizes such a configuration. To bias learning toward the cross-layer regime, we complement the architectural constraint with an input-level perturbation, *stochastic layer masking*.

For each input token and each layer ℓ, we independently draw a Bernoulli mask $m_\ell \in \{0, 1\}$ with $\Pr(m_\ell = 0) = p$, and the encoder receives the corrupted activations $\tilde{\boldsymbol{x}}_\ell \;=\; m_\ell \cdot \boldsymbol{x}_\ell$. The reconstruction loss targets the unmasked activations $\boldsymbol{x}_\ell$, meaning that the crosscoder is trained to

reconstruct every layer, including the masked ones. This places our objective in the denoising autoencoder framework [34, 35] (corrupting inputs and require reconstruction of clean targets), instantiated with layer-structured masking noise, a setting in which input noise acts as an implicit Tikhonov regularizer on the encoder-decoder mapping [41]. The masking probability p directly controls how strongly the regularizer suppresses localized features. Larger p makes the cross-layer pressure dominate while smaller p lets localized features survive.

A complementary perspective from coding theory shows that information traversing a sequence of channels retains capacity only when the code is redundant, and that networks trained under stochastic neuron dropout learn precisely such codes, distributing each feature across multiple units rather than localizing it [20]. Stochastic layer masking applies the same principle along the layer axis: a single-layer latent pays a reconstruction cost that grows with p, mirroring the redundancy induced by unit-level dropout [21]. This contrasts with the windowed "convolutional" crosscoder of [16], which architecturally restricts each latent to $K < L$ contiguous layers, imposing a hard locality constraint, opposite in effect to our stochastic regularizer.

4 Experiments

4.1 Experimental Setup

The training procedure for `fmxcoders` was implemented in `SAELens` [42], with all implementation details provided in Appendix A. The results presented in this study refer to crosscoders with $16{,}384$ latents trained on four pretrained base LLMs of different scales: GPT2-Small [43], Pythia-410M and Pythia-1.4B [44], and Gemma2-2B [45]. Activations were collected from the residual stream at eight post-MLP locations spanning the middle layers of each base LLM. All crosscoders were trained using a ReLU pre-activation, followed by a BatchTop$K$ [8] sparsifier with $K = 64$. Gemma2-2B and GPT2-Small activations were generated using OpenWebText [46], while the activations from the Pythia models were generated using an uncopyrighted variant of the deduplicated Pile [47]. Training runs used 500M tokens (300M for GPT2-Small) with a context length of 128. The evaluations that follow compare `fmxcoders` with and without layer masking against base SAEs and standard crosscoders trained with the same hyperparameters. In all main-text experiments, `fmxcoders` are matched to the standard crosscoder's parameter count by appropriate choice of decomposition ranks. Rank and masking probability ablations are presented in Appendix C, while ablations for other architectural components are reported in Appendix D.

4.2 Coherence Diagnostics

To assess whether a crosscoder latent neuron represents a genuinely multi-layer feature or merely responds to information from a single dominant layer, we introduce two complementary coherence metrics: *norm coherence* and *functional coherence*.

Norm Coherence. The simplest approach to measuring layer distribution is to examine the crosscoder's weights directly, which is a generalized version of the *relative decoder norms* shown in [16], suitable for any number of layers. For a crosscoder with decoder weights $\mathbf{D} \in \mathbb{R}^{d \times d_{\text{sae}} \times L}$, we compute the L2 norm for each latent's dictionary entry corresponding to each layer $\ell \in [1, L]$ via the decoder's mode-1 fibers $\boldsymbol{N}_{i,\ell} = \|\mathbf{D}_{:,i,\ell}\|_2$. The *norm coherence* of latent i is then

$$c_i^{\text{n}} = \frac{\sum_{\ell \in [1,L]} \boldsymbol{N}_{i,\ell}}{\max_{\ell \in [1,L]} \boldsymbol{N}_{i,\ell}}, \tag{4}$$

which ranges from 1 (all weight mass concentrated in one layer) to L (equal norms across all layers).

Functional Coherence. A more informative measure examines functional dependence, focusing on how a latent's activation changes when information from a specific layer is removed. For an input $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L) \sim \mathcal{D}$, let M_ℓ denote the operator that zeros the ℓ-th component of $\boldsymbol{x}$, $M_\ell(\boldsymbol{x}) = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\ell-1}, \mathbf{0}, \boldsymbol{x}_{\ell+1}, \ldots, \boldsymbol{x}_L)$, so that $z_i(M_\ell(\boldsymbol{x}))$ is the latent activation when layer ℓ is masked while all other layers remain intact. We define latent i's functional coherence c_i^{f} as

$$c_i^{\text{f}} \;=\; \frac{\sum_{\ell \in [1,L]} \boldsymbol{S}_i^\ell}{\max_{\ell \in [1,L]} \boldsymbol{S}_i^\ell}, \qquad \text{where} \quad \boldsymbol{S}_i^\ell \;=\; \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}} \left[\left| \frac{z_i(\boldsymbol{x}) - z_i(M_\ell(\boldsymbol{x}))}{z_i(\boldsymbol{x}) + \epsilon} \right| \right]. \tag{5}$$

$\boldsymbol{S}_i^\ell$ represents the relative change of latent i's activation with respect to masking layer ℓ, with $0 < \epsilon \ll 1$ a small constant for numerical stability. A high value of $\boldsymbol{S}_i^\ell$ indicates that latent i is sensitive to information from layer ℓ, while a low value indicates that layer ℓ contributes little to that latent's activation. c_i^{f} therefore, measures the ratio of the total relative change across layers to the largest single-layer relative change. Analogously to norm coherence, c_i^{f} ranges from 1 to L.

Figure 2 shows the latent distributions of norm and functional coherence for crosscoder and `fmxcoder` variants trained on Pythia-410M. Norm coherence remains high (avg. $c^{\mathrm{n}} > 7$) in all cases, while functional coherence is much lower (avg. $c^{\mathrm{f}} \approx 2$), except for `fmxcoder` with p=0.05, where it doubles. Factorization alone therefore does not increase functional coherence; *the gain emerges only when factorization and masking are combined*. This interaction reflects the complementary roles of the two interventions. In a standard (unfactorized) crosscoder, the per-layer dictionary entries (encoder/decoder mode-1 fibers) of each latent are independent parameters. Layer masking only adjusts the weights of layers a feature already occupies, with no mechanism to extend its support to new ones, despite the correlations between residual-stream activations across layers. Tensor factorization removes this barrier by tying the per-layer dictionary entries through shared factors, so weight updates propagate jointly across layers and latents. Layer masking, applied on top, supplies the optimization pressure that turns this coupling into genuinely cross-layer features.

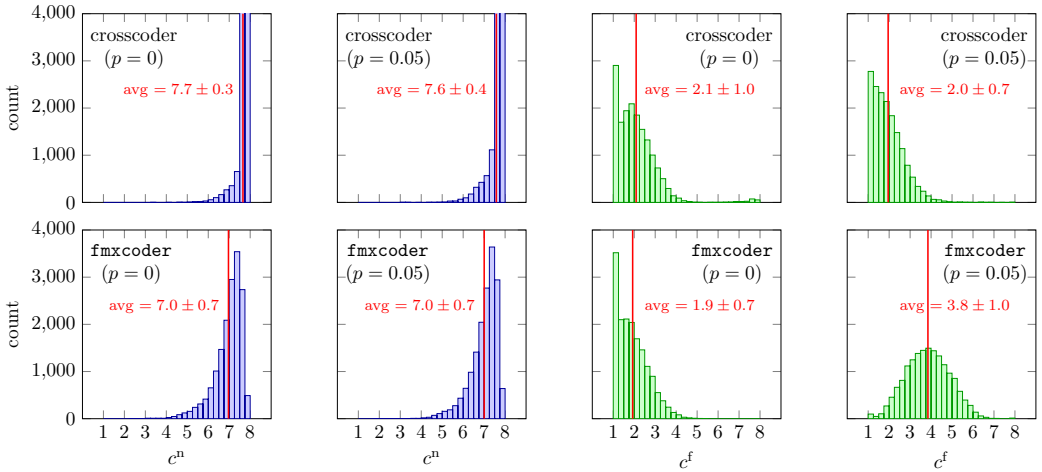

Figure 2: Norm coherence c^{n} (blue) and functional coherence c^{f} (green) latent distributions for crosscoders (top row) and `fmxcoder` (bottom row) variants trained on Pythia-410M. The average value and standard deviation of each coherence distribution are also shown in red color.

