DUNE: DISENTANGLING POLYSEMANTIC NEURONS WITHOUT LOSING PREDICTIVE PERFORMANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks (DNNs) are widely used, but understanding what they actually learn remains difficult. A major obstacle is that individual neurons often respond to multiple unrelated concepts, obscuring the network's internal decision process. Existing approaches, such as sparse autoencoders, can disentangle these mixed signals into more meaningful, "monosemantic" features, but typically introduce approximation errors that may degrade downstream performance. Consequently, these methods exhibit a Pareto frontier between monosemanticity and functional faithfulness. To address this, we introduce **DUNE** (**d**isentangling **u**nits with **n**o **e**rror), a method that decomposes DNN neurons into more monosemantic subunits while preserving their original function by design. DUNE identifies concept-specific contributions from the preceding layer and re-routes them into separate subunits whose aggregation exactly recovers the original activation. DUNE can be applied directly to pretrained models, without labels or training of an auxiliary model. Across several models, including DINOv2, DUNE achieves state-of-the-art or competitive monosemanticity while exactly preserving functional faithfulness, substantially advancing the existing Pareto frontier between these properties. DUNE operates efficiently and enables fine-grained interventions such as concept-specific steering.

1 INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable performance across a wide range of domains, from language (Brown et al., 2020) over vision (Oquab et al., 2024) to scientific modeling (Jumper et al., 2021). Despite their success, these models remain difficult to understand and control, posing significant challenges for their deployment in safety-critical applications (Bommasani et al., 2021). Issues such as hallucinations (Ji et al., 2023; Liu et al., 2024), unsafe outputs (Wang et al., 2024; Zhang et al., 2024), and failures under distribution shift (Hendrycks et al., 2021; Hesse et al., 2026) highlight the need for methods that provide insight into model internals and enable reliable intervention (Bereska & Gavves, 2024; Sharkey et al., 2025).

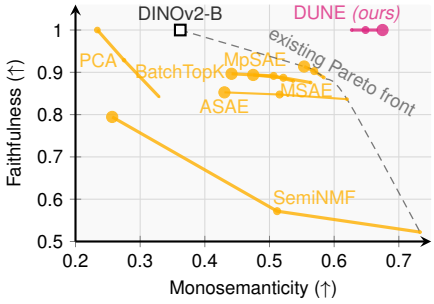


Figure 1: **Monosemanticity–faithfulness tradeoff.** We compare faithfulness and monosemanticity across disentanglement methods; marker size indicates the expansion factor. Existing approaches exhibit a clear Pareto front, where higher monosemanticity typically comes at the cost of lower faithfulness. In contrast, DUNE improves monosemanticity without sacrificing faithfulness, substantially advancing the Pareto front. See Section 4.1 for experimental details and detailed descriptions of the metrics.

A central goal in this context is *interpretability*: understanding how models represent and process information. However, DNNs are complex systems, where individual units (*e.g.*, neurons or channels) do not correspond cleanly to human-interpretable concepts. Instead, they often exhibit *polysemanticity*, responding to multiple unrelated concepts (Elhage et al., 2022). This obscures the internal structure of representations and makes it difficult to attribute model behavior to specific components.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

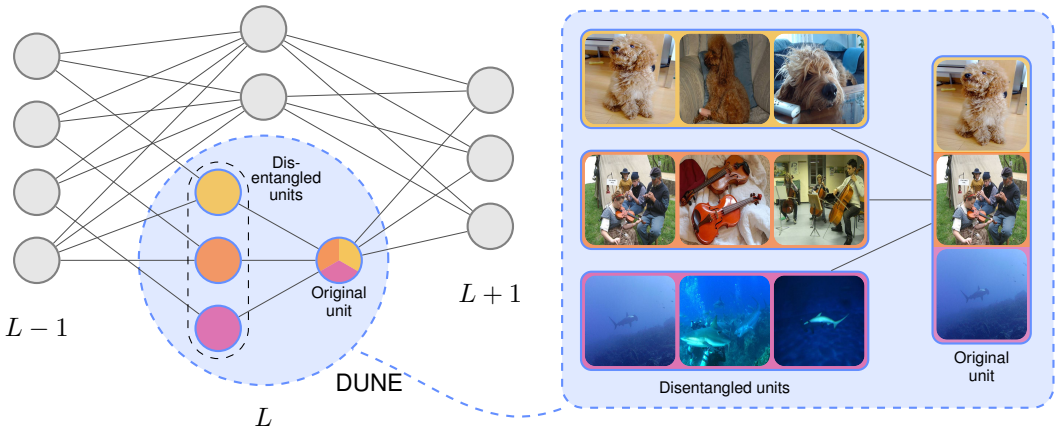


Figure 2: **DUNE for monosemantic disentanglement.** DUNE decomposes a polysemantic unit at layer L into more monosemantic subunits by restructuring incoming weights so that each sub-unit captures only a specific semantic concept. The sum of all subunits exactly recovers the original unit activation, ensuring perfect faithfulness. On the right, we provide a concrete example of this process by showing highly activating images for an actual neuron from the last layer of a DINOv2 model alongside its disentangled DUNE subunits. While the original neuron reacts to multiple semantically unrelated images, the disentangled units are more coherent and interpretable.

To overcome this limitation, recent work has explored disentangling internal representations into more interpretable components, aiming for *monosemanticity*, where individual units correspond to distinct semantic concepts. Among these, sparse autoencoders (SAEs) (Bricken et al., 2023; Huben et al., 2024) and transcoders (Dunefsky et al., 2024) have emerged as particularly prominent approaches, decomposing internal representations into sparse, interpretable features.

However, many recent approaches exhibit a practical tradeoff between monosemanticity and faithfulness, as visualized in Figure 1. As these methods rely on *learned* decompositions, they may introduce approximation errors and thus fail to exactly preserve the original model's behavior, limiting their applicability (Gao et al., 2025). In contrast, approaches that preserve functional faithfulness typically offer limited improvements in monosemanticity, do not yield explicit disentanglements that support fine-grained manipulation (O'Mahony et al., 2023; Dreyer et al., 2024), or require supervised training (Hesse et al., 2025), limiting their applicability or scalability.

Contributions. In this work, we introduce *DUNE* (**d**isentangling **u**nits with **n**o **e**rror), a method that mitigates this tradeoff between monosemanticity and faithfulness by achieving an explicit, lossless, and unsupervised disentanglement. Specifically, DUNE produces monosemantic and directly manipulable units *(explicit)*, exactly preserves the original model's behavior *(lossless)*, and requires no annotations *(unsupervised)*. It identifies monosemantic components within polysemantic units by uncovering structure in previous-layer attributions and restructuring model connections to isolate these components into independent, function-preserving subunits (*cf.* Figure 2). Due to its simplicity and lossless nature, DUNE can be applied post-hoc to a wide range of models, effectively providing improved monosemanticity without sacrificing downstream predictive performance.

Empirically, DUNE achieves state-of-the-art or competitive monosemanticity across a wide range of models and settings, substantially advancing the monosemanticity–faithfulness Pareto front (*cf.* Figure 1). We further show that its explicit, lossless decomposition enables fine-grained interventions, which we demonstrate through model steering in LLaVA-OneVision (Li et al., 2025).

2 Related Work

In the pursuit of gaining a deeper model understanding, *mechanistic interpretability* (Bereska & Gavves, 2024; Sharkey et al., 2025; Rai et al., 2024) aims to reverse-engineer the internal computations of neural networks. By exposing the units, directions, and circuits that carry meaningful computation, it promises a level of understanding that even supports targeted intervention, such as

Table 1: **Properties of existing disentanglement approaches.** We compare DUNE (shaded) against representative SAE variants and prior disentanglement methods across five properties: *stable* (gives a similar disentanglement on repeated runs), *explicit* (subunits can be inspected and modified directly, rather than only analyzed), *lossless* (the disentangled model produces exactly the same outputs as the original), *unsupervised* (no class labels needed) and *effective disentanglement* (produces subunits that are more semantically coherent than the original units). DUNE satisfies all five properties.

Method	Stable	Explicit	Lossless	Unsupervised	Effective Disentanglement
PCA (Pearson, 1901)	✓	✓	✓	✓	✗
Semi-NMF (Ding et al., 2010)	✗	✓	✗	✓	✓
Vanilla SAE (Bricken et al., 2023)	✗	✓	✗	✓	✓
Archetypal SAE (Fel et al., 2025)	✓	✓	✗	✓	✓
DNRwCV (O'Mahony et al., 2023)	✓	✗	✓	✓	✓
PURE (Dreyer et al., 2024)	✓	✗	✓	✓	✓
DPCiCNN (Hesse et al., 2025)	✓	✓	✓	✗	✓
DUNE *(ours)*	✓	✓	✓	✓	✓

steering (Turner et al., 2023; Templeton et al., 2024; Pach et al., 2025). However, a central challenge in mechanistic interpretability is *polysemanticity* (Elhage et al., 2022; Olah et al., 2020; Huben et al., 2024): individual units of modern neural networks frequently respond to multiple, semantically unrelated concepts rather than to a single coherent concept. This entanglement, often attributed to the superposition of more concepts than the layer has dimensions to allocate (Elhage et al., 2022), makes it difficult to attribute, explain, or steer model behavior at the level of individual units.

Recent work thus addresses polysemanticity by decomposing internal activations into more interpretable components. *Linear decomposition methods*, such as PCA, NMF, and Semi-NMF, decompose internal activations into linear components that can expose more monosemantic structure in the representation (Fel et al., 2023; Collins et al., 2018; Zhang et al., 2021; Fel et al., 2025). *Sparse autoencoders* (SAEs) (Bricken et al., 2023) train an overcomplete autoencoder to reconstruct activations as sparse combinations of learned features, and *transcoders* (Dunefsky et al., 2024) extend this idea to mappings between layers. Both produce an explicit disentanglement and are unsupervised but introduce a learned approximation that does not exactly preserve the original model's behavior. *DNRwCV* (O'Mahony et al., 2023) and its successor *PURE* (Dreyer et al., 2024) instead disentangle polysemantic units *"virtually"*: DNRwCV by finding concept-aligned directions in the activation space and PURE by clustering circuits. While both keep the model unchanged, they only yield "virtual" decompositions rather than explicit ones; hence, many applications such as feature-visualization and steering cannot be applied directly on the disentangled representations. Finally, *DPCiCNN* (Hesse et al., 2025) restructures the parameters of convolutional channels to produce explicit, function-preserving subunits, but requires supervision, introduces an uninterpretable residual neuron, and has so far been demonstrated only on convolutional architectures. Table 1 summarizes these approaches across the axes of being stable, explicit, lossless, unsupervised, and effective disentanglement.

To conclude, many existing approaches enable explicit and scalable unsupervised disentanglement, but incur information loss. In contrast, lossless disentanglement methods either lack an explicit disentanglement, achieve only mediocre monosemanticity, or do not scale to large vision foundation models, as they depend on supervision. Our method bridges this gap by providing an explicit, lossless, and unsupervised disentanglement while achieving strong monosemanticity.

3 DUNE: Disentangling Units with No Error

Given a probing dataset of N images $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ without annotations, a pre-trained vision backbone $f(\cdot)$, and a layer of interest L, our goal is to improve monosemanticity by disentangling polysemantic activation units a_u (*e.g.*, channels or neurons) into multiple, more monosemantic units, as previewed in Figure 2. Crucially, this is achieved without altering the overall function of $f(\cdot)$.

Our method decomposes a potentially polysemantic unit into a set of more interpretable subunits whose weights form an additive partition of the original unit, ensuring that their aggregated activation is exactly equal to the original (*cf.* Figure 2). By leveraging contribution structure in the preceding layer, we first identify inputs that strongly activate the unit to characterize its behavior. We then

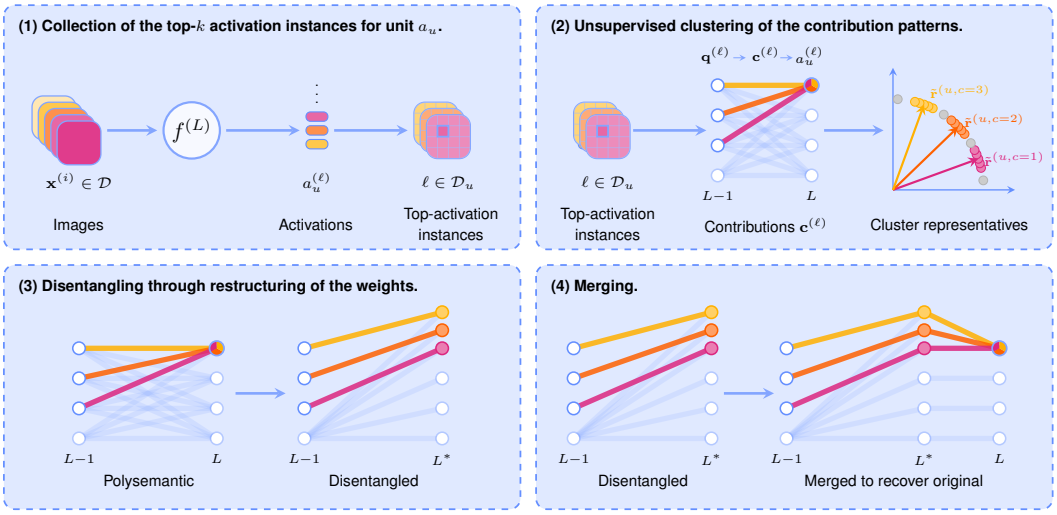


Figure 3: **Overview of DUNE.** *(1)* Embed every image $\mathbf{x}^{(i)}$ in the dataset $\mathcal{D}$ through layer L and consider the resulting activation instances ℓ (*e.g.*, spatial locations or tokens); select the top-k instances that activate unit a_u most strongly. *(2)* For each selected activation instance, compute a normalized contribution vector $\mathbf{c}^{(\ell)}$ from layer $L{-}1$ to $a_u^{(\ell)}$, revealing which upstream units drive the activation. Cluster these contribution vectors into concept groups, yielding K clusters represented by $\tilde{\mathbf{r}}^{(u,c)}$, one per concept encoded by a_u. *(3)* Replace a_u with K concept-specific units whose weights are constructed from the cluster representatives. *(4)* A summation operator reconstructs the original activation exactly.

analyze the corresponding contribution patterns in layer $L{-}1$ to uncover distinct groups of upstream units associated with different concepts. Using this structure, we disentangle the original unit into multiple subunits, each selectively connected to a concept-specific group of upstream units. Finally, we merge these subunits such that their aggregation exactly reconstructs the original unit's activation, preserving the model's function by construction.

More specifically, our method consists of four steps, which are illustrated in Figure 3: *(i)* collecting the top-k activation instances $\mathcal{D}_u$ from $\mathcal{D}$ for unit a_u that expose the concepts driving its activation (Section 3.1); *(ii)* clustering contribution patterns in layer $L{-}1$ to identify distinct concepts (Section 3.2); *(iii)* restructuring the weights between layers $L{-}1$ and L to produce a new layer L^* in which a_u is replaced by a set of subunits, each primarily connected to a concept-specific group of upstream units (Section 3.3); and *(iv)* merging the subunits to recover the original unit a_u (Section 3.4).

3.1 Collection of the top-k activation instances

To disentangle a unit of interest a_u in layer L, we first need to find the set of image patterns (respectively concepts) $\mathcal{D}_u$ for which it activates. To this end, we collect the top-k activation instances from a probing dataset $\mathcal{D}$ that activate a_u most strongly, as illustrated in Figure 3 *(1)*. Depending on the architecture, an activation instance is either a spatial location in a CNN feature map or a token in a Vision Transformers (ViT). The choice of the probing dataset $\mathcal{D}$ is task- and user-specific and defines the potential concepts that can be extracted; a common choice in related work is, *e.g.*, the ImageNet-1k dataset (Russakovsky et al., 2015; Dreyer et al., 2024).

Formally, we index all activation instances across the dataset by a single index $\ell \in \{1, \ldots, M\}$, where each ℓ corresponds to a specific pair (i, p) of image index i and spatial location or token index p; M is the total number of activation instances across the dataset. Let $\mathbf{a}^{(\ell)} \in \mathbb{R}^{d_L}$ denote the corresponding activation vector at target layer L, and let $a_u^{(\ell)}$ be the activation of unit u for that instance. We then select the top-k activation instances $\mathcal{D}_u = \left\{ \ell \;\middle|\; a_u^{(\ell)} \in \mathrm{TopK}\left(\{a_u^{(\ell')}\}_{\ell'}, k\right) \right\}$. Each $\ell \in \mathcal{D}_u$ is associated with an underlying image $\mathbf{x}^{(i)}$, which provides semantic context. Higher values of k

include a broader range of activation strengths and may capture a wider variety of concepts encoded by the unit. Note that several tokens of the same image can appear in the top-k activation instances since different regions of an image may represent multiple concepts. Such top-k set constructions have been proven successful in prior interpretability research. *E.g.*, PURE (Dreyer et al., 2024) and O'Mahony et al. (2023) both build similar reference sets before clustering, and maximally activating images are a common starting point in feature visualization more broadly (Olah et al., 2017).

The following steps leverage that for polysemantic units different concepts will appear among the image regions that activate it most strongly. For example, as in Figure 2, a unit might respond to image regions of dogs, violins, and oceans – visualized in yellow, orange, and pink in Figure 3.

3.2 Unsupervised clustering of the contribution patterns

Given a polysemantic unit, we now aim to identify the distinct concepts present in $\mathcal{D}_u$. To this end, we follow a simple assumption that has been established in prior work (Dreyer et al., 2024; Hesse et al., 2025): if two activation instances corresponding to distinct concepts activate the same unit in layer L, then different upstream circuits should have been used by the model. Instead of looking at the full circuits, common practice is to only consider the part of the circuit in layer $L{-}1$ (Dreyer et al., 2024; Hesse et al., 2025). Hence, for each of the selected activation instances ℓ, we now identify which units in the previous layer $L{-}1$ were responsible for activating unit $a_u^{(\ell)}$ of layer L. Concretely, we compute a *contribution vector* $\mathbf{c}^{(\ell)} \in \mathbb{R}^{d_{L-1}}$, where d_{L-1} is the number of units in layer $L{-}1$ and each entry measures how much one of those units contributed to the pre-activation of $a_u^{(\ell)}$.

Let $\mathbf{q}^{(\ell)} \in \mathbb{R}^{d_{L-1}}$ denote the activation vector at the preceding layer $L{-}1$. Since the contribution of the activation vector $\mathbf{q}^{(\ell)} \in \mathbb{R}^{d_{L-1}}$ in layer $L{-}1$ to the pre-activation $\tilde{a}_u^{(\ell)}$ reduces to a linear transformation for convolutional layers or MLP layers in ViTs, *i.e.*, $\tilde{a}_u^{(\ell)} = \mathbf{W}_u^\top \mathbf{q}^{(\ell)} + \mathbf{b}_u$, we can compute the exact contribution via $\mathbf{c}^{(\ell)} = \mathbf{W}_u \odot \mathbf{q}^{(\ell)}$ where $\odot$ denotes element-wise multiplication. We then L_2-normalize each contribution vector from all $\ell \in \mathcal{D}_u$ so that clustering in the next step focuses on *which* upstream units were involved, rather than *how strongly* they contributed. Doing this for all k selected activation instances gives us the contribution set $\mathcal{C}^{(u)} = \{\mathbf{c}^{(\ell)} \mid \ell \in \mathcal{D}_u\}$ for unit a_u.

If unit a_u is polysemantic, activation instances corresponding to different concepts will produce contribution vectors pointing in different directions, since they are driven by different upstream units. Coming back to our example of a unit activating for image regions corresponding to dogs, violins, and oceans, we would expect three distinct contribution clusters corresponding to each of the concepts. Hence, we next group the contribution vectors into clusters, with each cluster corresponding to one concept that unit a_u responds to. To this end, we cluster $\mathcal{C}^{(u)}$ using HDBSCAN (Campello et al., 2013), a density-based clustering algorithm that finds groups of points that are tightly packed together while leaving sparse or ambiguous points unclustered. Crucially, in contrast to k-means clustering (Hartigan & Wong, 1979), used for example in PURE (Dreyer et al., 2024), HDBSCAN does not require specifying the number of clusters K in advance. Instead, it infers K directly from the data. It also assigns each point a membership probability $p(\mathbf{c}^{(\ell)}) \in [0, 1]$, reflecting how confidently it belongs to its cluster. Each point is assigned to the cluster for which it has the highest membership probability, and this probability weights its contribution to the cluster representative (we ablate the effect of using HDBSCAN instead of k-means in Table 14).

For each cluster index $c \in [1, \ldots, K]$, we compute a single representative direction $\tilde{\mathbf{r}}^{(u,c)}$ by taking a normalized membership-probability-weighted average of the contribution vectors in that cluster:

$$\tilde{\mathbf{r}}^{(u,c)} = \frac{\mathbf{r}^{(u,c)}}{\|\mathbf{r}^{(u,c)}\|_2 + \epsilon}, \qquad \text{where} \qquad \mathbf{r}^{(u,c)} = \sum_{\mathbf{c}^{(\ell)} \in \mathcal{C}^{(u,c)}} p(\mathbf{c}^{(\ell)})\, \mathbf{c}^{(\ell)}. \tag{1}$$

The representative $\tilde{\mathbf{r}}^{(u,c)}$ summarizes which upstream units are most responsible for cluster (or concept) c, and will be used in the next section to construct the new concept-specific weights. The above steps are illustrated in Figure 3 *(2)*.

3.3 DISENTANGLING THROUGH RESTRUCTURING OF THE WEIGHTS

We now realize the discovered clusters as explicit subunits in a new layer L^*. The goal is to replace the polysemantic unit a_u with K new subunits $\{a_u^{(c)}\}_{c=1}^K$, one per concept, such that each new subunit responds selectively to its own concept. At the same time, we ensure that the behavior of the network remains unchanged, yielding a faithful decomposition.

Each concept subunit $a_u^{(c)}$ gets its own incoming weight vector $\mathbf{w}^{(u,c)}$ and bias $b_u^{(c)}$, constructed from the original weight vector $\mathbf{W}_u$, the original bias b_u, and the cluster representative $\tilde{\mathbf{r}}^{(u,c)}$. Specifically, for each upstream dimension j in layer $L{-}1$, we check whether one group of concept subunits clearly dominates the others in $\tilde{\mathbf{r}}_j^{(u,c)}$. If so, we assign that dimension exclusively to the winning group. If not, we treat it as shared and split the original weight equally across all concept subunits.

A set of concept indices $S \subseteq \{1, \ldots, K\}$ wins dimension j if the minimum score among its subunits exceeds the score of every concept subunit outside the set by a multiplicative margin $\rho \in (0, 1]$:

$$\rho \cdot \min_{c \in S} |\tilde{r}_j^{(u,c)}| > \max_{c' \notin S} |\tilde{r}_j^{(u,c')}|. \tag{2}$$

The smallest set of top-scoring subunits S that satisfies this condition is assigned dimension j exclusively, and the corresponding weight is distributed equally among the subunits in S. If no set wins clearly, the weight is split equally across all $a_u^{(c)}$ concept subunits:

$$w_j^{(u,c)} = \begin{cases} \dfrac{1}{|S|} W_{u,j} & \text{if } c \in S \text{ and } S \text{ wins dimension } j \text{ exclusively,} \\ \dfrac{1}{K} W_{u,j} & \text{otherwise.} \end{cases} \tag{3}$$

By construction, the concept subunit weights always sum back to the original: $\sum_{c=1}^K w_j^{(u,c)} = W_{u,j}$, and hence, for each upstream dimension j, subunit c receives a fraction $w_j^{(u,c)}/W_{u,j}$ of the original weight. We split the bias in proportion to the average of these fractions across all d_{L-1} dimensions: $b_u^{(c)} = \omega_c \, b_u$ with $\omega_c = \frac{1}{d_{L-1}} \sum_j w_j^{(u,c)}/W_{u,j}$, so that $\sum_{c=1}^K b_u^{(c)} = b_u$.

The margin ρ controls how aggressively dimensions are assigned exclusively to a group. We select it automatically by searching over all values in $\{0.05, 0.10, \ldots, 1.00\}$ and choosing the one for which each concept subunit activates most strongly on its own cluster and most weakly on the others (see Tables 16 and 17 for ablations).

Returning to our illustrative example, we might find that the third unit from the top in the preceding layer $L{-}1$ (■ in Figure 3) contributes particularly strongly on activation instances corresponding to oceans, and consequently, we connect it to the subunit corresponding to the ocean concept. Similarly, the fourth unit (bottom) in $L{-}1$ has no particularly high contribution to any activation instances, so we connect it to all the subunits. The process is visualized in Figure 3 *(3)*.

3.4 MERGING

Because the disentangled weights and biases form an additive partition of the originals, the activations of the concept subunits sum exactly to the pre-activation of the original unit, *i.e.*, $\sum_{c=1}^K a_u^{(c)} = a_u$ for all inputs. This allows us to directly reconstruct the original unit by summing the concept-specific subunits, ensuring that the overall function of the network remains unchanged, as illustrated in Figure 3 *(4)*.

4 EXPERIMENTS

We evaluate DUNE, focusing on the monosemanticity–faithfulness tradeoff (Section 4.1), model steering (Section 4.3), and practical considerations, such as stability and efficiency (Section 4.2).

4.1 MONOSEMANTICITY AND FAITHFULNESS

Experimental setup. We study the monosemanticity–faithfulness Pareto front of existing methods and DUNE, as previewed in Figure 1. We use the final pre-activation-function representation of

Table 2: **Monosemanticity *vs.* faithfulness.** We compare DUNE (shaded) against various linear, SAE, and transcoder baselines across three monosemanticity metrics and two faithfulness metrics on DINOv2-B/14 at expansion factor 8. We report mean and standard deviation across three seeded token probing datasets. The best method is shown in **bold** and the second best is underlined. DUNE simultaneously improves monosemanticity while preserving faithfulness up to numerical precision ($R^2 = 100\%$ and matching the backbone's accuracy). † denotes an expansion factor of 0.25.

		Monosemanticity			Faithfulness	
Model	**Method**	**LLM Rank** (↓)	**C-Score** (↑)	**SIM@1** (↑)	**Accuracy** (↑)	$\mathbf{R^2}$ (↑)
DINOv2 ViT-B/14	*Backbone only*	9	36.2	10.49 ± 0.06	84.3	100.00
	Linear baselines					
	PCA† (Pearson, 1901)	5	32.95 ± 0.60	14.49 ± 0.33	81.83 ± 0.04	84.17 ± 0.00
	Semi-NMF (Ding et al., 2010)	**1**	51.18 ± 0.00	25.67 ± 0.00	78.28 ± 0.00	57.17 ± 0.00
	Nonlinear baselines					
	Vanilla SAE (Bricken et al., 2023)	10	19.32 ± 0.26	10.79 ± 1.34	77.63 ± 0.01	47.76 ± 0.05
	TopK-SAE (Makhzani & Frey, 2014)	8	47.63 ± 0.30	26.65 ± 0.34	82.19 ± 0.07	88.58 ± 0.05
	JumpReLU (Rajamanoharan et al., 2024)	12	22.59 ± 0.16	8.13 ± 0.36	84.27 ± 0.02	99.97 ± 0.01
	BatchTopK-SAE (Bussmann et al., 2024)	6	50.64 ± 0.29	29.96 ± 0.16	82.06 ± 0.14	89.15 ± 0.01
	Matryoshka SAE (Bussmann et al., 2025)	7	52.14 ± 0.13	30.17 ± 0.20	82.13 ± 0.04	88.71 ± 0.02
	Archetypal SAE (Fel et al., 2025)	4	51.56 ± 0.52	33.01 ± 0.08	81.26 ± 0.08	84.75 ± 0.07
	Matching Pursuit (Costa et al., 2025)	3	56.92 ± 0.13	31.80 ± 0.21	82.86 ± 0.05	90.28 ± 0.01
	Transcoder (Dunefsky et al., 2024)	11	35.79 ± 2.05	17.16 ± 1.10	79.70 ± 0.81	82.17 ± 0.69
	DUNE *(ours)*	2	**64.85** ± 0.52	**34.09** ± 0.85	**84.29** ± 0.00	**100.00** ± 0.00

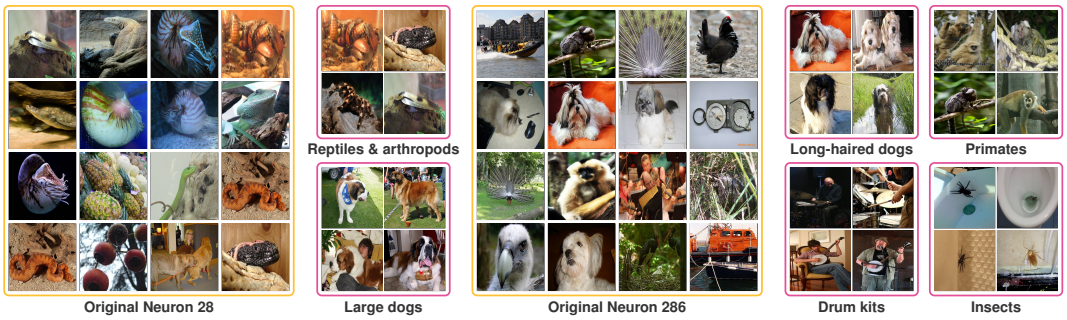


Figure 4: **Qualitative examples of DUNE disentanglements.** Unit 28 of DINOv2-B/14 (yellow frame, top-16) decomposed by DUNE into 2 (left) and unit 286 into 4 (right) subunits (pink frames, top-4 each). Labels are annotated by hand. Please refer to Section B.8 for additional examples.

a ViT-B/14-based DINOv2 vision backbone as the starting representation and compare DUNE to a broad set of linear, SAE, and transcoder baselines (see Table 2 for baselines). If applicable, we use an expansion factor of 8, *i.e.*, the size of the disentangled representation relative to the original one. For PCA (Pearson, 1901) we report the best-performing factor of 0.25 among $\{0.25, 0.5, 1\}$ (see Table 11), as its expansion factor cannot exceed 1. For all methods, we use the ImageNet-1k (Russakovsky et al., 2015) evaluation split for evaluation as described below and the training split as the probing dataset. To keep the activation cache tractable for training/clustering, we follow Pach et al. (2025) and retain only 2 randomly sampled tokens per image. We repeat the sampling three times to run each method independently on the resulting probing datasets, and report mean and standard deviation across runs.

We assess monosemanticity from two complementary perspectives to establish a clear neuron-concept correspondence: *(i)* the samples that highly activate a neuron should be semantically coherent, reflecting a consistent concept and *(ii)* given samples of the concept, the neuron should activate highly. We measure the first perspective, *i.e.* the *activation-to-concept* relationship, using a *Coherence Score* (C-Score) and an *LLM Rank*. Briefly, both metrics take the top-activating samples for a neuron and measure their semantic coherence (similarity) either using the model's internal representation (inspired by (Pach et al., 2025; Dreyer et al., 2025)) or an ensemble of three VLLM judges. For the second perspective, *i.e.* the *concept-to-activation* relationship, we use the *simulation metric* (SIM@1) from Oikarinen & Weng (2024). Here, each unit is described through a single concept to score how well this explanation predicts the unit's activation across the entire probing set. By

considering the full probing set, this metric also captures lower activation ranges, providing a more comprehensive assessment of monosemanticity than metrics restricted to top-activating samples from the first perspective. To assess faithfulness to the original function, we report the downstream classification *accuracy* when representations are replaced with method reconstructions and the reconstruction R^2 relative to the original representation. Metric details can be found in Section A.

Quantitative results. Results are reported in Table 2. As discussed in Section 1 and previewed in Figure 1, there is a clear tradeoff between monosemanticity and faithfulness among prior methods. In contrast, DUNE achieves competitive or even state-of-the-art results in the monosemanticity metrics while retaining perfect faithfulness as reflected by accuracy and R^2 scores that match the *backbone only* baseline (up to numeric precision). Remarkably, DUNE achieves the best SIM@1 score, despite this metric also considering lower activation regimes. Unlike TopK-based SAEs and Transcoders, DUNE does not threshold activations and therefore produces more weakly activating subunits, which we expected to reduce SIM@1. The strong performance suggests that DUNE remains interpretable even beyond its highest-activation regime. To conclude, DUNE consistently retains perfect faithfulness and hence downstream classification accuracy of the opaque baseline, whereas existing methods lose up to 7pp. At the same time, ours provides substantially more monosemantic neurons. As such, DUNE provides better monosemanticity without losing predictive performance

To strengthen these claims, we extend our evaluation across more metrics, layers, backbone sizes, model families, expansion factors, modalities (language), and ablations in Sections B and C.

Qualitative examples. We qualitatively compare two polysemantic DINOv2-B/14 units with their DUNE subunits in Figure 4. The original units respond to multiple concepts, whereas their subunits capture distinct, monosemantic concepts. Additional examples are provided in Section B.8.

4.2 Practical considerations

Computational efficiency. Table 7 shows the time and GPU memory used to process image batches with DUNE and the baselines. DUNE and Transcoder use the $3{,}072$-dim. input to the target layer whereas the SAEs use the 768-dim. input. Compared to Transcoders, DUNE takes 436 *vs.* 705 ms/batch and uses 577 *vs.* 954 MB during inference. DUNE also outperforms recent SAE variants. It uses less memory than Archetypal SAE (577 *vs.* $1{,}461$ MB) and runs faster than Matching Pursuit SAE (436 *vs.* $1{,}083$ ms/batch). Standard SAEs take 408–416 ms/batch and use about 558 MB.

Stability. A robust disentanglement should recover similar concepts when applied repeatedly to the same data. For learned feature dictionaries, variation across runs is a known issue (Paulo & Belrose, 2026a) that inspired stability-focused methods like Archetypal SAE (Fel et al., 2025). We assess stability by keeping the data fixed and comparing three runs of each method. The learned baselines use different initialization seeds; DUNE has no random initialization, so its three runs instead test whether its clustering produces consistent subunits. We match concept vectors between each pair of runs and measure their average similarity (Section A.6). Table 6 shows that DUNE achieves 91.66% similarity. This outperforms Archetypal SAE's 86.85%, all other SAEs and the Transcoder. Excluding the deterministic PCA (100%), DUNE recovers the most consistent subunits in this comparison.