Functional coherence measures dependence via causal masking and, despite every layer carrying highly correlated residual stream information, functional coherence is far from uniform. This new metric reveals that most crosscoder latents concentrate on one or two layers, thus functional coherence doesn't just measure correlation. Our LLM-as-a-judge evaluation in the next section provides further evidence that high-coherence latents are predominantly interpretable semantic concept detectors while low-coherence latents collapse onto surface-level, layer-localized patterns.

4.3 Qualitative and LLM-Judge Analysis of Features

Table 1 contrasts examples of highest- and lowest-c^{f} latents recovered by an `fmxcoder` with p=0.05 trained on the activations of the Gemma2-2B model. *High-c^{f} latents act as concept detectors* that fire on diverse tokens unified by meaning, including morphological variants and named-entity instances. *Low-c^{f} latents instead collapse onto character classes*, single punctuation symbols, or short subword fragments with no shared semantics. Across both halves of the table, c^{n} remains in the narrow band 5.5–7.4 while c^{f} spans the full 1.0–7.1 range, confirming that weight magnitudes spread evenly across layers even when functional dependence does not, and motivating c^{f} as a more informative diagnostic.

To scale this analysis, we perform an *LLM-as-a-judge* evaluation, asking DeepSeek-V4-Flash, GPT-5.4 nano, Gemini 3.5 Flash as the LLM judges to score each latent on two independent axes: (i) the latent has discovered a semantic concept (firings share a meaning, concept, topic, sentiment, entity,

Table 1: Examples of high functional coherence c^{f} latents (top half of the table) and low c^{f} latents (bottom half of the table). The corresponding norm coherence c^{n} is also reported. The results correspond to an `fmxcoder` with p=0.05 trained on the activations of Gemma2-2B.

Latent firing pattern	Description	c^{f}	c^{n}
[*lobbying, campaigned, pleaded, pitch, ...*]	verbs of advocacy	7.1	6.1
[*regret, despise, failure, gruesome, ...*]	negative-affect words	7.0	6.9
[*Philadelphia, Pennsylvania, Maryland, Florida, ...*]	US states and cities	6.9	6.7
[*colleagues, friends, accomplice, partner, ...*]	companion relationship	6.8	6.3
[*vampire, werewolf, supernatural, superpowers, ...*]	supernatural beings/powers	6.7	7.4
[*1, 2, 3, 4, ...*]	digit detector	1.0	5.8
[*def, set, define, ->, ...*]	code/markup punctuation	1.0	6.2
[*d#, 3, :#, bbbb, ...*]	hex/colour-code characters	1.0	6.0
[*G, TT, BB, QL, ...*]	short sub-word fragments	1.0	6.5
[*), ;,],), ...*]	closing-brace family	1.0	5.5

etc.), and (ii) a surface-level pattern (grammatical role, character class, formatting, etc.). The latents are then labeled accordingly as "semantic" or "surface". Full procedure and prompts are provided in Appendix F. The mean increase in `fmxcoder` (p=0.05) latents relative to crosscoder latents for each category is presented in Table 2. Across all four base LLMs, `fmxcoder` *recovers substantially more semantic latents* than the standard crosscoder (2.6–14$\times$ depending on the model). Surface latents rise considerably less on three of the four base LLMs, and decrease on Gemma2-2B.

Table 2: Mean increase (ratio) and standard deviation in `fmxcoder` (p=0.05) latents relative to crosscoder latents per latent category, using an LLM-as-a-judge procedure. DeepSeek-V4-Flash, GPT-5.4 nano, Gemini 3.5 Flash are used as the LLM judges, and their results are averaged. Full procedure and prompts are provided in Appendix F.

	GPT2-Small	**Pythia-410M**	**Pythia-1.4B**	**Gemma2-2B**
Semantic	14.0 ± 1.7	9.4 ± 0.9	2.6 ± 0.3	7.0 ± 1.0
Surface	2.2 ± 0.2	3.0 ± 0.2	1.2 ± 0.2	0.8 ± 0.2

4.4 Causal Validation Through Steering

Feature discovery in mechanistic interpretability aims at human-interpretable features that can be functionally used. The following steering experiments evidence that the discovered features drive the LLM output towards the semantics they represent [48–50]. For each prompt we build a per-layer steering vector, $c \cdot \alpha_{ref} \cdot d_{\ell,i}$ where $d_{\ell,i}$ is the decoder write direction for latent i at layer ℓ, α_{ref} is the latent's peak observed activation, and c sets the steering magnitude. Injecting at every spanned layer would over-apply the feature (by up to a factor of L), since the residual stream accumulates each addition and the decoded directions are themselves cumulative. We therefore take each feature's per-layer decoder directions and replaces them with the layer-to-layer differences before applying any steering vectors. In this way, the steering directions are decomposed into per-layer contributions, so that the injections add up to a single insertion.

First, using generic prompts drawn from 1 million OpenWebText tokens and targeting one `fmxcoder` ($p = 0.05$) latent at a time, considering Gemma2-2B as the base LLM, we measure the mean increase in percentage points of next-token probability:

- the latent for "stages/steps of a process" promotes "steps" by 56% (c=10);
- the latent for "once X, now Y" pivot promotes "now" by 53% (c=10);
- the latent for "tone" promotes "tone" by 44% (c=10);
- the latent for "vast number of available options" promotes "infinite" by 25% (c=20);
- the latent for "Taylor" promotes "Taylor" by 34% (c=20).

In these results, steering with `fmxcoder` discovered features shifts the next-token probability towards the targeted feature even in generic contexts.

Second, we utilize targeted prompts based on the semantics of different features in order to target a direct shift in the generated text. For each prompt we compare the unsteered baseline continuation with the steered one. These results are presented in Table 3, giving a direct qualitative validation of the causal effect of the discovered features.

Table 3: Steering examples using `fmxcoder` discovered features in Gemma2-2B.