4.3 Model steering

An application of disentangled representations is model steering, which changes a model's output by intervening on subunits associated with a target concept. We study whether the subunits identified by DUNE enable *targeted* steering, *i.e.*, removing a target concept from an image caption model while preserving the remaining scene description. We evaluate this task with LLaVA-OneVision (Li et al., 2025) on the COCO validation set (Lin et al., 2014). We use attribution patching (Syed et al., 2023; Kramár et al., 2024) to identify the subunits linked to the target concept. Specifically, we ask a binary question such as "Is there a cake in the image?" and compute the attribution of the DUNE subunits for the difference between the "Yes" and "No" logits. This difference measures the model's preference for the concept being present, quantifying each subunit's contribution. We suppress the highest-attributing subunits by setting their activations to $a_{\text{new}} = -s\, a_{\max}$, where s controls the intervention strength. Figure 5 illustrates this method.

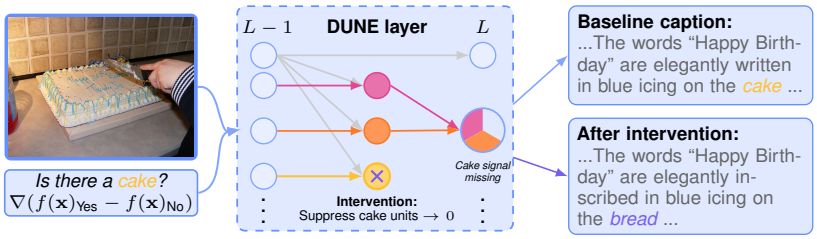


Figure 5: **Attribution patching for concept removal with DUNE.** Given an image and the query "Is there a cake?", we use the logit difference of "Yes" and "No" to identify DUNE subunits that support the target concept. We steer the top cake-attributing subunits (●) before merging their contributions into layer L. The resulting caption removes the target concept while retaining the remaining scene.

We compare DUNE with interventions on the original representation and the four best-performing SAE baselines from Table 2. We measure *target removal*, whether the edited caption omits the target concept, and *non-target retention*, whether the other concepts in the original caption are preserved. Figure 6 shows the trade-off. At multiple target-removal rates, DUNE-based steering preserves more non-target concepts than steering on the original neurons and the SAE baselines. This indicates that its subunits enable more targeted interventions, removing the target concept while causing fewer changes to the remaining caption. Full details and results are provided in Sections A.5 and B.4.

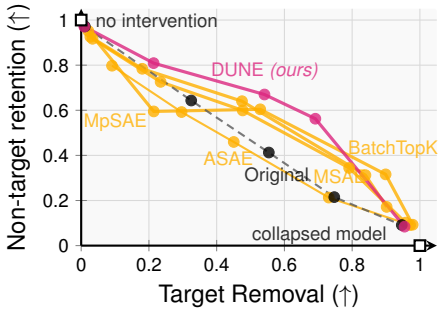


Figure 6: **Target removal *vs.* non-target retention on COCO.** We vary intervention strength and plot target removal against non-target retention. Points mark the setting that best balances target removal and non-target retention across multiple target-removal ranges per method. DUNE (●) achieves the best Pareto front relative to the tested baselines. See Section B.4 for full results.

5 Discussion

Limitations. In this work, we focus on linear layers and 1×1 convolutions, which cover a wide range of relevant layers; extending DUNE to attention heads and other architectural components remains future work. Since the disentanglement is run independently per neuron, subunits across different neurons may encode the same concept, creating redundancy at the layer level. This redundancy could be reduced through correlation-based pruning if required by the application, which however would sacrifice faithfulness. Relatedly, unlike SAEs, DUNE doesn't produce sparse or non-negative codes. Instead, its lossless linear disentanglement yields dense, partially negative activations, which may limit some aspects of interpretability. Nevertheless, our strong steering results show that concept-specific subunits can still be effectively identified and manipulated despite these properties. Moreover, monosemanticity is inherently hard to measure; like prior work, we rely on complementary proxy metrics that may not fully capture whether a unit represents a single human-interpretable concept (Dreyer et al., 2025; Hesse et al., 2025; Pach et al., 2025).

Conclusion. We introduce DUNE for disentangling polysemantic units into more monosemantic subunits. DUNE uses concept-specific structure in upstream circuits and restructures weights accordingly, ensuring that each subunit primarily connects to units in the preceding layer that respond to the corresponding concept. The disentanglement is constructed such that the sum of all subunits faithfully reconstructs the original neuron, enabling application without any loss in downstream predictive performance. Through a comprehensive empirical evaluation, we show that DUNE achieves near-perfect faithfulness (up to numerical precision) while often outperforming existing methods in terms of monosemanticity. We further demonstrate improved stability compared to SAEs and transcoders, and provide quantitative evidence that DUNE can be used for steering. Overall, we consider DUNE a promising addition to the interpretability toolkit: to the best of our knowledge, it is the first explicit disentanglement method to achieve state-of-the-art monosemanticity at scale without sacrificing any predictive performance, making it a strong foundation for future research.

AI USE STATEMENT

We used generative AI tools to assist with drafting and editing manuscript text and with writing and editing research code. We did not use them to develop the research idea, propose hypotheses, design the methodology or experiments, or analyze or interpret the results. The authors reviewed AI-assisted work and take responsibility for the final manuscript, code, and reported results.

ETHICS STATEMENT

Interpretability research of the kind presented here contributes to safer and more transparent AI: more monosemantic representations make it easier to audit models for spurious correlations, detect failure modes, and satisfy emerging regulatory requirements around model explainability. As with any interpretability method, however, a more precise understanding of feature structure could in principle assist adversarial attack design, and the concept-steering capability we demonstrate could be applied to manipulate model outputs in unintended ways. These risks are not unique to DUNE and are shared by mechanistic interpretability research broadly; they do not require mitigation beyond standard responsible-use practices for ML research artifacts.

REPRODUCIBILITY STATEMENT

We describe DUNE and the experimental comparisons in the main text. The appendix provides the metric definitions and evaluation procedures, including the judge prompts and steering protocol, as well as baseline training settings, datasets, model configurations, hardware, and software details. We report the data and initialization seeds used for repeated experiments so that the comparisons can be reproduced. Further, we will publish the code upon acceptance of the paper.

REFERENCES

Leonard Bereska and Stratis Gavves. Mechanistic interpretability for AI safety - A review. *TMLR*, 2024.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models. *OpenAI Blog*, 2023.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *arXiv:2108.07258 [cs.LG]*, 2021.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Bart Bussmann, Patrick Leask, and Neel Nanda. BatchTopK sparse autoencoders. In *NeurIPS*, 2024.

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. In *ICML*, 2025.

Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *PAKDD*, pp. 160–172, 2013.

Edo Collins, Radhakrishna Achanta, and Sabine Süsstrunk. Deep feature factorization for concept discovery. In *ECCV*, volume 11218, pp. 352–368. Springer, 2018.

Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba E. Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit. In *NeurIPS*, 2025.

Chris Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *PAMI*, 32(1):45–55, 2010.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Maximilian Dreyer, Erblina Purelku, Johanna Vielhaben, Wojciech Samek, and Sebastian Lapuschkin. PURE: Turning polysemantic neurons into pure features by identifying relevant circuits. In *CVPRW*, pp. 8212–8217, 2024.

Maximilian Dreyer, Jim Berend, Tobias Labarta, Johanna Vielhaben, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. Mechanistic understanding and validation of large AI models with semanticlens. *Nature Machine Intelligence*, 7:1572–1585, 2025.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. In *NeurIPS*, 2024.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger B. Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.

Thomas Fel, Agustin Martin Picard, Louis Béthune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. CRAFT: concept recursive activation factorization for explainability. In *CVPR*, pp. 2711–2721, 2023.

Thomas Fel, Ekdeep Singh Lubana, Jacob S. Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba E. Ba, and Talia Konkle. Archetypal SAE: Adaptive and stable dictionary learning for concept extraction in large vision models. In *ICML*, 2025.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *ICLR*, 2025.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, pp. 8320–8329, 2021.

Robin Hesse, Jonas Fischer, Simone Schaub-Meyer, and Stefan Roth. Disentangling polysemantic channels in convolutional neural networks. In *CVPRW*, pp. 4799–4803, 2025.

Robin Hesse, Dogukan Bagci, Bernt Schiele, Simone Schaub-Meyer, and Stefan Roth. Beyond accuracy: What matters in designing well-behaved image classification models? *TMLR*, 2026.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *ICLR*, 2024.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12):248:1–248:38, 2023.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

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

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing LLM behaviour to components. *arXiv:2403.00745 [cs.LG]*, 2024.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *TMLR*, 2025.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca D. Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv:2408.05147 [cs.LG]*, 2024.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, pp. 740–755, 2014.

Hanchao Liu, Wenyuan Xue, Yifei Chen, Dapeng Chen, Xiutian Zhao, Ke Wang, Liping Hou, Rongjun Li, and Wei Peng. A survey on hallucination in large vision-language models. *arXiv:2402.00253 [cs.CV]*, 2024.

Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. In *ICLR*, 2014.

Callum McDougall, Arthur Conmy, János Krámár, Tom Lieberum, Senthooran Rajamanoharan, and Neel Nanda. Gemma scope 2 - technical paper. *Google DeepMind Blog*, 2025.

Tuomas P. Oikarinen and Tsui-Wei Weng. Linear explanations for individual neurons. In *ICML*, volume 235, pp. 38639–38662, 2024.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020.

Christopher Olah, Ludwig Schubert, and Alexander Mordvintsev. Feature visualization. *Distill*, 2017.

Laura O'Mahony, Vincent Andrearczyk, Henning Müller, and Mara Graziani. Disentangling neuron representations with concept vectors. In *CVPRW*, pp. 3770–3775, 2023.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024.

Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge J. Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. In *NeurIPS*, 2025.

Gonçalo Paulo and Nora Belrose. Sparse autoencoders trained on the same data learn different features. *ICLR*, 2026a.

Gonçalo Paulo and Nora Belrose. Evaluating SAE interpretability without generating explanations. In *ICLR*, 2026b.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. In *ICML*, volume 267, 2025.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv:2407.02646 [cs.AI]*, 2024.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with JumpReLU sparse autoencoders. *arXiv:2407.14435 [cs.LG]*, 2024.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *EMNLP-IJCNLP*, pp. 3980–3990, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeffrey Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Isaac Bloom, Stella Biderman, Adrià Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, William Saunders, Eric J. Michaud, Stephen Casper, Max Tegmark, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. Open problems in mechanistic interpretability. *TMLR*, 2025.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, augmentation, and regularization in vision transformers. *TMLR*, 2022.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *NeurIPS ATTRIB Workshop*, 2023.

Gemma Team. Gemma 3 technical report. *arXiv:2503.19786 [cs.CL]*, 2025a.

Llama Team. The llama 3 herd of models. *arXiv:2407.21783 [cs.AI]*, 2024.

Qwen Team. Qwen3 technical report. *arXiv:2505.09388 [cs.CL]*, 2025b.

Qwen Team. Qwen3-VL technical report. *arXiv:2511.21631 [cs.CL]*, 2025c.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024.

Michael Tschannen, Alexey A. Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier J. Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv:2502.14786 [cs.CV]*, 2025.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv:2308.10248 [cs.CL]*, 2023.

Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. Do-not-answer: A dataset for evaluating safeguards in LLMs. In *EACL*, 2024.

Ross Wightman. Pytorch image models, 2019.

Kai Wittenmayer, Sukrut Rao, Amin Parchami-Araghi, Bernt Schiele, and Jonas Fischer. CFM: Language-aligned concept foundation model for vision. In *ECCV*, pp. 173–191, 2026.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-designing and scaling convnets with masked autoencoders. In *CVPR*, pp. 16133–16142, 2023.

Vladimir Zaigrajew, Hubert Baniecki, and Przemyslaw Biecek. Interpreting CLIP with hierarchical sparse autoencoders. In *ICML*, 2025.

Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A. Ehinger, and Benjamin I. P. Rubinstein. Invertible concept-based explanations for CNN models with non-negative concept activation vectors. In *AAAI*, pp. 11682–11690, 2021.

Yimeng Zhang, Jinghan Jia, Xin Chen, Aochuan Chen, Yihua Zhang, Jiancheng Liu, Ke Ding, and Sijia Liu. To generate or not? Safety-Driven Unlearned Diffusion Models Are Still Easy to Generate Unsafe Images ... For Now. In *ECCV*, pp. 385–403, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *NeurIPS*, 2023.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv:2504.10479 [cs.CV]*, 2025.

Appendix

A Metric definitions **16**

A.1 LLM Rank . . . 16
A.2 Coherence score . . . 16
A.3 Simulation metric (SIM@1) . . . 18
A.4 Reconstruction R^2 . . . 19
A.5 Quantitative steering benchmark . . . 19
A.6 Stability . . . 19
A.7 Computational efficiency . . . 20

B Results **20**

B.1 More polysemanticity metrics . . . 20
B.2 Stability . . . 20
B.3 Computational efficiency . . . 21
B.4 Quantitative steering benchmark . . . 21
B.5 PCA expansion factors . . . 22
B.6 Generalization . . . 22
B.7 Ablations . . . 23
B.8 Additional qualitative examples . . . 25

C Applying DUNE to Language models **26**

C.1 Experimental setup. . . . 26
C.2 Quantitative results. . . . 29
C.3 Qualitative results. . . . 30
C.4 Conclusion . . . 31

D Implementation Details **31**

D.1 Sparse autoencoders . . . 31
D.2 Linear baseline implementation . . . 32
D.3 Compute hardware . . . 32
D.4 Software stack . . . 32
D.5 Datasets . . . 33
D.6 Models and evaluation checkpoints . . . 33
D.7 Per-backbone disentanglement configurations . . . 33
D.8 Clustering hyperparameters . . . 33
D.9 Seeds . . . 33

A METRIC DEFINITIONS

This section gives the detailed definition of every metric used in Sections B and 4.

A.1 LLM RANK

The LLM Rank metric reported in Table 2, Table 12 and Table 13 ranks methods by how often three vision–language model (VLM) judges, used as proxies for a human evaluator (Zheng et al., 2023), prefer one method's disentangled units over another's in a pairwise coherence test. Each comparison shows two grids of top-activating images side by side and asks which is more coherent; methods are ranked by their consensus win rate across all such pairs.

Grid construction. For each unit, we build a 3×3 grid of its top-9 most-activating images from the ImageNet-1k evaluation split, at 336×336 pixels per cell. For backbones that produce token sequences, each image's activation score is the maximum unit activation across all tokens (matching the C-Score convention; *cf.* Section A.2). Figure 7 shows one such grid.

Judges. The three judges are Qwen3-VL-30B-A3B-Instruct (Team, 2025c) (Apache-2.0), Gemma-3-27B (Team, 2025a) (Gemma terms of use), and InternVL3-14B (Zhu et al., 2025) (MIT). Each is served through `vLLM` with `tensor_parallel_size` $= 1$, `max_model_len` $= 32\,768$, `limit_mm_per_prompt` $=$ `{image: 2}`, and a generation batch size of 64. Decoding is greedy: `temperature` $= 0$, `max_tokens` $= 256$, with a fixed seed of 42 passed to vLLM's `SamplingParams`.

Pairwise presentation. For each seeded token dataset, we sample 50 units from each method. For every pair of methods, we present the sampled units as 50 side-by-side comparisons. The paired units are not assumed to represent the same concept. We show each comparison again with the positions swapped to reduce position bias. With 12 methods, this yields $19{,}800$ presentations per VLM judge; the six-method generalization comparison yields $4{,}500$ per setting.

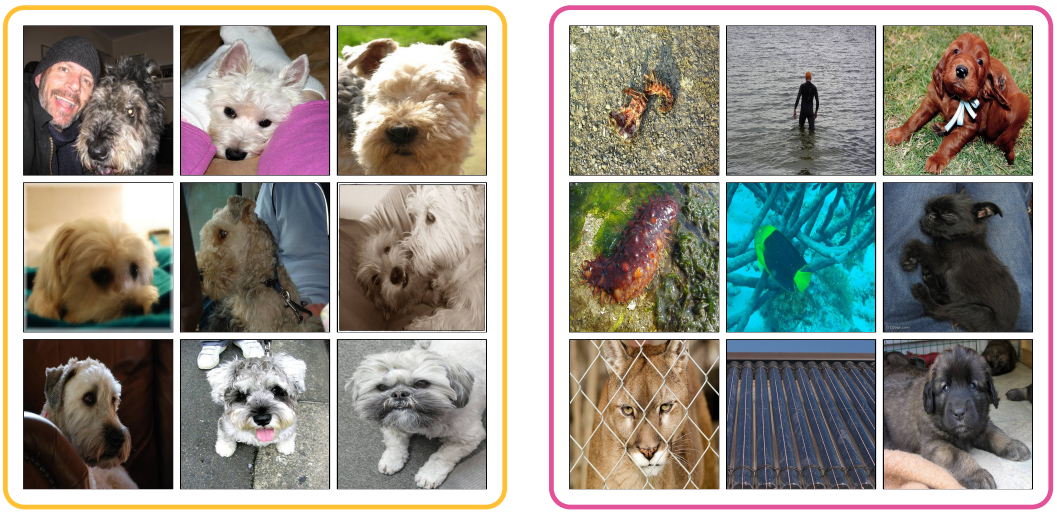
Aggregation and confidence intervals. Each pair of grids has a *consensus winner*: the option chosen by the majority of the three judges. Each method has a *consensus win rate*: the fraction of pairs it wins, pooled across the three seeded token datasets. Methods are ranked by this rate, and we attach 95% confidence intervals by bootstrapping the pairwise comparisons $1{,}000$ times. The ordinal LLM Rank in Table 2 is derived from these rates; the rates themselves and their intervals are listed in Table 3.