Feature	**Prompt**	**Regular output**	**Steered output**
Stages / steps of a process	Nobody warned me about the	pain of childbirth. I was told that it would be…	steps I would have to take to get my life back on track…
Country / state name	Dr. Amara Nwosu, who runs the computational biology unit at the children's hospital,	is one of the researchers who has been working on the project…	is a professor of pediatrics and biomedical informatics at the University of Michigan…
Accidentally / inadvertently	During the demo on Thursday afternoon, she	said, "I'm going to be a little bit more aggressive…	inadvertently revealed that the game will be released on the 25th…
The proper name "Taylor"	The street is named after a nineteenth-century mill owner named	John Smith. The mill was built in 1820…	John Taylor. The street is located in the area of…
Vast number of available options	By the end of the week he had received	a letter from the head of the department of the Ministry of…	thousands of messages of support from people across the country…
Concealment / covering up	The new packaging was designed to	be more environmentally friendly and to make it easier for consumers to…	hide the fact that the product is made from plastic…
Elapsed time relative to an event	They opened the restaurant and closed it	down in 2019. They are now open again…	2 weeks later. They were not ready to open…
The "once X, now Y" pivot	The factory where my father worked	was a small one, and the workers were all very close…	now is a museum. It is a museum of the history of the city…

4.5 Reconstruction and Sparse Probing Results

For the evaluation of the trained models we report reconstruction metrics on held-out data following the SAEBench library [51]. Table 4 reports the mean squared error (MSE), explained variance (EV), and cosine similarity (CS) across LLMs. We report all three metrics because they capture complementary aspects of reconstruction quality: MSE gives absolute error in native units, EV normalizes the error to enable cross-LLM comparison, and CS isolates directional fidelity, which downstream layers depend on. The standard crosscoder's high EV (0.986) but low CS (0.870) on Pythia-410M illustrate why all three metrics are needed. Across all four base LLMs, the standard crosscoder is the weakest reconstructor on every metric, with MSE up to roughly 100% higher than the per-layer SAE baseline and consistent drops in CS. The `fmxcoder` variants reverse this pattern, reducing MSE by 25–50% relative to the standard crosscoder, recovering CS that, in some cases, exceeds the SAE baseline, and matching the SAE explained variance. The masking variant pays a small reconstruction cost relative to the unmasked variant, which is the expected trade-off of a denoising objective and is offset by the functional coherence gains reported in Section 4.2.

Furthermore, Table 4 also reports sparse probing performance on six classification tasks: Bias in Bios [52], AG News [53], EuroParl [54], GitHub programming languages [55], Amazon Sentiment, and Amazon 15 [56]. While the F1 score measures whether a latent fires on the correct class, it is invariant to how strongly the firing distributions for positive and negative examples are separated. We complement it with the 1-Wasserstein distance between the activation distributions of the two classes, computed per latent and aggregated across the probing dataset [9]. Wasserstein distance measures the minimum "transport cost" to convert one distribution into the other, capturing both the gap between their means and differences in their shapes. A latent with high F1 but low Wasserstein distance discriminates the classes only marginally, whereas a latent with both high F1 and high Wasserstein distance separates the classes with a wide margin, indicating a more robust feature. Averaged across all six classification tasks, the standard crosscoder underperforms the per-layer SAE baseline by 13–18 F1 points on Pythia-410M, Pythia-1.4B, and Gemma2-2B, and shows near-zero Wasserstein distance across all four base LLMs, indicating latents that barely separate the probing classes. As with reconstruction metrics, `fmxcoders` reverse this gap entirely: without masking, F1 rises to 74–82%, exceeding the SAE baseline by 9–17 points on every model, and Wasserstein distance increases by one to three orders of magnitude, confirming that the gains reflect well-separated activation distributions rather than thin-margin classifications. The masking variants incur a small F1 cost (4–9

Table 4: Reconstruction metrics and probing-based results for different base LLMs, for SAE, crosscoder, and `fmxcoders` variants. We report average F1 across six classification tasks and the detailed results are presented in Appendix E. SAE results were reproduced using the settings from [9].

LLM	Variant	MSE ↓	EV ↑	CS ↑	Avg. F1 ↑	Avg. Wass. ↑ ($\times 10^{-3}$)
GPT2-Small	SAE	0.53	0.9256	0.9623	65.7	8.4
	Crosscoder	0.64	0.8704	0.9346	67.6	11.6
	`fmxcoder` ($p = 0$)	**0.32**	**0.9359**	**0.9682**	**78.3**	**120.2**
	`fmxcoder` ($p = 0.05$)	0.33	0.9332	0.9670	73.5	68.4
Pythia-410m	SAE	**0.03**	**0.9918**	**0.9364**	65.0	0.5
	Crosscoder	0.05	0.9856	0.8696	47.4	<0.1
	`fmxcoder` ($p = 0$)	**0.03**	0.9917	0.9275	**74.3**	**10.2**
	`fmxcoder` ($p = 0.05$)	**0.03**	0.9914	0.9257	68.1	4.5
Pythia-1.4b	SAE	**0.22**	**0.9784**	**0.9289**	64.6	0.8
	Crosscoder	0.31	0.9691	0.8976	47.0	0.2
	`fmxcoder` ($p = 0$)	**0.22**	0.9780	0.9273	**78.4**	**36.1**
	`fmxcoder` ($p = 0.05$)	0.23	0.9774	0.9252	71.2	13.1
Gemma2-2b	SAE	**1.58**	**0.8702**	0.9215	64.8	2.7
	Crosscoder	3.29	0.7809	0.8737	51.2	0.2
	`fmxcoder` ($p = 0$)	2.03	0.8632	**0.9229**	**81.9**	**198.7**
	`fmxcoder` ($p = 0.05$)	2.07	0.8608	0.9218	73.0	39.1

points) and lower Wasserstein distance, but they nonetheless remain well above the SAE and standard crosscoder baselines on F1 and Wasserstein distance. The individual sparse probing results across all the classification datasets considered in this study are presented in Appendix E.

To assess the variance of the results presented in Table 4 we trained 5 different `fmxcoder` ($p = 0.05$) models for each base LLM, all corresponding to different seeds. The standard deviation of reconstruction metrics is in the order of 1% for the average values. The standard deviation in probing metrics is larger ($< 3.6\%$ of Avg. F1 and $< 9\%$ Avg. Wass.) but much smaller compared to the improvements of `fmxcoder` variants with respect to crosscoders/SAEs.

5 Limitations

A few aspects of the present study limit the scope of these conclusions. First, `fmxcoders` introduce two new hyperparameters, the masking probability p and the factorization ranks, whose optimal settings appear to be model- and dataset-dependent (see Appendix C). Our sweeps explored only a coarse grid and we do not provide an *a priori* procedure for selecting p when the downstream task is unknown. Second, distinguishing semantic from surface latents is itself a subtle judgment, often hinging on whether a shared grammatical role also carries shared meaning, and affordable judges are not always reliable at this distinction. Finally, `fmxcoders` incur a slight additional training-time cost relative to standard crosscoders (<10%) due to the tensor-factorization forward/backward passes. Total compute costs are reported in Appendix A.

6 Conclusion

We introduced the `fmxcoder` architecture, that uses low-rank tensor decomposition for the encoder/decoder tensors and applies stochastic layer masking to act as a remedy to standard crosscoders' tendency to learn primarily layer-localized features. Together, the two interventions raise mean probing F1 by 10–30 points across GPT2-Small, Pythia-410M, Pythia-1.4B, and Gemma2-2B, surpassing per-layer SAE baselines that the standard crosscoder fails to reach, reduce reconstruction MSE by 25–50%, and roughly double the average functional coherence of the recovered latents. A qualitative inspection further indicates that high-coherence latents act as semantic concept detectors (US states and cities, negative-affect words, verbs of advocacy), while low-coherence latents collapse onto surface-level character classes and syntactic tokens. An LLM-as-a-judge evaluation shows that `fmxcoders` recover 2.6–14$\times$ more semantic concept latents across the different models. Finally, steering experiments show the causal effect of the discovered features, both in generic and targeted contexts.

Acknowledgements

This work is supported by the TensorICE project (EXCELLENCE/0524/0407), implemented under the social cohesion programme "THALIA 2021-2027", co-funded by the European Union through the Research and Innovation Foundation. Yannis Panagakis was supported by the project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program. The authors gratefully acknowledge the EuroHPC Joint Undertaking for awarding this project access to the MareNostrum5 supercomputer, hosted by the Barcelona Supercomputing Center, Spain, under Development Access project ID EHPC-DEV-2025D06-069. This work was also supported by computing time awarded on the Aphroditi supercomputer of the High Performance Computing Facility of The Cyprus Institute.