Prompt. All three VLM judges receive the identical model-agnostic prompt:

```
Each grid shows the 9 images that most strongly activate a
neural network unit.
A COHERENT unit captures a single, nameable concept: most
images share a common object category, object part, color,
texture, material, shape, or other visual or semantic
attribute.
An INCOHERENT unit shows images with no discernible shared
property.
Grid A = first image. Grid B = second image.
Describe what concept each grid captures (or why it seems
incoherent), then state which is more coherent.
Respond as JSON only:
{"reasoning_a": "...", "reasoning_b": "...", "answer":
"A" or "B"}
```

A.2 COHERENCE SCORE

To complement the LLM-based coherence metric, we also measure whether the images that most strongly activate a unit have similar representations. Like the monosemanticity score of Pach et al.



InternVL3-14B (PICKS: LEFT)
"*Grid A captures the concept of small dogs. Most images in this grid show different breeds of small dogs, indicating a coherent theme centered around this category.*"

"*Grid B is incoherent. The images show a variety of unrelated subjects, including a marine creature, a person in water, a dog, another marine creature, a fish, a sleeping kitten, a cat behind a fence, a roof, and a puppy. There is no single discernible shared property.*"

Qwen3-VL-30B-A3B (PICKS: LEFT)
"*Grid A is coherent because all nine images feature dogs. Although the breeds, sizes, and settings vary, the unifying concept is the animal itself. The images show different dogs in various poses and environments, but the shared object category is clearly 'dog'.*"

"*Grid B is incoherent because the images do not share a single, nameable concept. The images depict a wide variety of unrelated subjects: a dead crab, a person in water, a puppy, a sea cucumber, a fish, a sleeping kitten, a cougar, solar thermal tubes, and a puppy. There is no common object, color, texture, or semantic attribute that connects all nine images.*"

Gemma-3-27B (PICKS: LEFT)
"*Grid A predominantly features images of small, fluffy dogs, with variations in breed and color, but a clear focus on canine features. There's a strong visual similarity in terms of texture and general shape.*"

"*Grid B is much more diverse, containing images of a sea lion, a person in a wetsuit, a puppy, a sea creature, a fish, a black puppy, a lion, bamboo, and another black puppy. There's no obvious common theme or visual property linking these images.*"

CONSENSUS: Left

Figure 7: **Example pairwise grid shown to a VLM judge.** Each panel is a 3×3 collage of the top-9 most-activating images for one disentangled unit; two such panels from competing methods (left: DUNE, right: Archetypal SAE) are placed side by side and the three judges select the more coherent grid. Beneath the grids each judge's per-grid reasoning and the resulting consensus winner is shown. The example is from DINOv2-S/14.

(2025), C-Score compares images according to their representation similarity and gives more weight to images that activate the unit strongly. Our calculation differs in two ways: we evaluate only the most-activating images, and we measure their similarity in the backbone's own representation space before disentanglement, rather than with a separate image encoder.

Top-K selection. For each unit, we select its 100 highest-activating images from the ImageNet-1k evaluation split, using the unit's maximum activation across image positions. Evaluating a fixed set of 100 highly activating examples follows Dreyer et al. (2024); O'Mahony et al. (2023); unlike the score of Pach et al. (2025) over a broader image set, our C-Score focuses on the examples that most strongly activate each unit.

Table 3: **Per-method consensus win rate.** Fraction of pairwise comparisons in which a method was declared the winner under the majority-of-three VLM consensus (ties dropped), pooled across the three seeded token datasets. These are the continuous values backing the ordinal LLM Rank in Table 2 and **??**. Brackets give the 95% bootstrap confidence interval from 1,000 resamples of unit-pair comparisons. Same backbones, method order, and shading as Table 2 and Table 12.

		Consensus win rate (%)	95% CI
ImageNet-21k ViT-B/16	*Backbone only*	39.82	[38.1, 41.4]
	Linear baselines		
	PCA	42.27	[40.6, 44.4]
	Semi-NMF	46.55	[44.9, 48.1]
	Nonlinear baselines		
	Vanilla SAE	49.18	[47.4, 50.8]
	TopK-SAE	55.42	[53.7, 57.0]
	JumpReLU	21.52	[20.2, 22.9]
	BatchTopK-SAE	52.79	[51.3, 54.6]
	Matryoshka SAE	53.42	[51.6, 55.2]
	Archetypal SAE	62.79	[61.1, 64.4]
	Matching Pursuit	58.64	[56.9, 60.3]
	Transcoder	38.48	[36.8, 40.1]
	DUNE *(ours)*	**79.12**	[77.8, 80.4]
DINOv2 ViT-B/14	*Backbone only*	42.79	[41.0, 44.45]
	Linear baselines		
	PCA	53.91	[52.1, 55.6]
	Semi-NMF	**78.12**	[76.7, 79.5]
	Nonlinear baselines		
	Vanilla SAE	35.27	[33.6, 36.9]
	TopK-SAE	47.39	[45.7, 49.1]
	JumpReLU	27.27	[25.8, 28.8]
	BatchTopK-SAE	52.70	[51.0, 54.4]
	Matryoshka SAE	49.18	[47.5, 50.8]
	Archetypal SAE	55.82	[54.1, 57.4]
	Matching Pursuit	57.00	[55.3, 58.8]
	Transcoder	33.67	[32.0, 35.3]
	DUNE *(ours)*	66.88	[65.3, 68.5]

Per-image embedding. For each selected image, we localize the spatial position (or token) at which the unit attains its maximum activation, and extract the backbone's representation at that position. We use the *original (pre-disentanglement) backbone representation* – the same activation space in which the units live – rather than a fixed external vision model. This ensures the metric only measures separability of concepts already encoded in the model, aligning the metric with the capabilities of any disentanglement method that operates on this backbone. Each embedding is ℓ_2-normalized before scoring.

Weight construction. Each image i is weighted by the unit's activation at its chosen position, min–max normalized to $[0, 1]$ using the activation range observed across the entire evaluation split. Higher-activating images contribute more.

Score. Let $\{(e_i, w_i)\}_{i=1}^{K}$ be the ℓ_2-normalized embeddings and weights for the top-K images. The C-Score is the weighted mean off-diagonal pairwise cosine similarity:

$$\mathrm{C}(u) \;=\; \frac{\left\|\sum_i w_i e_i\right\|^2 \;-\; \sum_i w_i^2}{\left(\sum_i w_i\right)^2 \;-\; \sum_i w_i^2}\,. \tag{4}$$

The score lies in $[0, 1]$: 1 corresponds to all top-K embeddings being identical (perfectly monosemantic), 0 to them being orthogonal in expectation. We report the metric as a percentage. Units that never activate on the evaluation split are excluded from the per-method aggregation.

A.3 Simulation metric (SIM@1)

To test whether a concept explains a unit's behavior, we ask whether the concept can predict that unit's activation on images not used to choose the explanation. Following Oikarinen & Weng (2024),

Table 4: **Prompts for the quantitative steering benchmark.**

Purpose	Prompt
Caption generation	"Describe this image in detail, mentioning every distinct object you can see."
Attribution VQA	"Is there a `{concept}` in the image? Answer Yes or No."
Target judge	"Does this caption mention `{concept}` (or a close synonym)? Caption: `{caption}`. Answer YES or NO."
Coherence judge	"Is this caption grammatically correct and coherent? Caption: `{caption}`. Answer YES or NO."
Scene judge	"Caption A: `{caption A}`. Caption B: `{caption B}`. Do these describe the same overall scene? Answer YES or NO."

we select one concept per unit using training and validation images. The "@1" in SIM@1 denotes this one-concept explanation. We use SigLIP2-Base (`google/siglip2-base-patch16-224`) to estimate concept presence while selecting the explanation. On held-out images, a separate model, SigLIP2-SO400M (`google/siglip2-so400m-patch14-384`), estimates how strongly that concept is present. We use these estimates and a fitted weight to predict the unit's activation, then measure the correlation between predicted and actual activations across the held-out images. We average this correlation across units and report it as a percentage. Higher scores mean that a single concept better predicts when and how strongly a unit responds—not only for its most activating images, but across the test images. This complements the top-activating-image metrics used for the activation-to-concept perspective in Section 4.1.

A.4 Reconstruction R^2

We compare each reconstructed activation with the activation produced by the original layer. Let a_i be an original activation value, $\hat{a}_i$ its reconstruction, and $\bar{a}$ the mean of the original activation values. We compute $R^2 = 1 - \frac{\sum_i (a_i - \hat{a}_i)^2}{\sum_i (a_i - \bar{a})^2}$ over all evaluated images, tokens, and channels, and report $100R^2$ in the table. An exact reconstruction scores 100%; always predicting the mean scores 0%; a worse reconstruction can score below 0%.

A.5 Quantitative steering benchmark

We report target removal and non-target retention. For target removal, we ask Gemma-3-12B-IT (Team, 2025a) whether the edited caption mentions the target concept or a close synonym. Target removal is the fraction of captions for which the answer is "No." We evaluate only image–concept pairs for which the judge confirms that the original caption mentions the target concept.

Non-target retention is the fraction of other COCO concepts named in the original caption that remain in the edited caption. We use whole-word matching for this metric. We also report removal–retention utility, $\mathcal{U} = 100 r_{\mathrm{rem}} r_{\mathrm{ret}}$, where r_{rem} is target removal rate and r_{ret} is non-target retention rate.

The tested intervention settings and the procedure used to select points for the main-paper scatter plot are described alongside the full results in Section B.4.

Table 4 lists the prompts used for attribution and target removal.

A.6 Stability

This subsection details the cross-init stability protocol used in Section 4.2 and reported in Table 6, following Fel et al. (2025).

Setup. We evaluate stability on DINOv2-B/14 using the configurations from the main comparison: expansion factor 8 for the learned baselines and 0.25 for PCA. All three runs of a method use the same token dataset, selected with data seed 42. The learned baselines are fitted with initialization seeds $\{42, 7, 1024\}$ while keeping their data, hyperparameters, and training schedule fixed. We fit PCA three times on the same data; because it is deterministic, the resulting dictionaries should agree. DUNE has no random initialization, so we likewise repeat its construction three times on identical inputs to measure any variation in clustering. We exclude dead concept vectors before comparing dictionaries.

Measuring stability. We measure whether separate runs recover the same concept directions. A dictionary is a collection of vectors, but their order has no meaning: the first vector in one run need not represent the same concept as the first vector in another. Following Fel et al. (2025), we therefore find the one-to-one pairing that makes the vectors from two runs as similar as possible, then average the cosine similarity of the paired vectors. For PCA, Semi-NMF, the SAEs, and the Transcoder, vectors can be paired across the whole dictionary. See Fel et al. (2025) for further details of this matching metric.

DUNE's subunits belong to particular original neurons, so we pair subunits only when they come from the same neuron. The number of subunits for a neuron can differ between runs. When this happens, the implementation takes the first set of vectors up to the size of the smaller group, finds their best pairing, and lowers that neuron's similarity score to account for subunits without a counterpart. For example, if two runs produce five and six subunits and their paired vectors score 0.90, the additional penalty of one-sixth lowers the score to approximately 0.73. We average the scores across neurons and then across the three pairs of runs. The DUNE value in Table 6 includes this penalty; higher values indicate greater stability.

A.7 Computational efficiency

Protocol. We measure how long each method takes to process a batch of 256 images on one H100 GPU, and how much GPU memory it uses while doing so. Each method and seed runs in a fresh process. After 200 practice batches that prepare the model, we measure 500 further batches and report their average time. For memory, we report the largest amount occupied by PyTorch tensors during those 500 measured batches. This includes the model and inputs already in memory, but excludes temporary memory used earlier while loading and preparing the model. Thus, the memory column describes repeated inference, not the maximum memory needed to start the program.

All methods process the images through the backbone model. PCA and the SAEs take the activation at the target layer, reconstruct it, and pass that reconstruction through the remainder of the backbone. The Transcoder replaces the target layer's final projection, so the original projection is not run as well. DUNE likewise replaces that projection. It produces sub-unit activations and sums those belonging to each original neuron, restoring the layer's original output size before the rest of the backbone runs.

For methods that support it, we prepare the computation for repeated execution using `torch.compile` with CUDA graphs. This preparation happens before the measured batches, so the table reports the cost of repeatedly running the prepared model. Semi-NMF requires an additional calculation for every new image. It repeatedly searches for the values that best reconstruct that image's activation. Our Semi-NMF row does not time this optimization step. It measures only the final reconstruction step using placeholder values of the correct size. Its time and memory therefore understate the cost of using Semi-NMF on new images and should be interpreted cautiously.

B Results

B.1 More polysemanticity metrics

To complement C-Score and SIM@1, we adapt the Clarity and Polysemanticity measures of Dreyer et al. (2025). For each unit, we collect its 30 most activating examples and represent them using CLIP ViT-B/16 (Radford et al., 2021) CLS embeddings. Clarity is the mean pairwise cosine similarity between these embeddings. A high Clarity score indicates that the examples share a common semantic theme. For Polysemanticity, we partition the same embeddings into two k-means clusters and compute one minus the similarity between their centroids. A high Polysemanticity score therefore indicates that a unit responds to two semantically distinct groups, whereas a low score indicates more coherent behavior. As shown in Table 5, DUNE achieves the highest Clarity and the second-lowest Polysemanticity score.

B.2 Stability

Table 6 reports the per-method cross-init stability on DINOv2-B/14. The analysis is in Section 4.2; the measurement protocol is in Section A.6.

Table 5: **Additional monosemanticity metrics on DINOv2 ViT-B/14.** Following Dreyer et al. (2025), Clarity measures the similarity among each unit's 30 most-activating examples (higher is better); Polysemanticity measures the separation between two groups of those examples (lower is better). Both use CLIP ViT-B/16 image representations. Scores are percentages, reported as mean ± standard deviation across three seeded token datasets. DUNE's row is shaded; the best and second-best values in each column are bold and underlined.

		Monosemanticity	
Model	**Method**	**Clarity** (↑)	**Polysemanticity** (↓)
DINOv2 ViT-B/14	*Backbone only*	59.91	23.45
	Linear baselines		
	PCA	62.84 ± 0.11	23.45 ± 0.65
	Semi-NMF	55.90 ± 0.02	21.69 ± 0.05
	Nonlinear baselines		
	Vanilla SAE	54.80 ± 1.20	22.88 ± 0.37
	TopK-SAE	61.42 ± 0.19	16.97 ± 0.13
	JumpReLU	53.46 ± 0.40	19.95 ± 0.32
	BatchTopK-SAE	62.89 ± 0.10	16.53 ± 0.06
	Matryoshka SAE	63.53 ± 0.10	15.24 ± 0.04
	Archetypal SAE	64.82 ± 0.11	15.52 ± 0.17
	Matching Pursuit	<u>65.53</u> ± 0.13	**15.04** ± 0.04
	Transcoder	57.73 ± 0.37	20.62 ± 0.62
	DUNE *(ours)*	**67.98** ± 0.43	<u>15.07</u> ± 0.49

Table 6: **Stability.** Cross-init stability for all methods on DINOv2-B/14 at the same configuration as Section 4.1. Protocol details are in Section A.6. Highest similarity in **bold**, second-highest <u>underlined</u>.

		Stability
Model	**Method**	**Cross-Init Sim.** (↑)
DINOv2 ViT-B/14	*Linear baselines*	
	PCA	**100**
	Semi-NMF	77.31 ± 0.1
	Nonlinear baselines	
	Vanilla SAE	13.32 ± 0.01
	TopK-SAE	81.24 ± 0.44
	JumpReLU	13.38 ± 0.01
	BatchTopK-SAE	58.93 ± 0.29
	Matryoshka SAE	50.77 ± 0.12
	Archetypal SAE	<u>86.85</u> ± 0.58
	Matching Pursuit	78.08 ± 0.81
	Transcoder	65.07 ± 0.88
	DUNE *(ours)*	<u>91.66</u> ± 3.85

B.3 Computational efficiency

Table 7 reports per-method throughput and peak GPU memory on DINOv2-B/14 layer 12 at batch size 256 on a single H100, with the original backbone as reference. The analysis is in Section 4.2; the measurement protocol is in Section A.7.

B.4 Quantitative steering benchmark

We evaluate target-concept suppression in LLaVA-OneVision (Li et al., 2025) on COCO validation images (Lin et al., 2014). We form an image–concept pair when a COCO category is present in an image and mentioned in its original caption. We sample 80 pairs per retained category, following Wittenmayer et al. (2026). This gives 2,960 pairs across 37 categories. A sanity check excludes five pairs for which the judge does not identify the target concept in the original caption, since a concept cannot be removed if it is not present initially.

For each image–concept pair, we use attribution patching (Syed et al., 2023; Kramár et al., 2024) to rank features by their gradient×activation attribution for the difference between the "Yes" and "No" logits. We select the smallest set of positively attributing features that accounts for a fraction m of the

Table 7: **Computational Efficiency.** We measure time per batch (in millisecond per batch) and peak GPU memory during measured inference for methods on DINOv2-B/14 layer 11, batch size 256, single H100. Input dim is the size of the activation each method receives at the target layer. Means ± std. over three seeds; the original backbone is a reference. Semi-NMF measures reconstruction from placeholder values only, without the search required for new images, and is not directly comparable to the other methods. Full protocol: Section A.7.