References

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

[2] L. Sharkey, B. Chughtai, J. Batson, J. Lindsey, J. Wu, L. Bushnaq, N. Goldowsky-Dill, S. Heimersheim, A. Ortega, J. Bloom, S. Biderman, A. Garriga-Alonso, A. Conmy, N. Nanda, J. Rumbelow, M. Wattenberg, N. Schoots, J. Miller, E. J. Michaud, S. Casper, M. Tegmark, W. Saunders, D. Bau, E. Todd, A. Geiger, M. Geva, J. Hoogland, D. Murfet, and T. McGrath, "Open problems in mechanistic interpretability," *arXiv preprint arXiv:2501.16496*, 2025.

[3] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, and C. Olah, "Towards monosemanticity: Decomposing language models with dictionary learning," *Transformer Circuits Thread*, 2023. [Online]. Available: https://transformer-circuits.pub/2023/monosemantic-features/index.html

[4] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey, "Sparse autoencoders find highly interpretable features in language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[5] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, and T. Henighan, "Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet," *Transformer Circuits Thread*, 2024. [Online]. Available: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html

[6] L. Gao, T. Dupré la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu, "Scaling and evaluating sparse autoencoders," in *International Conference on Learning Representations (ICLR)*, 2025, oral presentation.

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

[8] B. Bussmann, P. Leask, and N. Nanda, "BatchTopK sparse autoencoders," in *NeurIPS 2024 Workshop on Scientific Methods for Understanding Neural Networks*, 2024.

[9] P. Koromilas, A. D. Demou, J. Oldfield, Y. Panagakis, and M. Nicolaou, "Polysae: Modeling feature interactions in sparse autoencoders via polynomial decoding," *arXiv preprint arXiv:2602.01322*, 2026.

[10] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang, and F. Wei, "Knowledge neurons in pretrained transformers," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022, pp. 8493–8502.

[11] V. Lad, J. H. Lee, W. Gurnee, and M. Tegmark, "Remarkable robustness of LLMs: Stages of inference?" in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. [Online]. Available: https://openreview.net/forum?id=Wxh5Xz7NpJ

[12] M. Geva, A. Caciularu, K. Wang, and Y. Goldberg, "Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space," in *Proceedings of the 2022 conference on empirical methods in natural language processing*, 2022, pp. 30–45.

[13] W. Gurnee, T. Horsley, Z. C. Guo, T. R. Kheirkhah, Q. Sun, W. Hathaway, N. Nanda, and D. Bertsimas, "Universal neurons in gpt2 language models," *arXiv preprint arXiv:2401.12181*, 2024.

[14] T. Lawson, L. Farnik, C. Houghton, and L. Aitchison, "Residual stream analysis with multi-layer SAEs," in *International Conference on Learning Representations (ICLR)*, 2025.

[15] N. Balagansky, I. Maksimov, and D. Gavrilov, "Mechanistic permutability: Match features across layers," in *International Conference on Learning Representations (ICLR)*, 2025.

[16] J. Lindsey, A. Templeton, J. Marcus, T. Conerly, J. Batson, and C. Olah, "Sparse crosscoders for cross-layer features and model diffing," *Transformer Circuits Thread*, 2024. [Online]. Available: https://transformer-circuits.pub/2024/crosscoders/index.html

[17] E. Ameisen, J. Lindsey, A. Pearce, W. Gurnee, N. L. Turner, B. Chen, C. Citro, D. Abrahams, S. Carter, B. Hosmer, J. Marcus, M. Sklar, A. Templeton, T. Bricken, C. McDougall, H. Cunningham, T. Henighan, A. Jermyn, A. Jones, A. Persic, Z. Qi, T. B. Thompson, S. Zimmerman, K. Rivoire, T. Conerly, C. Olah, and J. Batson, "Circuit tracing: Revealing computational graphs in language models," *Transformer Circuits Thread*, 2025. [Online]. Available: https://transformer-circuits.pub/2025/attribution-graphs/methods.html

[18] C. Dumas, J. Minder, C. Juang, B. Chughtai, and N. Nanda, "Overcoming sparsity artifacts in crosscoders to interpret chat-tuning," in *Mechanistic Interpretability Workshop at NeurIPS 2025*, 2025.

[19] Y. Panagakis, J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, "Tensor methods in computer vision and deep learning," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 863–890, 2021.

[20] S. C. Marshall and J. H. Kirchner, "Understanding polysemanticity in neural networks through coding theory," *arXiv preprint arXiv:2401.17975*, 2024.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[22] A. Conmy, A. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso, "Towards automated circuit discovery for mechanistic interpretability," *Advances in Neural Information Processing Systems*, vol. 36, pp. 16 318–16 352, 2023.

[23] M. Hanna, O. Liu, and A. Variengien, "How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model," *Advances in Neural Information Processing Systems*, vol. 36, pp. 76 033–76 060, 2023.

[24] S. Marks, C. Rager, E. J. Michaud, Y. Belinkov, D. Bau, and A. Mueller, "Sparse feature circuits: Discovering and editing interpretable causal graphs in language models," *arXiv preprint arXiv:2403.19647*, 2024.

[25] D. Troitskii, K. Pal, C. Wendler, C. S. McDougall, and N. Nanda, "Internal states before wait modulate reasoning patterns," *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP*, 2025.

[26] A. Wadell, A. Bhutani, V. Azumah, A. R. Ellis-Mohr, C. Kelly, H. Zhao, A. K. Nayak, K. Hegazy, A. Brace, and H. Lin, "Foundation models for discovery and exploration in chemical space," *arXiv preprint arXiv:2510.18900*, 2025.

[27] D. D. Baek and M. Tegmark, "Towards understanding distilled reasoning models: A representational approach," *arXiv preprint arXiv:2503.03730*, 2025.

[28] X. Ge, W. Shu, J. Wu, Y. Zhou, Z. He, and X. Qiu, "Evolution of concepts in language model pre-training," in *The Fourteenth International Conference on Learning Representations*, 2026.

[29] D. Bayazit, A. Mueller, and A. Bosselut, "Crosscoding through time: Tracking emergence & consolidation of linguistic representations throughout LLM pretraining," *arXiv preprint arXiv:2509.05291*, 2025.

[30] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

[31] Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki, "Tensor Ring Decomposition," *arXiv preprint arXiv:1606.05535*, 2016.

[32] J. Oldfield, M. Georgopoulos, G. G. Chrysos, C. Tzelepis, Y. Panagakis, M. A. Nicolaou, J. Deng, and I. Patras, "Multilinear mixture of experts: Scalable expert specialization through factorization," *Advances in Neural Information Processing Systems*, vol. 37, pp. 53 022–53 063, 2024.

[33] J. Oldfield, S. Im, S. Li, M. A. Nicolaou, I. Patras, and G. G. Chrysos, "Towards interpretability without sacrifice: Faithful dense layer decomposition with mixture of decoders," *arXiv preprint arXiv:2505.21364*, 2025.

[34] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*. ACM, 2008, pp. 1096–1103.

[35] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019, pp. 4171–4186.

[37] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 000–16 009.

[38] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on signal processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[39] B. Romera-Paredes, H. Aung, N. Bianchi-Berthouze, and M. Pontil, "Multilinear multitask learning," in *International Conference on Machine Learning*. PMLR, 2013, pp. 1444–1452.

[40] Y. Yang and T. Hospedales, "Deep multi-task representation learning: A tensor factorisation approach," *arXiv preprint arXiv:1605.06391*, 2016.

[41] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Computation*, vol. 7, no. 1, pp. 108–116, 1995.

[42] J. Bloom, C. Tigges, A. Duong, and D. Chanin, "SAELens," https://github.com/jbloomAus/SAELens, 2024, gitHub repository.