			Comp. Efficiency	
Model	**Method**	**Input dim**	**ms/batch (↓)**	**Peak mem MB (↓)**
DINOv2 ViT-B/14	*Backbone only*	N/A	374.06 ± 0.72	501.80 ± 0.00
	Linear baselines			
	PCA (EF 1)	768	377.48 ± 0.21	506.52 ± 0.00
	Semi-NMF (EF 8, decode only)	768	387.44 ± 0.45	539.55 ± 0.00
	Nonlinear baselines			
	Vanilla SAE	768	407.60 ± 0.76	558.45 ± 0.00
	TopK-SAE	768	415.72 ± 0.31	558.45 ± 0.00
	JumpReLU	768	409.37 ± 0.48	558.48 ± 0.00
	BatchTopK-SAE	768	409.13 ± 0.23	558.45 ± 0.00
	Matryoshka SAE	768	409.42 ± 0.07	558.45 ± 0.00
	Archetypal SAE	768	418.22 ± 0.26	1461.28 ± 0.00
	Matching Pursuit	768	1083.01 ± 1.24	539.55 ± 0.00
	Transcoder	3072	705.42 ± 1.16	953.84 ± 0.00
	DUNE *(ours)*	3072	436.44 ± 3.62	577.49 ± 4.33

total positive attribution. We evaluate $m \in \{20\%, 40\%, 60\%, 80\%, 100\%\}$. For steering the selected neurons, we set their activations to

$$a_{\text{new}} = -s\, a_{\max},$$

where s is the steering factor and $a_{\max}$ is the feature's maximum activation on the COCO validation set. We evaluate $s \in \{0, \pm 0.5, \pm 1, \pm 2.5, \pm 5\}$. For the no-intervention condition, we use the reconstructed activation, $a_{\text{new}} = a_{\text{recon}}$.

Tables 8 to 10 report removal–retention utility, non-target retention, and target removal, respectively. Setting $s = 0$ with DUNE achieves the highest removal–retention utility: 36.25 at $m = 80\%$ and 38.89 at $m = 100\%$.

Scatter-plot construction. For Figure 6, we divide each method's evaluated settings into target-removal bins of width 0.33 and show the setting with the highest removal–retention utility in each bin, where $\mathcal{U} = 100 r_{\text{rem}} r_{\text{ret}}$. Here, r_{rem} is the fraction of target concepts removed, and r_{ret} is the fraction of non-target concepts retained. We also show the setting with the highest removal–retention utility in the 0.9–1.0 interval and the no-intervention condition defined above.

B.5 PCA EXPANSION FACTORS

Table 11 reports PCA at expansion factors 0.25, 0.5, and 1. Retaining more components improves reconstruction and downstream accuracy, reaching exact reconstruction at expansion factor 1, but reduces C-Score and SIM@1. Thus, PCA is lossless only at full rank; lower-rank PCA can improve SIM@1 only by discarding information.

B.6 GENERALIZATION

To assess whether our findings extend beyond the DINOv2-B/14 setting, we consider two complementary evaluations. Table 12 repeats the complete comparison on the supervised ImageNet-21k ViT-B/16, including PCA and Semi-NMF, alongside the SAE and transcoder baselines. Table 13 then evaluates DUNE across layers, backbone sizes, model families, and expansion factors. Given the breadth of this evaluation, we compare DUNE with the four strongest SAE baselines from the primary comparison across all settings. Both tables report LLM Rank, C-Score, SIM@1, downstream accuracy, and R^2. The analysis is in Section 4.1; metric definitions are in Section A.

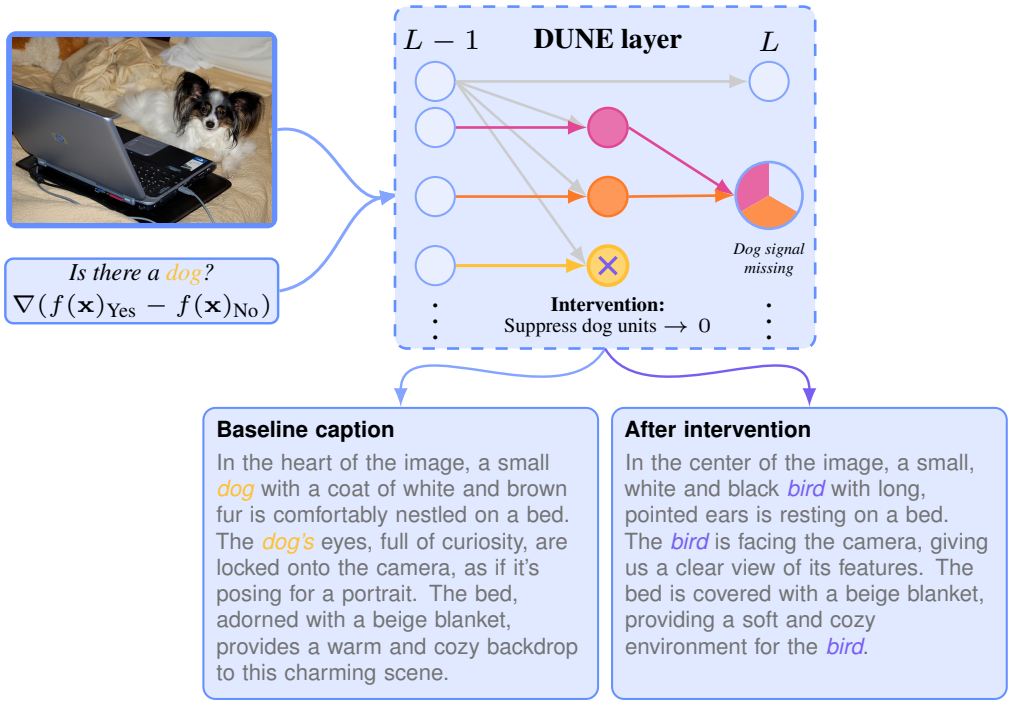


Figure 8: **Additional dog-suppression example with DUNE.** For a dog scene, we use the query "Is there a dog?" to identify dog-supporting DUNE subunits. Suppressing these subunits removes the dog, while keeping a description of the scene and replacing the dog with a bird.

B.7 Ablations

We evaluate two components of DUNE's construction on DINOv2-B/14, layer 12. DUNE obtains subunits by clustering each neuron's contribution patterns and restructuring its weights. The following ablations isolate the roles of feature construction and the clustering algorithm. The fixed-width alternatives use expansion factor 8; HDBSCAN determines the number of subunits from the data. We use expansion factor 8 to align them with the configuration in Table 2.

Feature construction. We compare DUNE with three alternatives. The *Random* control partitions each neuron's weights into random groups without using its contribution patterns. This tests whether an increase in the number of units alone accounts for the improvement. The other two alternatives cluster raw activations with either HDBSCAN or k-means. They use the resulting centroids as feature directions, compute feature activations by dot product, and reconstruct the original representation with a pseudoinverse decoder. Thus, these alternatives change how features are constructed, rather than only changing the clustering algorithm.

Clustering algorithm. To isolate this choice, we retain DUNE's contribution patterns and weight-restructuring procedure and replace HDBSCAN with k-means. The DUNE row represents the HDBSCAN configuration from Table 2.

Table 14 shows our ablation results. Random splitting performs substantially worse across all three metrics, indicating that gains do not solely result from expansion. The dot-product variants achieve higher C-Scores, which is expected because their directions are constructed from centroids in the same backbone representation space used by C-Score. However, this advantage does not transfer to the complementary metrics. DUNE ranks first under VLM judgments and substantially outperforms both dot-product variants on SIM@1. Moreover, unlike the dot-product variants with pseudoinverse decoding, DUNE preserves the original representation exactly by construction. Overall, DUNE provides the strongest balance between semantic coherence, concept-to-activation predictability, and

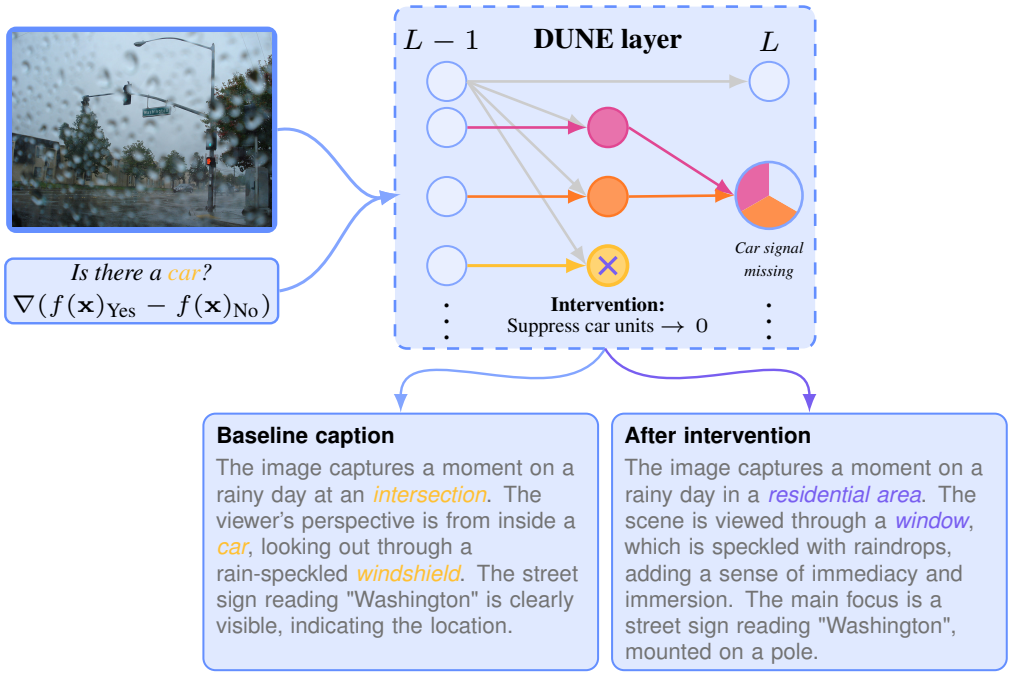


Figure 9: **Additional car-suppression example with DUNE.** For a rainy street scene, we use the query "Is there a car?" to identify car-supporting DUNE subunits. Suppressing these subunits removes the car, windshield, and intersection description from the generated caption, while retaining a description of the rainy scene and the Washington street sign.

faithfulness. Within the attribution-based construction, HDBSCAN improves LLM Rank and C-Score over k-means, while k-means performs slightly better on SIM@1.

HDBSCAN hyperparameter ablation We ablate DUNE's clustering choices, one hyperparameter at a time, around the default configuration (HDBSCAN `cluster_selection_method`, `min_cluster_size`, `min_samples`, and UMAP pre-reduction), evaluated by C-Score on DINOv2-B/14 layer 12. Because R^2 and downstream accuracy are exact by construction regardless of clustering choices. We omit the LLM Rank due to limited computational budget. A random neuron disentanglement baseline confirms that the gains are driven by the structure in the attribution vectors, not the expansion factor alone. Results in Table 15 show DUNE is robust across all axes. Table 15 reports C-Score on DINOv2-B/14 for the ablation axes swept around the DUNE baseline (HDBSCAN cluster-selection method, `min_cluster_size`, `min_samples`, UMAP dimensionality reduction) at expansion factor 4. We run the broader hyperparameter sweep at an effective expansion factor of approximately 4 to keep the cost of evaluating multiple clustering settings manageable. The baseline clustering hyperparameters are listed in Section D.8.

ρ selection and ablation. For each split neuron, we select ρ independently from $\{0.05, 0.10, \ldots, 1.00\}$. For every candidate value, we construct the corresponding subunits and measure, for each cluster, how much more strongly its assigned subunit activates than the other subunits on examples from that cluster. We select the value that maximizes this average separation across the neuron's clusters (see Equation (2)). The resulting selection distribution is reported in Table 16.

To assess the effect of this selection, we repeat the DINOv2-B/14, configuration from Table 15 while fixing one value of ρ for all neurons. In this configuration, clustering divides 673 original neurons into multiple subunits; neurons for which it finds no split remain unchanged. Because, clustering is independent of ρ, all runs split the same 673 neurons; only the assignment of the original weights to subunits changes. Faithfulness remains exact in every setting, since the subunit weights always sum to the original weight.

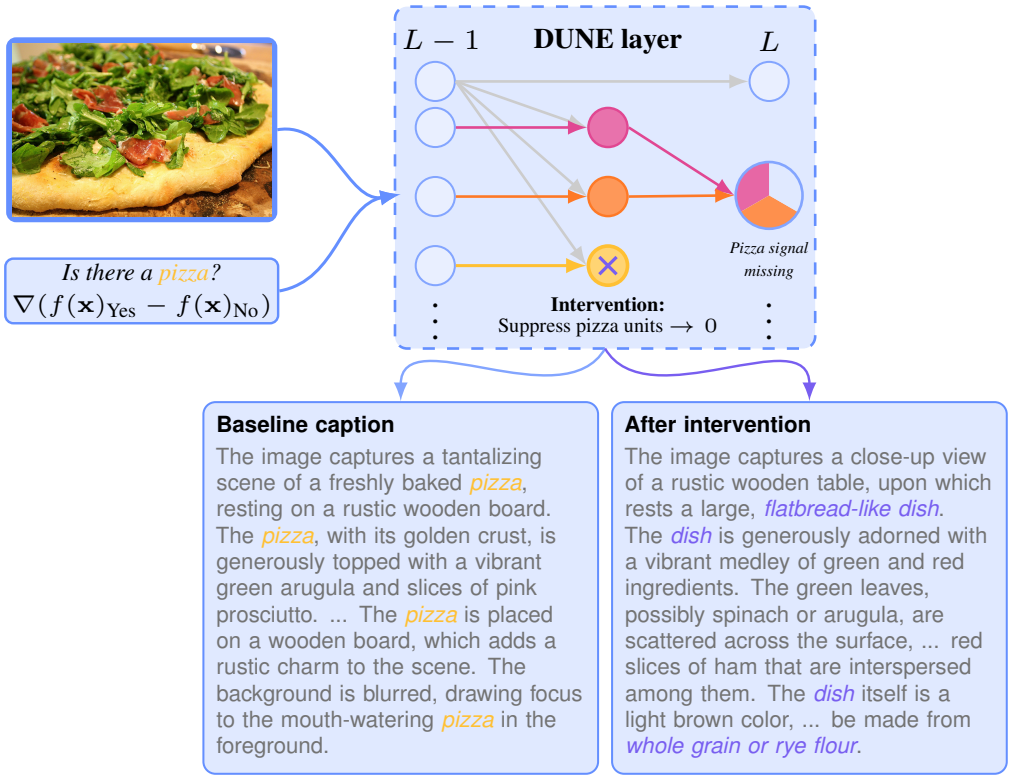


Figure 10: **Additional pizza-suppression example with DUNE.** The original caption identifies the food as a pizza with arugula and prosciutto. Removing the pizza-attributing DUNE subunits instead yields a generic flatbread description, while preserving details of the wooden board and its green and red toppings.

The selected margins span the full candidate grid, confirming that no single value is preferred across all neurons. When ρ is fixed globally, C-Score remains relatively stable from $\rho = 0.05$ to $\rho = 0.95$, ranging from 61.86 to 64.51, but drops sharply to 50.07 at $\rho = 1.00$. Although $\rho = 1.00$ is selected for some individual neurons, applying it globally makes every split maximally exclusive and is detrimental overall. The best fixed value, $\rho = 0.20$, can only be identified in hindsight. More generally, selecting an optimal per-neuron assignment through validation would require considering 20×673 possible combinations in this setting, while a global value would need to be retuned for each model and configuration. The automatic rule instead selects ρ locally for each neuron without model-level tuning.

B.8 Additional qualitative examples

Figure 4 shows representative DUNE disentanglements on DINOv2-B/14 (a 2-way and a 4-way split). The number of subunits is not a fixed hyperparameter – HDBSCAN discovers it automatically per unit (*cf.* Section 3.2) and varies neuron-by-neuron depending on how many distinct concepts drive the unit's activation. Some units produce a single cluster (effectively monosemantic already and not split further); others split into many. To illustrate the range we observe in practice, we show one representative example for each of $K \in \{1, 3, 5, 6, 7, 8\}$ subunits in Figures 13 to 18. The format mirrors that of Figure 4: the left panel shows the top-16 activating images of the parent unit, and each pink panel on the right shows the top-4 activating images for one DUNE subunit. Concept labels above each subunit panel are short human-assigned descriptions of the visible theme. All examples use the same configuration as Section 4.1: DINOv2-B/14, layer 12, TopK $= 400$k, seed 42.