[43] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI, Tech. Rep., 2019. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[44] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. Van Der Wal, "Pythia: A suite for analyzing large language models across training and scaling," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 2397–2430. [Online]. Available: https://proceedings.mlr.press/v202/biderman23a.html

[45] Gemma Team, "Gemma 2: Improving open language models at a practical size," Google DeepMind, Tech. Rep., 2024. [Online]. Available: https://storage.googleapis.com/deepmind-media/gemma/gemma-2-report.pdf

[46] A. Gokaslan, V. Cohen, E. Pavlick, and S. Tellex, "OpenWebText corpus," Zenodo, 2019. [Online]. Available: https://zenodo.org/records/3834942

[47] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy, "The Pile: An 800GB dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2021.

[48] A. M. Turner, L. Thiergart, G. Leech, D. Udell, J. J. Vazquez, U. Mini, and M. MacDiarmid, "Steering language models with activation engineering," *arXiv preprint arXiv:2308.10248*, 2023.

[49] N. Rimsky, N. Gabrieli, J. Schulz, M. Tong, E. Hubinger, and A. Turner, "Steering llama 2 via contrastive activation addition," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 15 504–15 522.

[50] S. Chalnev, M. Siu, and A. Conmy, "Improving steering vectors by targeting sparse autoencoder features," *arXiv preprint arXiv:2411.02193*, 2024.

[51] A. Karvonen, C. Rager, J. Lin, C. Tigges, J. I. Bloom, D. Chanin, Y.-T. Lau, E. Farrell, C. S. McDougall, K. Ayonrinde, D. Till, M. Wearden, A. Conmy, S. Marks, and N. Nanda, "SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 267. PMLR, 2025, pp. 29 223–29 264. [Online]. Available: https://proceedings.mlr.press/v267/karvonen25a.html

[52] M. De-Arteaga, A. Romanov, H. Wallach, J. Chayes, C. Borgs, A. Chouldechova, S. C. Geyik, K. Kenthapadi, and A. T. Kalai, "Bias in bios: A case study of semantic representation bias in a high-stakes setting," in *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19)*. ACM, 2019, pp. 120–128.

[53] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, vol. 28, 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf

[54] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *Proceedings of Machine Translation Summit X: Papers*, Phuket, Thailand, Sep. 13-15 2005, pp. 79–86.

[55] CodeParrot, "Github code dataset," https://huggingface.co/datasets/codeparrot/github-code, 2022.

[56] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, and J. McAuley, "Bridging language and items for retrieval and recommendation," *arXiv preprint arXiv:2403.03952*, 2024.

[57] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis," *UCLA working papers in phonetics*, vol. 16, no. 1, p. 84, 1970.

A Implementation Details

This appendix documents the training setup, hyperparameters, and infrastructure used for all experiments in the main paper. All crosscoder variants (standard crosscoder, TR `fmxcoder`, and CP `fmxcoder`) were trained with identical optimizer, schedule, and data settings; the only differences are the encoder/decoder parameterization and the presence of stochastic layer masking.

A.1 Architectures and Hyperparameters

Dictionary configuration. All crosscoders and `fmxcoders` use a dictionary width of $d_{\text{sae}} = 16{,}384$ latents. The sparsity operator is BatchTopK [8] with $K = 64$ active latents per token. A pre-activation ReLU is applied before BatchTopK selection. The encoder bias $\boldsymbol{b}^{\text{enc}} \in \mathbb{R}^{d_{\text{sae}}}$ is shared across layers; per-layer decoder biases $\boldsymbol{b}_\ell^{\text{dec}} \in \mathbb{R}^d$ are learned independently for each $\ell \in [1, L]$.

Layer selection. For each base LLM, we use $L = 8$ residual-stream activations collected at post-MLP positions spanning the middle layers. The specific layer indices are (zero-indexed):

- GPT2-Small (12 layers): layers 2–9.
- Pythia-410M (24 layers): layers 8–15.
- Pythia-1.4B (24 layers): layers 8–15.
- Gemma2-2B (26 layers): layers 9–16.

Factorization ranks. For TR `fmxcoders` matching the standard crosscoder parameter count, the ranks (R_1, R_2, R_3) were chosen subject to $R_2/R_1 = \sqrt{d/L}$ and $R_3/R_1 = \sqrt{d_{\text{sae}}/d}$. The specific values per base LLM are:

- GPT2-Small ($d = 768$): $(5, 244, 25)$.
- Pythia-410M ($d = 1024$): $(7, 302, 27)$.
- Pythia-1.4B ($d = 2048$): $(11, 501, 31)$.
- Gemma2-2B ($d = 2304$): $(12, 545, 32)$.

For CP `fmxcoders`, the rank R was chosen to match the standard crosscoder parameter count; specific values per base LLM are:

- GPT2-Small: $R = 5866$.
- Pythia-410M: $R = 7707$.
- Pythia-1.4B: $R = 14557$.
- Gemma2-2B: $R = 16153$.

Masking probability. The default masking probability for `fmxcoders` reported in the main results is $p = 0.05$. The unmasked variant uses $p = 0$. Bernoulli masks are drawn independently per token and per layer at every training step. Ablations of the masking probability are presented in Appendix C.

A.2 Optimization

Optimizer. We use Adam with a learning rate 3×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, with no warmup or decay schedules. For training stabilization, we use gradient clipping with a maximum norm of 1.0.

Batch size. 4096.

Training tokens. All training runs use 500M tokens (300M for GPT2-Small) at a context length of 128.

Initialization. Encoder and decoder tensors / decomposition factors follow a variance-matched i.i.d. Gaussian initialization, chosen so that the materialized 3-D weight tensor has the same per-element variance as a Kaiming-uniform-initialized 2-D slice. Biases are initialized as zero.

Decoder normalization. No decoder normalization is applied (e.g. similar to [3]).

A.3 Data

Activation generation. For Gemma2-2B and GPT2-Small, activations were generated using OpenWebText [46]. For Pythia-410M and Pythia-1.4B, activations were generated using an uncopyrighted variant of the deduplicated Pile [47]. All activations are cached at the post-MLP residual-stream position for each of the $L = 8$ selected layers.

Held-out evaluation set. Reconstruction metrics are computed on a held-out set of approximately 410k tokens not seen during training. The same held-out set is used for all variants of a given base LLM to ensure direct comparability.

A.4 Sparse Probing Setup

We follow the protocol of [9] for sparse probing. For each classification task, we select the top-$K = 1$ latent that best separates classes by F1 on a training split, then report F1 and Wasserstein distance on a held-out evaluation split. Class labels and dataset splits follow the original sources for Bias in Bios [52], AG News [53], EuroParl [54], GitHub programming languages [55], Amazon Sentiment, and Amazon 15 [56].

A.5 Functional Coherence Computation

The functional coherence metric (Section 4.2) is computed on a held-out set of 10M tokens, in order to obtain enough sample size for individual latent evaluations. For each layer $\ell \in [1, L]$, we run an additional forward pass through the crosscoder with $\boldsymbol{x}_\ell$ set to zero while leaving the other layers intact, and record the resulting per-latent activations. The numerical-stability constant in Equation (5) is set to $\epsilon = 10^{-8}$.

A.6 Compute and Reproducibility

Hardware. All experiments were run on NVIDIA H100 64GB GPUs, with a single GPU per training run.

Wall-clock time. A single `fmxcoder` run takes approximately 12 hours for GPT2-Small, 16 hours for Pythia-410M, 30 hours for Pythia-1.4B, and 42 hours for Gemma2-2B.

Total compute. The full set of experiments reported in the main paper and appendices required approximately 2000 GPU hours of compute across all base LLMs, factorization variants, and ablation settings.

Software. Crosscoder training is implemented in SAELens [42], a PyTorch-based SAE training library in Python. Tensor Ring and CP factorizations are implemented as native PyTorch modules. Reconstruction metrics (MSE, EV, CS) are computed following the protocols of SAEBench [51]. Probing F1 and Wasserstein distance follow [9]. The LLM-as-a-judge evaluation calls three models from different frontier LLM labs through the OpenRouter API.