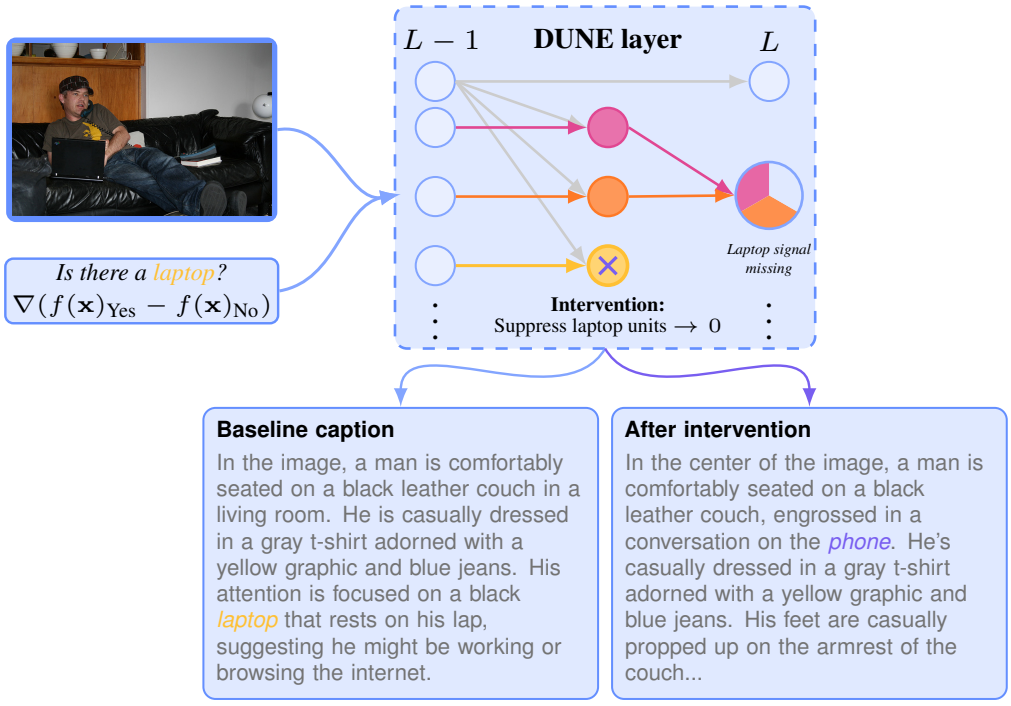


Figure 11: **Additional laptop-suppression example with DUNE.** For an image of a man seated on a couch, we use the query "Is there a laptop?" to identify laptop-supporting DUNE subunits. Suppressing these subunits removes the laptop and related work-or-browsing description from the caption, while retaining the man, couch, clothing, and living-room context.

C Applying DUNE to Language models

We assess whether DUNE transfers from vision to language models without changing its construction. We apply it to MLP down-projection output neurons in two instruction-tuned models, Gemma 3 270M (Team, 2025a) and Qwen3 0.6B (Team, 2025b), similar to Lieberum et al. (2024); McDougall et al. (2025). For each neuron, we collect activating text examples, cluster the corresponding contribution patterns, and split its incoming weights into subunits whose sum preserves the original neuron. We limit DUNE runs to approximately two billion tokens per model to keep its computational cost manageable.

C.1 Experimental setup.

We study layer 16 of Gemma 3 270M-IT (Team, 2025a), then repeat the evaluation on Gemma layer 13 and Qwen3 0.6B layer 21 (Team, 2025b). We compare DUNE with the backbone-only reference and four self-trained methods: Archetypal SAE (ASAE) (Fel et al., 2025), BatchTopK-SAE (Bussmann et al., 2024), Matching Pursuit SAE (MpSAE) (Costa et al., 2025), and Matryoshka SAE (MSAE) (Bussmann et al., 2025). We chose these four because they were the strongest SAE baselines in our main vision experiments. For Gemma, we also evaluate published GemmaScope2 SAEs and transcoders (McDougall et al., 2025). To compare methods with similar feature capacity, we use GemmaScope2 width-16k checkpoints at three sparsity levels (expansion factor 25.6), self-trained SAE checkpoints at expansion factor 25, and the DUNE checkpoint closest to that factor for each layer.

We assess monosemanticity using four complementary evaluations. AutoInterp (Paulo et al., 2025) generates an explanation of a feature from activating examples and tests whether that explanation predicts activity on new examples. Simulation (Bills et al., 2023) tests whether an explanation predicts *how strongly* the feature activates across a new passage. Intruder (Paulo & Belrose, 2026b) asks a language-model judge to identify a non-matching passage among four activating passages.

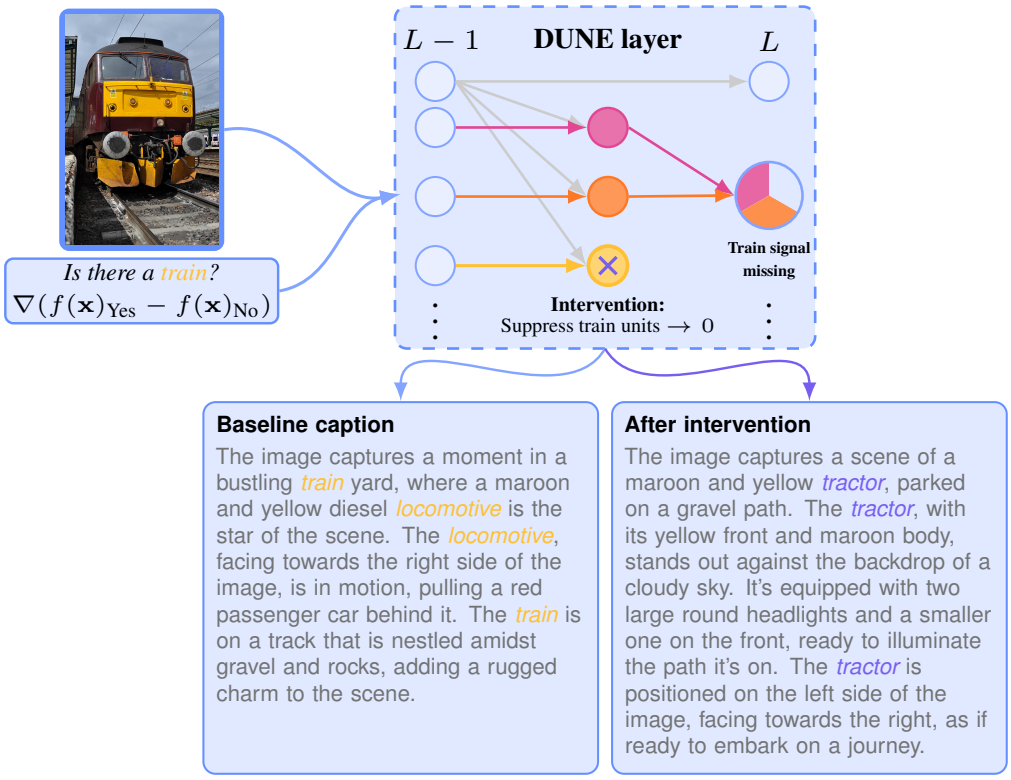


Figure 12: **Additional train-suppression example with DUNE.** The original caption interprets the maroon-and-yellow vehicle as a train and describes its locomotive, passenger car, and tracks. Once train-supporting DUNE subunits are suppressed, the same visual features are instead described as a tractor on a gravel path, while preserving its color, orientation, and outdoor setting.



Figure 13: $K = 1$**: an already-monosemantic unit.** HDBSCAN finds only a single cluster in the unit's contribution-vector set, so DUNE leaves the unit unchanged.

Embedding (Paulo & Belrose, 2026b) tests whether activating and comparison passages can be distinguished using an independent text encoder. Intruder and embedding complement explanation-based scores because they do not depend on a generated explanation or on the choice of the language model that generates it. We separately report reconstruction R^2 and next-token KL divergence to assess how faithfully each representation preserves the original model's computation. The KL divergence compares the model's predicted distribution over the next token with that of the unchanged backbone on the same text; lower values mean that the predictions are more similar, and zero means they are identical.

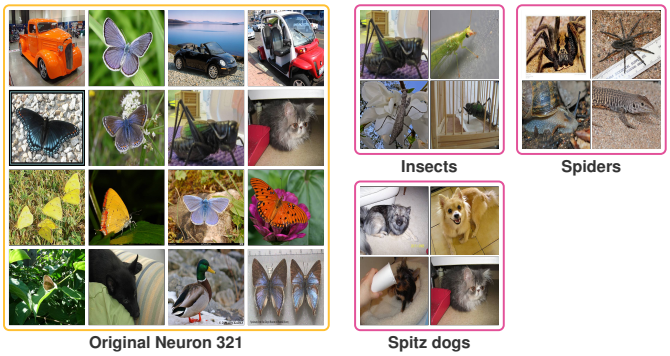


Figure 14: **3-way DUNE disentanglements.** A polysemantic DINOv2-B/14 unit (yellow frame, left; top-16) decomposed by DUNE into 3 subunits (pink frames, right; top-4 each).



Figure 15: **5-way DUNE disentanglements.** A polysemantic DINOv2-B/14 unit (yellow frame, left; top-16) decomposed by DUNE into 5 subunits (pink frames, right; top-4 each).

We use Gemma 3 12B Instruct (Team, 2025a) (`google/gemma-3-12b-it`) to generate explanations and make judge decisions for AutoInterp and Intruder. Simulation uses Llama 3.1 8B (Team, 2024) (`meta-llama/Llama-3.1-8B`) to predict activation magnitudes. Embedding uses `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019) and does not use a language-model judge.

The monosemanticity evaluations use the same fixed two-million-token Pile-validation corpus. Each metric samples up to 1,000 eligible features. Reconstruction follows the separate OpenWebText(Gokaslan et al., 2019) setup used by SAEBench (Karvonen et al., 2025).

We report the original AutoInterp, Intruder, and embedding protocols alongside variants with activation thresholds of 25%, 50%, and 75% of each feature's observed maximum in Figures 19 to 21 alongsinde the reconstruction R^2 on the x-axis. The original AutoInterp protocol uses a 1% threshold. For denser representations such as DUNE, this low threshold can label weak responses as active even when they differ substantially from the feature's strongest responses. The threshold variants let us examine how this choice affects the scores. For Intruder and embedding, an activating 128-token block must have a peak activation above the specified cutoff. In the original evaluation, a comparison passage must have zero activation of the tested feature; in the threshold variants, its activation may be nonzero but must remain at or below the cutoff. Each metric and threshold selects its eligible features separately.

We construct only one DUNE checkpoint and one checkpoint per self-trained SAE configuration because repeating checkpoint construction would exceed our computational budget. We repeat evaluation with seeds 42, 43, and 44, which resample the evaluated features and examples, and report the mean and standard deviation across these runs. These deviations reflect evaluation sampling, not variation across independently constructed checkpoints.



Figure 16: **6-way DUNE disentanglements.** A polysemantic DINOv2-B/14 unit (yellow frame, left; top-16) decomposed by DUNE into 6 subunits (pink frames, right; top-4 each).



Figure 17: **7-way DUNE disentanglements.** A polysemantic DINOv2-B/14 unit (yellow frame, left; top-16) decomposed by DUNE into 7 subunits (pink frames, right; top-4 each).

C.2 Quantitative results.

Table 18 compares methods with similar numbers of features on Gemma layer 16 under the original evaluation protocols. It reports AutoInterp, Intruder, embedding, Simulation, reconstruction R^2, and next-token KL divergence. Tables 19 and 20 repeat the comparison on Gemma layer 13 and Qwen layer 21 as generalization experiments.

On Gemma layer 16, DUNE achieves 100.0% reconstruction R^2, and its next-token KL divergence rounds to 0.00 at the table's displayed precision. It also obtains the highest Simulation score (8.3). Some SAEs and transcoders outperform DUNE on AutoInterp, Intruder, and Embedding under the evaluation protocols of SAEBench (Karvonen et al., 2025) and Paulo & Belrose (2026b). Together, these results show that DUNE produces interpretable subunits while preserving the backbone's computation.

The scatter plots in Figures 19 to 21 show that the definition of an active feature matters. At the 25% activation threshold, DUNE reaches an AutoInterp score of 85.5, exceeding every compared method at that threshold while retaining exact reconstruction. At the 75% threshold, its Intruder score reaches 38.3, however, it still lacks behind BatchTopK SAE's 42.63 which, however, has a lower reconstruction R^2 of 94.35%. DUNE also improves the Embedding score compared to the original model, from 69.5 to 77.3, but remains below the strongest baselines. This sensitivity to activation thresholds appears to be more pronounced in the language-model evaluation than in our vision experiments, where DUNE scored well on SIM@1 despite that metric considering lower-activation examples (Section 4.1). Furthermore, DUNE achieves the highest language-model Simulation score, which evaluates predictions of activation strength across new text Table 18. Weak activations are therefore not inherently a problem for DUNE; rather, the language-model scores depend more strongly on where the evaluation draws the line between active and inactive features.

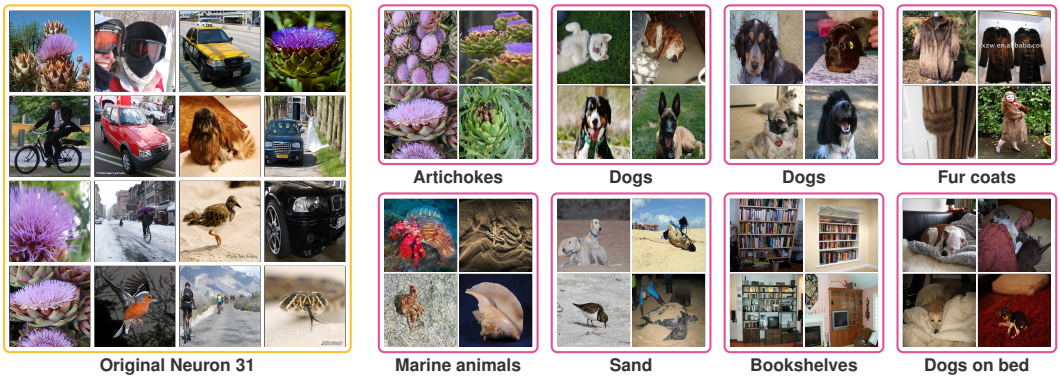


Figure 18: **8-way DUNE disentanglements.** A polysemantic DINOv2-B/14 unit (yellow frame, left; top-16) decomposed by DUNE into 8 subunits (pink frames, right; top-4 each).

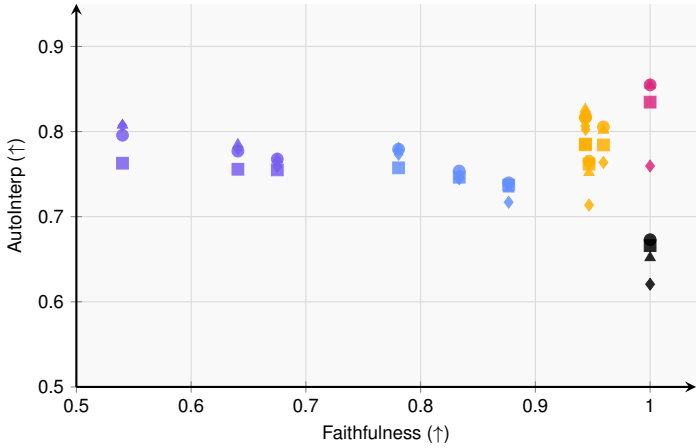


Figure 19: **AutoInterp versus reconstruction on Gemma 3 270M-IT, layer 16.** Diamonds, triangles, circles, and squares denote the original evaluation (◆) and activation thresholds of 25% (▲), 50%(●), and 75%(■), respectively. Colors identify the backbone-only reference (●), GemmaScope2 SAEs (●), transcoders (●), self-trained SAEs (●), and DUNE (●). Each metric and threshold independently samples eligible features, so plotted points need not represent the same features.

On Gemma layer 13 and Qwen layer 21, DUNE again retains 100.0% reconstruction R^2. Its monosemanticity scores vary across the two settings and metrics, suggesting that the lossless construction transfers more consistently than any particular ranking against the SAE baselines.

C.3 Qualitative results.

We illustrate three Gemma layer-16 neurons and the identified subunits for which our AutoInterp evaluation saved descriptions and activating passages (Tables 21 to 23). The original neurons have AutoInterp scores of 0.64, 0.79, and 0.93 for neurons 139, 331, and 543, respectively. This DUNE run contains 24 subunits for neuron 139, 19 for neuron 331, and 24 for neuron 543. AutoInterp sampled features rather than exhaustively describing all of them; it evaluated eight, four, and six of these subunits, respectively. We show all of those evaluated subunits.

The neurons respond to several kinds of text that are difficult to capture with one description. Neuron 139 responds to both web styling and conversational prose; neuron 331 responds to chemical, biological, and software-related passages; and neuron 543 responds to mathematical notation as well as everyday prose. Their displayed subunits show more specific recurring patterns, including web styling, music releases, cell biology, sports scores, and mathematical notation. As in the vision examples, the split makes patterns within an original neuron easier to inspect.

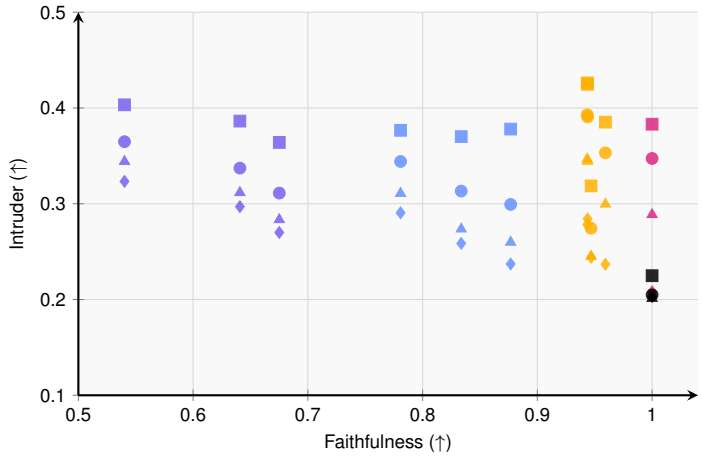


Figure 20: **Intruder versus reconstruction on Gemma 3 270M-IT, layer 16.** Colors and markers follow Figure 19.