B CP `fmxcoders`

B.1 Mathematical definition

The Canonical Polyadic (CP) decomposition [30, 57] approximates a tensor as a sum of rank-one tensors formed from vector outer products. For the encoder tensor with rank R,

$$\mathbf{E} \approx \sum_{r\in[1,R]} \boldsymbol{w}_r \circ \boldsymbol{u}_r \circ \boldsymbol{v}_r, \qquad \mathbf{E}_{j,i,\ell} \approx \sum_{r\in[1,R]} W_{jr}\, U_{ir}\, V_{\ell r}, \tag{6}$$

where $\boldsymbol{w}_r \in \mathbb{R}^d$, $\boldsymbol{u}_r \in \mathbb{R}^{d_{\text{sae}}}$, $\boldsymbol{v}_r \in \mathbb{R}^L$ are the factor vectors, collected into factor matrices $\boldsymbol{W} \in \mathbb{R}^{d\times R}$, $\boldsymbol{U} \in \mathbb{R}^{d_{\text{sae}}\times R}$, $\boldsymbol{V} \in \mathbb{R}^{L\times R}$. The decoder factorizes independently with factors $\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{U}}, \tilde{\boldsymbol{V}}$ of matching shapes.

Each rank-one term is an outer product of an activation-space direction $\boldsymbol{w}_r$, a feature-weight vector $\boldsymbol{u}_r$, and a layer pattern $\boldsymbol{v}_r$, making the full encoder a sum of R atoms shared across all features. This results in a *uniform coupling* inductive bias: a single rank R ties the three modes together at once, and every feature expresses its cross-layer behavior in the same R-dimensional basis of layer patterns (the rows of $\boldsymbol{V}$). Crucially, a feature that appears across layers *is represented once* through its R-dimensional coefficients rather than re-discovered at each layer, and the parameter count drops from $\mathcal{O}(L \cdot d_{\text{sae}} \cdot d)$ to $\mathcal{O}(R \cdot (d + d_{\text{sae}} + L))$ per tensor.

B.2 Results

This section presents reconstruction and sparse-probing results for CP `fmxcoders`, reported in Table 5. The qualitative picture matches the TR results in Table 4: CP `fmxcoders` improve over the standard crosscoder on every metric and every LLM, reducing MSE by up to 50%, recovering cosine similarity to within 0.01 of the per-layer SAE baseline, and lifting average probing F1 by up to 30 points above the crosscoder. As in the TR case, the masking variant trades a small amount of reconstruction quality and F1 for a denoising effect that is reflected in the functional coherence metric (see Appendix C). Comparing the two factorizations directly, CP `fmxcoders` match TR `fmxcoders` closely on MSE, explained variance, and cosine similarity, but under-perform on average F1 on 3 out of 4 LLMs, and on Wasserstein distance (by roughly a factor of 2–5 $\times$). They nonetheless remain well above the crosscoder baseline on Wasserstein, by one to two orders of magnitude. We attribute this gap to the structural difference between the two decompositions discussed in Section 3.2: TR couples the three modes through three independent ranks (R_1, R_2, R_3) with context-dependent interaction across features and layers, while CP ties all three modes through a single shared rank R and a uniform basis of layer patterns. The TR parameterization therefore admits richer per-feature layer profiles, more suitable for our setting where $L << d_{sae}$, which appears to translate into more separable latent activations on the probing tasks.

Table 5: Reconstruction metrics and probing-based results for different base LLMs, for SAE, crosscoder, and `fmxcoder` variants. `fmxcoders` adopt the CP decomposition, with ranks chosen to match the parameters of the crosscoders in each case. We report average F1 across six classification tasks and the detailed results are presented in Appendix E. SAE results were reproduced using the settings from [9].

LLM	Variant	MSE	EV ↑	CS ↑	Avg. F1 ↑	Avg. Wass. ↑ ($\times 10^{-3}$)
GPT2-Small	SAE	0.53	0.9256	0.9623	65.7	8.4
	Crosscoder	0.64	0.8704	0.9346	67.6	11.6
	`fmxcoder` ($p = 0$)	**0.32**	**0.9355**	**0.9679**	**74.3**	**25.8**
	`fmxcoder` ($p = 0.05$)	0.33	0.9330	0.9668	69.7	18.5
Pythia-410m	SAE	**0.03**	**0.9918**	**0.9364**	65.0	0.5
	Crosscoder	0.05	0.9856	0.8696	47.4	<0.1
	`fmxcoder` ($p = 0$)	**0.03**	0.9917	0.9273	**77.5**	**4.7**
	`fmxcoder` ($p = 0.05$)	**0.03**	0.9915	0.9263	65.6	2.1
Pythia-1.4b	SAE	**0.22**	**0.9784**	**0.9289**	64.6	0.8
	Crosscoder	0.31	0.9691	0.8976	47.0	0.2
	`fmxcoder` ($p = 0$)	**0.22**	0.9781	0.9275	**76.2**	**10.8**
	`fmxcoder` ($p = 0.05$)	0.23	0.9777	0.9263	65.8	5.5
Gemma2-2b	SAE	**1.58**	**0.8702**	0.9215	64.8	2.7
	Crosscoder	3.29	0.7809	0.8737	51.2	0.2
	`fmxcoder` ($p = 0$)	2.05	0.8622	**0.9226**	**79.0**	**38.3**
	`fmxcoder` ($p = 0.05$)	2.05	0.8621	0.9223	65.4	16.9

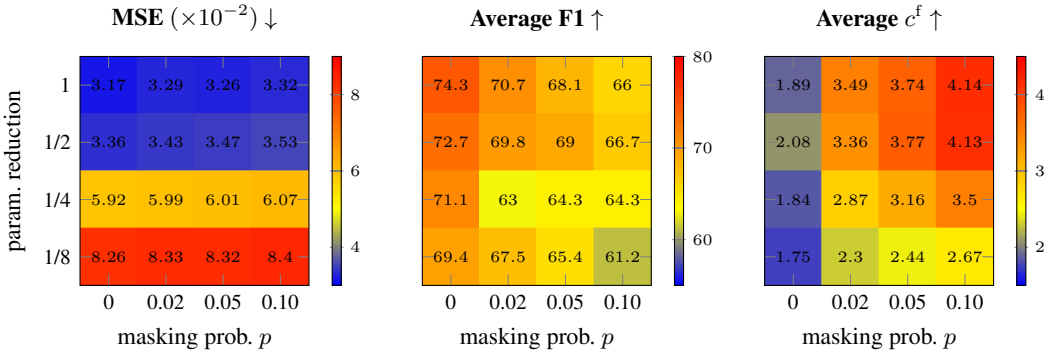

Figure 3: Heat maps for MSE, mean F1 and mean functional coherence c^{f}, for different values of the masking probability p and parameter reduction associated with decreasing factorization ranks for TR `fmxcoders`. (R_1, R_2, R_3) take the values $(7, 302, 27)$ for matching the unfactorized crosscoder number of parameters, $(5, 214, 19)$ for a $1/2$ reduction, $(3, 151, 13)$ for a $1/4$ reduction, and $(2, 107, 9)$ for a $1/8$ reduction. In all cases, the activations were collected from Pythia-410m.

C Mask Probability and Rank Ablations

This appendix reports ablations of the two hyperparameters introduced by `fmxcoders`: the masking probability p and the factorization ranks. Figure 3 shows results for the TR variant and Figure 4 for the CP variant, both trained on Pythia-410m. Each figure reports MSE, mean probing F1, and average functional coherence c^{f} as p varies in $\{0, 0.02, 0.05, 0.10\}$ and the parameter count is reduced by factors of 1, $1/2$, $1/4$, and $1/8$ relative to the standard crosscoder. The ranks were chosen to match the target parameter count subject to the constraints $R_2/R_1 = \sqrt{d/L}$ and $R_3/R_1 = \sqrt{d_{\mathrm{sae}}/d}$, which balance the contribution of each mode to the factorization. The corresponding rank settings are listed in the figure captions.

Effect of masking probability. Across both factorizations, p controls a clean trade-off between reconstruction quality and functional coherence. Functional coherence increases monotonically with p, roughly doubling from $c^{\mathrm{f}} \approx 2$ at p=0 to $c^{\mathrm{f}} \approx 4$ at p=0.1 for both TR and CP at full rank. Reconstruction MSE is essentially flat in p at any fixed rank (e.g. $3.17 \to 3.32$ for TR at full rank), confirming that masking exerts pressure on *which latents are learned* rather than on overall reconstruction capacity. Average F1 declines moderately with p (up to 14 points), consistent with the denoising objective penalizing layer-localized features in favor of more distributed ones. The choice of p is therefore a knob the user can set based on whether downstream interpretability or

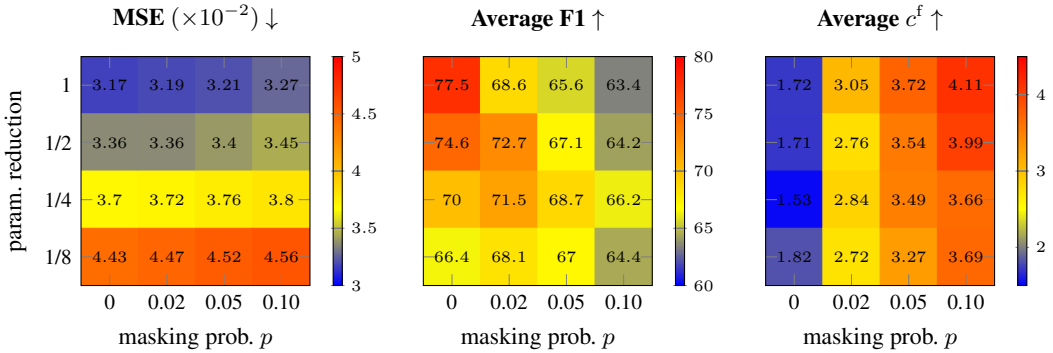

Figure 4: Similar to Figure 3, for CP `fmxcoders`. R=7707 for matching the unfactorized crosscoder number of parameters, R=3853 for a $1/2$ reduction, R=1926 for a $1/4$ reduction, and R=963 for a $1/8$ reduction.

sparse-probing performance is the priority: small p (≤ 0.02) preserves probing F1 with modest coherence gains, while $p \geq 0.05$ produces substantially more cross-layer latents at a small reconstruction and F1 cost.

Effect of factorization rank. Reducing the rank reduces parameter count and degrades reconstruction and F1, but does not affect functional coherence. At p=0, c^{f} remains in the narrow band 1.5–2.1 across the full $8\times$ compression sweep, indicating that capacity controls how well features are reconstructed but not how distributed they are across layers. Functional coherence is evidently governed by the optimization pressure from masking, not by the size of the parameter budget. The two factorizations differ markedly in how gracefully they tolerate compression: CP's MSE rises only from approximately 3.2 to 4.5 at $1/8$ reduction, while TR's rises from approximately 3.2 to 8.3, more than doubling.

Joint behavior. The two hyperparameters act largely independently: p shifts coherence with weak effect on reconstruction, while rank shifts reconstruction and F1 with no effect on coherence. The best F1 is consistently obtained at full rank with p=0 (77.5 for CP, 74.3 for TR), and the best coherence at full rank with p=0.1 ($c^{\mathrm{f}} \approx 4.1$ for both). No setting in the sweep simultaneously maximizes both, reflecting the fundamental trade-off introduced by the denoising objective; intermediate values such as p=0.05 at full rank offer a reasonable balance and are the settings used in the main results.

D Ablation study of architectural components

Table 6 presents results from `fmxcoder` ($p = 0.05$) experiments on Pythia-410M, by varying different architectural components. Some observations are listed below:

- Dictionary size: Increasing the dictionary size consistently improves the reconstruction metrics, with MSE decreasing, and EV/CosSim increasing. This can be attributed to the corresponding increase of the total parameters of the model. Nonetheless, there is no clear trend on how it affects the Avg. F1, while the Avg. Wasserstein distance decreases significantly, hinting that class separation during probing becomes less effective.
- Number of active latents K: Similar to the dictionary size, increasing K has a positive effect on the reconstruction metrics, as the activation of more latents can provide more expressive outputs. There is also a clear gain in Avg. F1 and Wasserstein distances, indicating better probing performance.
- Different sparsifier: We compare a Top$K$ sparsifier against the BatchTop$K$ sparsifier. Due to the different inductive bias of the two sparsifiers, the Top$K$ exhibits different results, specifically slightly worse reconstruction metrics, but noticeably better Avg. F1 and Wasserstein distances.
- Different number of layers: As the number of layers increases, the reconstruction cost increases, which we hypothesize is due to reconstructing more layer activations from the same latents. Moreover, there is no clear trend of the effects on probing metrics, even though the full-depth model achieves the best F1 and Wasserstein distance results.
- Attention hooks: The results when collecting activations from the output of the attention layer reveal reduced MSE, EV and CS, highlighting the significant differences of the input signal. The differences in the probing metrics are relatively smaller. It is noted that almost all works in the literature apply crosscoders on the residual stream (post MLP). Only the cross-layer transcoder [17] uses activations at the input and output of each MLP, but this is a significantly different architecture compared to our

Table 6: Ablations of dictionary size, number of active latents K, sparsifier, number of layers and hook position. Unless varied, all runs use 16,384 latents, $K = 64$, the BatchTop$K$ sparsifier, $L = 8$ and residual (post-MLP) hook position.

Ablation	**Variant**	MSE ↓	EV ↑	CS ↑	Avg. F1 ↑	Avg. Wass. ↑ ($\times 10^{-3}$)
Dictionary size	8,192	0.037	0.990	0.914	68.9	12.6
	16,384	0.033	0.991	0.926	68.1	4.5
	32,768	0.030	0.992	0.933	69.0	2.4
K	32	0.039	0.990	0.913	63.8	3.2
	64	0.033	0.991	0.926	68.1	4.5
	128	0.028	0.993	0.935	70.5	7.3
Sparsifier	BatchTop$K$	0.033	0.991	0.926	68.1	4.5
	Top$K$	0.034	0.991	0.923	76.3	14.9
Layers	$L = 4$	0.030	0.992	0.929	70.6	4.8
	$L = 8$	0.033	0.991	0.926	68.1	4.5
	$L = 12$	0.036	0.990	0.920	64.6	7.8
	$L = 24$ (full depth)	0.052	0.946	0.913	73.8	19.7
Hook position	Residual (post-MLP)	0.033	0.991	0.926	68.1	4.5
	Attention output	0.003	0.760	0.867	72.6	2.1

work. We are not aware of any other work that applies crosscoders using attention hooks to compare or draw insight from it.

E Full probing results

Table 7 presents the per-dataset probing results underlying the averages reported in Table 4 and Table 5. Three patterns are worth noting. First, the standard crosscoder's under-performance relative to the per-layer SAE is broad rather than driven by any single dataset, with losses of 10–30 F1 points across most LLM and dataset combinations on the three larger models. Second, the gains of TR `fmxcoders` are similarly systematic, beating the SAE on 22 of the 24 LLM and dataset combinations. Third, gaps in Wasserstein distances are larger than gaps in F1 in relative terms, with `fmxcoder` values exceeding the crosscoder by one to three orders of magnitude even on datasets where F1 improvements are modest, supporting the main-text observation that F1 alone understates the separation gain. The comparison between TR and CP is dataset-dependent: TR achieves the higher average F1 on 3 out of 4 LLMs, but CP wins on individual datasets in several cases.