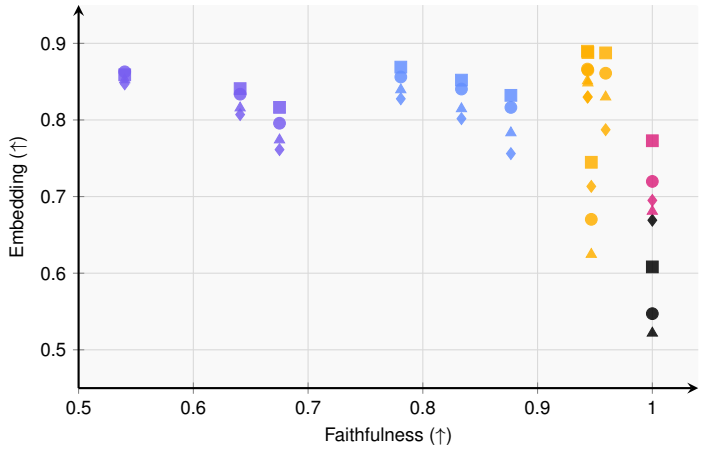


Figure 21: **Embedding versus reconstruction on Gemma 3 270M-IT, layer 16.** Colors and markers follow Figure 19.

C.4 Conclusion

To summarize, these experiments show that DUNE can be readily applied to language models without adjustments to its algorithm. The same lossless construction applies beyond vision, preserving the model's next-token predictions. In particular at higher activation threshold evaluations, DUNE achieves strong monosemanticity results.

D Implementation Details

D.1 Sparse autoencoders

Sparsity target and training budget. For each SAE and transcoder family, the target sparsity K (active features per token) is fixed at 20, following established practice in the SAE literature (Costa et al., 2025; Pach et al., 2025; Zaigrajew et al., 2025); Archetypal SAE uses a sparsity ratio of 10% as in Fel et al. (2025). All baselines are trained for 50 epochs (Costa et al., 2025; Fel et al., 2025).

Per-family grid sweep. At the fixed sparsity above, we sweep the learning rate $\lambda \in \{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$ and the dead-feature regularization weight $w_{\text{dead}} \in \{10^{-5}, 10^{-4}, 10^{-3}\}$ (which scales a re-animation term $\frac{1}{|\mathcal{F}_\varnothing|\,B} \sum_s \sum_{k \in \mathcal{F}_\varnothing} \tilde{z}_{sk}$ subtracted from the loss, where $\mathcal{F}_\varnothing$ denotes batch-dead

features and $\tilde{z}_{sk}$ their pre-activations; maximizing this term encourages dormant features to activate) in all families. This dead-feature regularization is adapted from the `overcomplete` library. The Vanilla SAE additionally varies the ℓ_1-sparsity weight $w_{\ell_1} \in \{10^{-3}, 5 \cdot 10^{-3}, 2 \cdot 10^{-2}\}$, and the JumpReLU SAE varies its sparsity-penalty weight $w_{\text{sp}} \in \{10^{-4}, 10^{-3}, 10^{-2}\}$, giving $3^3 = 27$ configurations each. The remaining families – TopK-SAE, BatchTopK-SAE, Matching Pursuit, Matryoshka SAE, and Archetypal SAE – sweep only $\lambda \times w_{\text{dead}} = 9$ configurations. In total this yields 99 configurations per (backbone, expansion factor). The Transcoder uses the Vanilla architecture and sweeps the same 27-cell $\lambda \times w_{\ell_1} \times w_{\text{dead}}$ grid as the Vanilla SAE. All sweep runs use a single fixed training seed and the first of the three seeded token datasets used in Section 4.1; the configuration selected per family is then applied to all three runs.

Selection criterion. For each candidate, we compute the equally-weighted average of normalized reconstruction quality and sparsity,

$$s = \tfrac{1}{2}\left(1 - \widetilde{\mathrm{MSE}}\right) + \tfrac{1}{2}\left(1 - \widetilde{\mathrm{L}_0}\right), \tag{5}$$

where $\widetilde{\mathrm{MSE}}$ and $\widetilde{\mathrm{L}_0}$ are min–max normalized within each family, and L_0 is the mean number of active features per token, divided by the total feature count $\mathrm{ef} \cdot d$ before normalization. The configuration with the highest s is selected per family.

D.2 Linear baseline implementation

We implement the linear baselines as frozen encoder–dictionary pairs, enabling their evaluation with the same reconstruction and unit-level metrics as the SAE baselines.

We fit PCA (Pearson, 1901) using scikit-learn's `PCA` implementation with its covariance eigendecomposition solver. For evaluation, we extract the fitted components and mean into a frozen linear encoder–dictionary pair. PCA is deterministic for a fixed token dataset and therefore has no initialization seed; we refit it only for each of the three seeded token datasets.

We use the `Semi-NMF` (Ding et al., 2010) implementation from the `overcomplete` library because the target-layer activations are non-negative. Semi-NMF (Ding et al., 2010) learns non-negative codes with an unconstrained dictionary using projected-gradient updates. To encourage sparse codes, we add an ℓ_1 penalty to the code matrix. Because Semi-NMF (Ding et al., 2010) maintains a code matrix for all fitting tokens on the GPU, we use a seeded subsample of $400{,}000$ tokens to remain within GPU memory limits. We fit Semi-NMF (Ding et al., 2010) at expansion factor 8, sweep its learning rate and sparsity penalty, and select the configuration using the same equally weighted reconstruction–sparsity criterion as for the SAE baselines.

D.3 Compute hardware

Different tasks run on different NVIDIA GPUs. The three vision–language judges of Section A.1 (Qwen3-VL-30B-A3B-Instruct, Gemma-3-27B, InternVL3-14B) are served via vLLM on H200 GPUs. DUNE checkpoint generation – top-k activation collection, contribution-vector clustering, and weight restructuring – runs on H100 or A100 GPUs; a single full-pipeline DUNE run on DINOv2-B/14 layer 12 at top-$k = 400{,}000$ takes approximately 26–30 hours on a single A100. This runtime reflects a conservative, unoptimized implementation: neurons are processed sequentially on a single GPU, top-k is set to $400{,}000$ (the high end of our ablation range), and `min_cluster_size`$= 1000$ is chosen for cluster quality – reducing it substantially shortens runtime. The cost is one-time and offline; its effect on inference is minimal, as shown in Section B.3. SAE and transcoder training, together with the downstream evaluation pipeline (classification accuracy, C-Score, R^2 reconstruction), runs on L40S GPUs. The throughput benchmark runs exclusively on a single H100.

D.4 Software stack

The codebase is implemented in PyTorch (BSD 3-Clause License) with torchvision (MIT License). Model loading uses `transformers` (Apache License 2.0), `timm` (Wightman, 2019) (Apache License 2.0), and `torch.hub`. Clustering uses `hdbscan` (Campello et al., 2013) (MIT License), with `cuML` (RAPIDS) (Apache License 2.0) accelerating the same algorithm at scale (the

source of the residual numerical non-determinism noted in Section B.2); both are installed separately from the vision environment. PCA uses `scikit-learn`'s `PCA` implementation (BSD 3-Clause License). The SAE, Semi-NMF and transcoder baselines reuse the implementations in the `overcomplete` library (MIT License). Auxiliary dependencies include `scipy` (Hungarian matching; BSD 3-Clause License), `umap-learn` (one ablation axis; BSD 3-Clause License), and `sentence-transformers` (Reimers & Gurevych, 2019) (Apache License 2.0). The VLM judges are served through `vLLM` (Apache License 2.0).

The language-model evaluations use the SAEBench repository (Karvonen et al., 2025), with `TransformerLens` for model hooks and `SAE Lens` for published SAE checkpoints. The embedding-based evaluation uses `sentence-transformers`. We implement the Simulation scoring procedure locally rather than importing OpenAI's automated-interpretability library.

D.5 Datasets

All methods use ImageNet-1k (Russakovsky et al., 2015) (standard train/validation split) with each backbone's native preprocessing pipeline; ImageNet-1k is used under its research-only terms.

D.6 Models and evaluation checkpoints

Tables 24 and 25 list the studied models, published baselines, and evaluation models, with their upstream identifiers, loading sources, and stated licenses.

D.7 Per-backbone disentanglement configurations

The 1-indexed layer numbers used throughout the paper map to 0-indexed indices in our codebase as layer_1idx $-$ 1 (e.g. "layer 12" on DINOv2-B/14 is layer index 11).

Matching the SAE expansion factor. For each experiment we first set the expansion factor for the SAE and transcoder baselines, then match it with DUNE by adjusting either the TopK activation budget or the HDBSCAN `min_cluster_size`: smaller `min_cluster_size` yields more, smaller clusters and a higher effective expansion factor. SAEs and transcoders use a fixed expansion factor of 8 unless specified otherwise.

Per-backbone settings. Table 26 lists the TopK and clustering deviations from the default (`min_cluster_size` $= 1000$, `cluster_selection_method=leaf`, `min_samples` at the library default, no UMAP; Section D.8) for each (backbone, layer) configuration evaluated in Section 4.1 and Table 13.

D.8 Clustering hyperparameters

DUNE uses HDBSCAN (Campello et al., 2013) with `cluster_selection_method=leaf`, `min_cluster_size` $= 1000$, `min_samples` at the library default, ARV unit-normalization enabled, and no UMAP dimensionality reduction. The ablations in Section B.7 sweep each of these one at a time around this baseline.

D.9 Seeds

For SAE and transcoder baselines, training initialization uses seed 42 for the headline run and $\{7, 1024\}$ for the cross-seed stability evaluation in Section 4.2. The three *seeded token datasets* of Section 4.1 use seeds $\{42, 123, 999\}$; these select random subsets of 2 tokens per ImageNet image and are independent of the SAE training seed.

Table 8: **Quantitative concept suppression: removal–retention utility.** Results on 2,955 COCO image–concept pairs. Columns show the selected fraction of positive attribution mass. Utility is target removal × non-target retention. No intervention uses the reconstructed representation without modifying features; all other rows use steering strength s. DUNE rows are shaded. The best and second-best utility in each column are shown in **bold** and underlined, respectively.

Intervention	Method	$m = 20\%$	$m = 40\%$	$m = 60\%$	$m = 80\%$	$m = 100\%$
No intervention	Original	1.05	1.05	1.05	1.05	1.05
	BatchTopK	2.51	2.51	2.51	2.51	2.51
	MSAE	3.07	3.07	3.07	3.07	3.07
	MpSAE	2.03	2.03	2.03	2.03	2.03
	ASAE	7.26	7.26	7.26	7.26	7.26
	DUNE *(ours)*	0.89	0.89	0.89	0.89	0.89
0	Original	6.75	8.03	10.67	12.21	14.66
	BatchTopK	5.80	6.97	7.93	8.20	8.51
	MSAE	6.23	7.53	7.96	8.26	8.61
	MpSAE	6.48	7.70	8.58	9.05	9.47
	ASAE	7.46	7.13	7.33	7.37	7.99
	DUNE *(ours)*	9.02	17.30	<u>26.88</u>	**36.25**	**38.89**
-0.5	Original	6.44	7.96	10.38	11.87	17.44
	BatchTopK	3.26	2.90	4.29	8.36	2.63
	MSAE	3.28	2.74	2.99	7.28	7.98
	MpSAE	12.74	7.36	9.07	7.19	1.25
	ASAE	11.36	17.40	20.22	4.24	8.88
	DUNE *(ours)*	8.16	11.52	18.52	11.23	5.26
$+0.5$	Original	7.67	11.15	15.33	<u>20.90</u>	<u>22.86</u>
	BatchTopK	14.14	<u>28.85</u>	26.21	3.22	0.84
	MSAE	17.02	28.31	**27.21**	5.20	0.35
	MpSAE	**31.94**	6.00	0.10	0.08	0.21
	ASAE	12.08	20.70	13.65	3.41	0.03
	DUNE *(ours)*	16.94	**29.61**	8.02	3.16	3.59
-1	Original	7.17	10.21	14.50	19.50	16.07
	BatchTopK	2.50	3.48	5.00	5.58	1.54
	MSAE	5.35	5.35	5.14	6.15	7.46
	MpSAE	12.58	7.94	8.00	6.25	1.49
	ASAE	17.55	19.13	6.18	4.48	5.34
	DUNE *(ours)*	12.50	11.13	6.92	5.00	4.42
$+1$	Original	9.32	18.06	20.75	13.59	5.59
	BatchTopK	<u>30.40</u>	24.95	5.71	2.36	0.73
	MSAE	28.58	27.25	9.76	1.78	0.33
	MpSAE	28.39	1.51	0.00	0.00	0.27
	ASAE	19.60	15.57	4.80	0.72	0.00
	DUNE *(ours)*	26.09	5.65	2.86	3.21	2.82
-2.5	Original	17.47	10.54	7.37	6.42	6.26
	BatchTopK	4.56	2.85	3.96	4.91	1.42
	MSAE	6.96	3.07	3.64	4.85	7.15
	MpSAE	3.95	8.64	7.12	5.24	1.53
	ASAE	5.63	4.46	5.85	6.72	2.60
	DUNE *(ours)*	6.54	5.21	5.06	4.81	2.66
$+2.5$	Original	17.78	5.38	2.96	2.53	2.58
	BatchTopK	15.48	4.68	1.97	3.03	0.62
	MSAE	15.60	3.98	2.04	0.98	0.27
	MpSAE	4.91	0.09	0.00	0.05	0.06
	ASAE	6.56	3.49	0.81	0.12	0.00
	DUNE *(ours)*	3.88	2.78	2.36	1.92	1.57
-5	Original	8.63	6.07	5.59	4.63	4.78
	BatchTopK	2.29	2.74	4.44	4.47	1.20
	MSAE	2.58	3.10	3.87	4.68	6.89
	MpSAE	5.38	8.57	7.03	4.99	1.59
	ASAE	1.19	3.10	5.18	5.75	1.79
	DUNE *(ours)*	5.23	4.65	4.50	3.78	0.00
$+5$	Original	4.18	2.86	2.35	2.60	2.17
	BatchTopK	2.98	2.38	1.27	2.32	0.41
	MSAE	4.03	1.67	0.97	0.77	0.33
	MpSAE	1.49	0.00	0.00	0.05	0.21
	ASAE	2.86	1.06	0.06	0.06	0.00
	DUNE *(ours)*	3.11	2.51	2.62	1.86	0.00

Table 9: **Quantitative concept suppression: non-target retention.** Results on 2,955 COCO image–concept pairs. Values are the percentage of non-target COCO concepts from the original caption that remain after intervention. Columns show the selected fraction of positive attribution mass. No intervention uses the reconstructed representation without modifying features; all other rows use steering strength s. DUNE rows are shaded.

Intervention	Method	$m = 20\%$	$m = 40\%$	$m = 60\%$	$m = 80\%$	$m = 100\%$
No intervention	Original	97.17	97.17	97.17	97.17	97.17
	BatchTopK	92.58	92.58	92.58	92.58	92.58
	MSAE	91.69	91.69	91.69	91.69	91.69
	MpSAE	95.11	95.11	95.11	95.11	95.11
	ASAE	79.75	79.75	79.75	79.75	79.75
	DUNE *(ours)*	97.10	97.10	97.10	97.10	97.10
0	Original	84.57	83.00	81.03	78.95	77.65
	BatchTopK	87.07	83.74	81.09	80.74	79.37
	MSAE	84.78	81.48	80.32	79.25	79.52
	MpSAE	85.47	83.30	82.30	81.57	79.47
	ASAE	79.61	79.48	78.45	78.94	78.14
	DUNE *(ours)*	83.59	80.90	75.09	66.99	56.25
-0.5	Original	82.69	79.21	78.28	75.26	70.02
	BatchTopK	81.70	77.33	63.43	27.10	2.67
	MSAE	81.33	77.95	67.93	36.64	8.26
	MpSAE	59.49	8.14	9.27	7.25	1.26
	ASAE	72.79	60.35	32.38	4.41	9.14
	DUNE *(ours)*	79.34	69.22	45.11	13.91	5.50
$+0.5$	Original	81.22	78.28	73.43	64.33	41.27
	BatchTopK	78.39	65.33	31.24	3.28	0.85
	MSAE	72.59	59.12	34.63	5.33	0.35
	MpSAE	60.43	6.13	0.10	0.08	0.21
	ASAE	68.37	45.95	16.78	3.51	0.03
	DUNE *(ours)*	75.72	40.33	8.40	3.21	3.68
-1	Original	77.60	72.91	60.42	39.59	21.51
	BatchTopK	68.27	49.98	20.28	8.40	1.56
	MSAE	65.05	44.63	22.32	9.76	7.76
	MpSAE	25.82	8.35	8.15	6.31	1.50
	ASAE	59.21	31.34	6.69	4.55	5.45
	DUNE *(ours)*	59.21	14.14	7.33	5.20	4.57
$+1$	Original	77.54	66.07	41.28	17.97	5.88
	BatchTopK	64.03	29.00	5.84	2.40	0.74
	MSAE	59.94	34.44	10.19	1.81	0.33
	MpSAE	31.58	1.52	0.00	0.00	0.27
	ASAE	47.72	21.30	5.03	0.73	0.00
	DUNE *(ours)*	43.78	5.87	2.91	3.26	2.87
-2.5	Original	45.57	11.80	7.67	6.59	6.44
	BatchTopK	12.27	4.78	5.18	5.97	1.44
	MSAE	21.80	5.67	5.13	6.06	7.41
	MpSAE	5.11	8.91	7.23	5.27	1.54
	ASAE	6.27	4.69	5.97	6.84	2.63
	DUNE *(ours)*	6.94	5.44	5.29	5.01	2.73
$+2.5$	Original	32.86	5.95	3.06	2.60	2.64
	BatchTopK	17.17	4.76	2.00	3.09	0.63
	MSAE	18.14	4.05	2.05	0.99	0.27
	MpSAE	4.95	0.09	0.00	0.05	0.06
	ASAE	7.14	3.63	0.82	0.12	0.00
	DUNE *(ours)*	3.98	2.83	2.39	1.95	1.59
-5	Original	9.12	6.26	5.76	4.75	4.91
	BatchTopK	3.88	3.81	5.39	5.38	1.22
	MSAE	3.83	4.56	5.01	5.71	7.13
	MpSAE	6.29	8.82	7.14	5.03	1.60
	ASAE	1.23	3.15	5.24	5.83	1.80
	DUNE *(ours)*	5.40	4.84	4.73	3.95	0.00
$+5$	Original	4.33	2.91	2.40	2.66	2.21
	BatchTopK	3.03	2.41	1.28	2.36	0.41
	MSAE	4.13	1.69	0.97	0.78	0.33
	MpSAE	1.49	0.00	0.00	0.05	0.21
	ASAE	2.93	1.08	0.06	0.06	0.00
	DUNE *(ours)*	3.17	2.55	2.66	1.89	0.00

Table 10: **Quantitative concept suppression: target removal.** Results on 2,955 COCO image–concept pairs. Values are the percentage of edited captions for which the VLM judge finds the target concept or a close synonym absent. Columns show the selected fraction of positive attribution mass. No intervention uses the reconstructed representation without modifying features; all other rows use steering strength s. DUNE rows are shaded.