Table 7: Probing experiments across datasets at K=1. Format: F1 scores (%) / Wasserstein distance ($\times 10^{-3}$). The last two columns show average F1 and average Wasserstein distance. `fmxcoders` correspond to p=0.

LLM	Variant	EuroParl	Bios	Amazon Sentiment	GitHub	AG News	Amazon 15	Avg. F1 (%) ↑	Avg. Wass. ↑ ($\times 10^{-3}$)
GPT2-Small	SAE	67.4 / 19.0	59.6 / 7.3	68.8 / 4.4	68.1 / 8.8	65.3 / 8.2	65.1 / 2.9	65.7	8.4
	Crosscoder	85.0 / 23.0	55.7 / 6.2	67.9 / 7.5	66.6 / 20.4	62.5 / 6.0	68.1 / 6.2	67.6	11.6
	CP `fmxcoder`	84.8 / 55.7	72.3 / 24.3	81.9 / 11.3	68.9 / 28.0	66.8 / 26.8	71.1 / 8.5	74.3	25.8
	TR `fmxcoder`	93.2 / 256.6	78.8 / 114.7	82.1 / 53.8	67.8 / 126.1	75.6 / 130.9	72.5 / 38.9	78.3	120.2
Pythia-410m	SAE	90.9 / 0.8	60.5 / 0.3	63.9 / 0.4	59.7 / 1.1	58.7 / 0.3	56.6 / 0.3	65.0	0.5
	Crosscoder	58.1 / 0.2	31.4 / 0.0	55.6 / 0.0	58.7 / 0.0	33.9 / 0.0	46.7 / 0.0	47.4	0.0
	CP `fmxcoder`	91.5 / 7.0	74.9 / 4.4	79.7 / 2.7	72.0 / 7.8	73.6 / 4.3	73.0 / 1.9	77.5	4.7
	TR `fmxcoder`	87.3 / 13.7	65.1 / 8.9	86.6 / 6.8	75.6 / 18.6	69.3 / 8.8	61.9 / 4.5	74.3	10.2
Pythia-1.4b	SAE	74.0 / 0.6	65.0 / 0.5	57.2 / 0.6	63.3 / 2.3	65.2 / 0.4	63.2 / 0.4	64.6	0.8
	Crosscoder	40.1 / 0.0	40.2 / 0.0	50.7 / 0.2	59.3 / 0.9	33.5 / 0.0	58.5 / 0.2	47.0	0.2
	CP `fmxcoder`	98.7 / 14.1	70.4 / 9.6	88.1 / 6.4	60.4 / 20.4	69.2 / 9.6	70.6 / 4.6	76.2	10.8
	TR `fmxcoder`	94.3 / 46.6	72.2 / 34.0	89.2 / 22.9	78.4 / 63.1	66.6 / 34.3	69.6 / 15.8	78.4	36.1
Gemma2-2b	SAE	68.3 / 1.9	67.6 / 2.6	71.1 / 2.8	64.4 / 4.5	59.9 / 2.7	57.6 / 1.9	64.8	2.7
	Crosscoder	69.9 / 1.0	55.6 / 0.2	36.6 / 0.0	51.4 / 0.0	50.7 / 0.1	43.0 / 0.0	51.2	0.2
	CP `fmxcoder`	94.8 / 38.3	81.5 / 47.4	93.1 / 28.7	75.3 / 46.7	59.9 / 49.1	69.6 / 19.6	79.0	38.3
	TR `fmxcoder`	97.3 / 190.5	82.0 / 252.1	93.3 / 150.9	86.1 / 231.8	63.8 / 268.0	68.7 / 99.2	81.9	198.7

F LLM-as-a-Judge Procedure

For each latent we collect its top-activating tokens, sorted by maximum activation across the residual-stream activations of the base LLM on a held-out subset of its training corpus. We additionally include up to two context windows of 80 characters on each side of the activating token. The resulting prompt is sent to the LLM judge of choice via the OpenRouter API, with a system message instructing it to evaluate the latents. The returned `semantic_score` and `surface_score` are each clipped to $[0, 1]$, and a latent is labeled *semantic* when `semantic_score` > 0.7 and `surface_score` < 0.3, and *surface* with these conditions reversed. In the end, the results from different LLM judges are averaged. The full user-message template is shown below:

```
You are evaluating a latent feature from a sparse autoencoder trained on a language
model. You are given the tokens that most strongly activate this feature.

Score the feature on two independent dimensions:

**semantic_score** (0 to 1, continuous): How strongly do these tokens collectively
represent a coherent, high-level concept that is interpretable by humans?
- 1.0 = tokens clearly belong to a unified semantic category (e.g. US states and cities,
negative-emotion words, cooking verbs, animal species, financial terms)
- 0.5 = tokens share some thematic connection but it's loose or partial
- 0.0 = no discernible high-level concept; tokens seem unrelated or random

**surface_score** (0 to 1, continuous): How strongly do these tokens collectively
describe low-level linguistic or surface patterns rather than meaning?
- 1.0 = tokens are unified by syntax, morphology, punctuation, character class, or
formatting (e.g. closing braces, digits, -ing suffixes, markup tags, subword fragments)
- 0.5 = tokens partially share surface-level properties but also carry some semantic
content
- 0.0 = tokens are not unified by any surface-level pattern

These two scores are INDEPENDENT and can vary continuously from 0 to 1. A feature can be
high on both (rare), low on both (noise/random), or high on one and low on the other
(typical).

Think carefully about whether the unifying pattern is semantic (about meaning) or
surface-level (about form/syntax/characters).

**Feature ID**: {feature_id}
**Top-activating tokens** (ordered by activation strength): {tokens}

**Example contexts** (activating token wrapped in <<>>):
- token "{tok}":
    - {context_window}
    ...

Return ONLY valid JSON with keys "semantic_score" and "surface_score", both floats
between 0 and 1.
Example: {"semantic_score": 0.85, "surface_score": 0.1}
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Contributions C1–C5 (Section 1) are supported by Sections 4.2 to 4.5 and Sections B to F.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5: sensitivity to p and ranks, LLM-judge reliability, and $<$10% training overhead.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is empirical and contains no theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Method in Section 3, setup in Section 4.1, and all hyperparameters, ranks, layers, data and evaluation protocols in Section A; judge prompt in Section F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code (MIT license, with install, training and evaluation instructions) at `https://github.com/AndreasDemou/fmxcoders`; all base LLMs and datasets are public (Section 4.1).

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 and Section A (optimizer, learning rate, batch size, tokens, ranks, p, held-out splits, probing protocol); ablations in Sections C and D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Std over 5 seeds for `fmxcoder` (p=0.05) in Section 4.5, mean $\pm$ std across judges in Table 2, and coherence std in Figure 2; baselines and ablations are single runs due to compute.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section A (Compute and Reproducibility): one H100 64GB GPU per run, 12–42 h per run, $\sim$2000 GPU-hours in total.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Only public models and datasets are used, and no human subjects are involved.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: Not discussed: the work is foundational interpretability research aimed at understanding LLMs, though feature steering (Section 4.4) could in principle be used to manipulate outputs either for positive or negative impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No high-risk models or data are released, only code for training crosscoders on public LLMs.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models, datasets and libraries are cited (Sections 4.1 and 4.5, Section A), but their licenses are not stated explicitly.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The only new asset is the code, released under the MIT license with a README covering installation, training and evaluation; no new datasets or checkpoints.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Three LLM judges (DeepSeek-V4-Flash, GPT-5.4 nano, Gemini 3.5 Flash) label latents (Section 4.3); full procedure and prompt in Section F.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.