Intervention	Method	$m = 20\%$	$m = 40\%$	$m = 60\%$	$m = 80\%$	$m = 100\%$
No intervention	Original	1.08	1.08	1.08	1.08	1.08
	BatchTopK	2.71	2.71	2.71	2.71	2.71
	MSAE	3.35	3.35	3.35	3.35	3.35
	MpSAE	2.13	2.13	2.13	2.13	2.13
	ASAE	9.10	9.10	9.10	9.10	9.10
	DUNE *(ours)*	0.91	0.91	0.91	0.91	0.91
0	Original	7.99	9.68	13.16	15.47	18.88
	BatchTopK	6.67	8.32	9.78	10.15	10.73
	MSAE	7.34	9.24	9.92	10.42	10.83
	MpSAE	7.58	9.24	10.42	11.10	11.91
	ASAE	9.37	8.97	9.34	9.34	10.22
	DUNE *(ours)*	10.80	21.39	35.80	54.11	69.14
-0.5	Original	7.78	10.05	13.27	15.77	24.91
	BatchTopK	3.99	3.76	6.77	30.83	98.48
	MSAE	4.03	3.52	4.40	19.86	96.68
	MpSAE	21.42	90.52	97.87	99.15	99.09
	ASAE	15.60	28.83	62.44	96.21	97.12
	DUNE *(ours)*	10.29	16.65	41.05	80.78	95.63
$+0.5$	Original	9.44	14.25	20.88	32.49	55.40
	BatchTopK	18.04	44.16	83.93	98.31	99.36
	MSAE	23.45	47.88	78.58	97.73	99.29
	MpSAE	52.86	97.97	99.90	99.90	98.65
	ASAE	17.66	45.04	81.35	97.33	99.56
	DUNE *(ours)*	22.37	73.43	95.47	98.27	97.73
-1	Original	9.24	14.01	23.99	49.24	74.72
	BatchTopK	3.65	6.97	24.64	66.43	98.61
	MSAE	8.22	11.98	23.05	63.01	96.21
	MpSAE	48.73	95.03	98.17	99.05	98.92
	ASAE	29.64	61.05	92.32	98.38	98.04
	DUNE *(ours)*	21.12	78.75	94.35	96.11	96.72
$+1$	Original	12.01	27.34	50.25	75.63	95.03
	BatchTopK	47.48	86.06	97.77	98.38	99.19
	MSAE	47.68	79.12	95.77	98.58	99.29
	MpSAE	89.88	99.22	99.76	99.83	98.88
	ASAE	41.08	73.10	95.43	98.68	99.83
	DUNE *(ours)*	59.59	96.28	98.31	98.44	98.31
-2.5	Original	38.34	89.34	96.04	97.33	97.12
	BatchTopK	37.16	59.63	76.55	82.23	98.65
	MSAE	31.91	54.25	71.00	79.97	96.55
	MpSAE	77.36	97.02	98.48	99.46	98.95
	ASAE	89.85	94.96	97.94	98.34	98.92
	DUNE *(ours)*	94.31	95.60	95.50	96.04	97.26
$+2.5$	Original	54.11	90.36	96.58	97.39	97.87
	BatchTopK	90.19	98.44	98.58	98.14	99.19
	MSAE	85.96	98.38	99.42	98.95	99.42
	MpSAE	99.22	99.53	99.63	99.56	98.51
	ASAE	91.95	96.07	98.88	99.49	99.80
	DUNE *(ours)*	97.63	98.38	98.71	98.21	99.15
-5	Original	94.65	96.95	97.06	97.53	97.39
	BatchTopK	59.02	71.95	82.40	83.01	98.68
	MSAE	67.34	68.12	77.33	82.03	96.68
	MpSAE	85.41	97.19	98.48	99.36	98.98
	ASAE	97.06	98.48	98.82	98.75	98.95
	DUNE *(ours)*	96.92	96.04	95.16	95.70	100.00
$+5$	Original	96.68	98.21	97.80	97.73	98.34
	BatchTopK	98.48	98.95	99.05	98.41	99.19
	MSAE	97.56	99.05	99.39	98.95	99.59
	MpSAE	99.70	99.32	99.63	99.66	98.27
	ASAE	97.43	98.27	99.29	99.56	99.83
	DUNE *(ours)*	98.17	98.41	98.54	98.31	100.00

Table 11: **Monosemanticity *vs.* faithfulness.** We compare multiple PCA expansion factors across two monosemanticity metrics—SIM@1 (higher is better) and C-Score (higher is better)—and two faithfulness metrics—downstream classification accuracy and R^2 reconstruction (for both higher is better)—on DINOv2-B/14. We report mean and standard deviation across three seeded token probing datasets. The best method is shown in **bold** and the second best is underlined.

	Monosemanticity		Faithfulness	
Method	**C-Score** (↑)	**SIM@1** (↑)	**Accuracy** (↑)	$\mathbf{R^2}$ (↑)
Backbone only	36.2	10.49 ± 0.06	84.3	100.00
PCA *(EF=0.25)*	**32.95** ± 0.60	14.49 ± 0.33	81.83 ± 0.04	84.17 ± 0.00
PCA *(EF=0.5)*	27.50 ± 0.36	11.34 ± 0.02	83.36 ± 0.04	92.88 ± 0.00
PCA *(EF=1)*	23.38 ± 0.21	9.10 ± 0.12	84.29 ± 0.00	**100.00** ± 0.00

Table 12: **Complete comparison on supervised ImageNet-21k ViT-B/16.** We compare DUNE (shaded) with PCA, Semi-NMF, SAE, and transcoder baselines. Metrics and formatting conventions match Table 2; the best method per metric is shown in **bold** and the second best in underlined.

		Interpretability			Faithfulness	
Model	**Method**	**LLM Rank** (↓)	**C-Score** (↑)	**SIM@1** (↑)	**Accuracy** (↑)	$\mathbf{R^2}$ (↑)
ImageNet-21k ViT-B/16	*Backbone only*	10	36.42	11.31 ± 0.06	85.1	100.00
	Linear baselines					
	PCA†	9	35.61 ± 0.44	12.23 ± 0.07	85.00 ± 0.01	83.86 ± 0.00
	Semi-NMF	8	41.91 ± 0.03	14.59 ± 0.06	84.90 ± 0.01	66.93 ± 0.05
	Nonlinear baselines					
	Vanilla SAE	7	54.55 ± 0.06	34.59 ± 0.19	85.01 ± 0.01	91.65 ± 0.02
	TopK-SAE	4	54.50 ± 0.17	34.08 ± 0.49	84.94 ± 0.01	87.00 ± 0.01
	JumpReLU	12	31.82 ± 0.08	6.05 ± 0.16	85.10 ± 0.01	99.99 ± 0.00
	BatchTopK-SAE	6	53.56 ± 0.55	34.40 ± 0.26	84.98 ± 0.04	87.76 ± 0.08
	Matryoshka SAE	5	51.24 ± 0.21	30.29 ± 0.30	84.97 ± 0.04	86.85 ± 0.01
	Archetypal SAE	2	59.18 ± 4.03	**45.11** ± 7.89	84.88 ± 0.03	79.76 ± 0.57
	Matching Pursuit	3	56.91 ± 0.12	32.81 ± 0.29	85.03 ± 0.05	89.66 ± 0.07
	Transcoder	11	46.31 ± 0.03	23.00 ± 0.12	85.05 ± 0.03	93.54 ± 0.01
	DUNE *(ours)*	**1**	**71.05** ± 0.06	42.90 ± 0.15	**85.10** ± 0.00	**100.00** ± 0.00

Table 13: **Generalization.** We compare DUNE (shaded) against the four strongest baselines from Table 2 – Matching Pursuit, Archetypal SAE, Matryoshka SAE, and BatchTopK-SAE – across four axes of generalization: layer, backbone size, model family, and expansion factor. Metrics and formatting conventions match Table 2; the best method per axis and metric is shown in **bold** and the second best is underlined.

		Monosemanticity			Faithfulness	
Axis	**Method**	**LLM Rank** (↓)	**C-Score** (↑)	**SIM@1** (↑)	**Accuracy** (↑)	$\mathbf{R^2}$ (↑)
Layer	*DINOv2 ViT-B/14 layer 10 (backbone only)*	6	37.02	8.95	84.3	100
	BatchTopK-SAE	2	51.58 ± 0.49	17.98 ± 0.54	83.61 ± 0.04	81.66 ± 0.03
	Matryoshka SAE	3	50.40 ± 0.27	16.40 ± 0.27	83.61 ± 0.06	81.12 ± 0.07
	Archetypal SAE	5	46.73 ± 0.06	15.37 ± 0.07	81.22 ± 0.24	69.01 ± 0.00
	Matching Pursuit	4	55.77 ± 0.07	18.09 ± 0.02	83.91 ± 0.03	83.57 ± 0.07
	DUNE *(ours)*	**1**	**70.00** ± 0.70	**28.42** ± 0.96	**84.29** ± 0.00	**100.00** ± 0.00
	DINOv2 ViT-B/14 layer 11 (backbone only)	6	30.7	9.12	84.3	100
	BatchTopK-SAE	2	43.92 ± 0.30	16.13 ± 0.31	83.54 ± 0.07	83.22 ± 0.06
	Matryoshka SAE	3	46.55 ± 0.43	17.42 ± 0.27	83.61 ± 0.06	83.84 ± 0.03
	Archetypal SAE	4	41.51 ± 0.35	15.77 ± 0.15	81.77 ± 0.05	71.21 ± 0.13
	Matching Pursuit	5	53.91 ± 0.16	19.14 ± 0.05	83.89 ± 0.08	86.46 ± 0.06
	DUNE *(ours)*	**1**	**63.53** ± 0.22	**24.85** ± 0.21	**84.29** ± 0.00	**100.00** ± 0.00
Size	*DINOv2 ViT-S/14 (backbone only)*	6	48.33	24.51	81.16	100
	BatchTopK-SAE	3	68.08 ± 0.20	30.21 ± 0.37	78.69 ± 0.14	87.24 ± 0.04
	Matryoshka SAE	2	70.22 ± 0.06	32.76 ± 0.18	78.80 ± 0.19	87.64 ± 0.03
	Archetypal SAE	5	58.62 ± 0.47	23.13 ± 1.32	77.38 ± 0.85	81.43 ± 0.07
	Matching Pursuit	4	70.65 ± 0.02	26.32 ± 0.54	79.34 ± 0.24	90.28 ± 0.10
	DUNE *(ours)*	**1**	**73.31** ± 0.06	**39.70** ± 0.35	**81.15** ± 0.00	**100.00** ± 0.00
	DINOv2 ViT-L/14 (backbone only)	6	30.06	10.10	86.12	100
	BatchTopK-SAE	4	56.97 ± 0.24	31.90 ± 0.19	84.48 ± 0.08	81.28 ± 0.15
	Matryoshka SAE	**1**	59.03 ± 0.06	33.47 ± 0.07	84.48 ± 0.09	81.15 ± 0.07
	Archetypal SAE	2	60.02 ± 1.32	**37.23** ± 1.08	83.81 ± 0.10	74.35 ± 0.38
	Matching Pursuit	5	61.41 ± 0.18	32.15 ± 0.24	84.95 ± 0.06	83.10 ± 0.11
	DUNE *(ours)*	3	**65.86** ± 0.79	30.24 ± 1.09	**86.12** ± 0.00	**100.00** ± 0.00
Family	*SigLIP2-B/16 (backbone only)*	6	34.97	12.43	78.08	100.00
	BatchTopK-SAE	3	44.83 ± 0.31	14.64 ± 0.17	77.69 ± 0.01	93.69 ± 0.12
	Matryoshka SAE	4	44.22 ± 0.32	13.44 ± 0.34	77.73 ± 0.04	93.82 ± 0.05
	Archetypal SAE	5	36.84 ± 0.51	9.24 ± 0.40	74.48 ± 0.18	77.54 ± 0.17
	Matching Pursuit	**1**	55.70 ± 0.22	22.81 ± 0.42	77.94 ± 0.05	95.62 ± 0.02
	DUNE *(ours)*	2	**67.98** ± 3.20	**24.04** ± 1.07	**78.08** ± 0.00	**100.00** ± 0.00
	ConvNeXtV2-Base (backbone only)	6	9.94	18.2	86.74	100
	BatchTopK-SAE	**1**	25.65 ± 0.12	29.81 ± 0.19	86.70 ± 0.01	84.77 ± 0.06
	Matryoshka SAE	3	23.44 ± 0.07	27.46 ± 0.08	86.67 ± 0.01	84.42 ± 0.01
	Archetypal SAE	5	17.35 ± 0.32	19.13 ± 0.59	86.66 ± 0.02	73.85 ± 0.06
	Matching Pursuit	2	30.43 ± 0.08	**32.97** ± 0.08	86.69 ± 0.01	84.03 ± 0.01
	DUNE *(ours)*	4	**32.66** ± 0.09	18.53 ± 0.62	**86.74** ± 0.00	**100.00** ± 0.00
Expansion Factor	*DINOv2 ViT-B/14 layer 12 (backbone only) (EF4)*	6	36.2	10.49	84.3	100.00
	BatchTopK-SAE	5	53.67 ± 0.43	32.42 ± 0.57	81.93 ± 0.02	87.96 ± 0.07
	Matryoshka SAE	**1**	56.36 ± 0.60	33.65 ± 0.68	81.97 ± 0.03	87.58 ± 0.04
	Archetypal SAE	3	61.98 ± 1.47	**40.40** ± 1.20	81.03 ± 0.08	83.65 ± 0.10
	Matching Pursuit	2	58.40 ± 0.12	32.35 ± 0.13	82.51 ± 0.05	88.77 ± 0.01
	DUNE *(ours)*	4	**62.72** ± 1.58	28.64 ± 0.52	**84.29** ± 0.00	**100.00** ± 0.00
	DINOv2 ViT-B/14 layer 12 (backbone only) (EF16)	6	36.2	10.49	84.3	100.00
	BatchTopK-SAE	5	44.13 ± 0.91	23.60 ± 0.90	82.16 ± 0.05	89.65 ± 0.06
	Matryoshka SAE	4	47.48 ± 0.03	25.86 ± 0.13	82.15 ± 0.05	89.38 ± 0.01
	Archetypal SAE	2	43.03 ± 1.33	26.09 ± 1.51	81.32 ± 0.11	85.26 ± 0.07
	Matching Pursuit	3	55.35 ± 0.36	30.19 ± 0.41	83.04 ± 0.02	91.37 ± 0.02
	DUNE *(ours)*	**1**	**66.77** ± 0.95	**36.21** ± 0.65	**84.29** ± 0.00	**100.00** ± 0.00

Table 14: **Clustering and feature construction ablations at expansion factor 8.** C-Score on DINOv2-B/14, layer 12. The first block compares random splitting with dot-product representations obtained by clustering raw activations. The second block varies the clustering algorithm for attribution patterns; DUNE uses HDBSCAN.

		Monosemanticity		
Model	**Ablation axis**	**LLM Rank (↓)**	**C-Score (↑)**	**SIM@1 (↑)**
DINOv2 ViT-B/14	*Feature construction*			
	Random	4	35.71 ± 0.08	13.42 ± 0.03
	HDBSCAN + dot product	3	**80.08** ± 0.28	25.68 ± 0.13
	k-Means + dot product	5	<u>78.57</u> ± 0.13	27.30 ± 0.11
	Clustering algorithm			
	k-Means	<u>2</u>	52.67 ± 0.07	**35.56** ± 0.36
	DUNE *(HDBSCAN)*	**1**	64.85 ± 0.52	<u>34.09</u> ± 0.85

Table 15: **Ablations at expansion factor 4.** C-Score on DINOv2-B/14 as we vary one DUNE clustering hyperparameter at a time around the baseline (`cluster_selection_method` = `leaf`, `min_cluster_size` = 1000, `min_samples` at the library default, ARV unit-normalization enabled, no UMAP). Random Baseline partitions the original channels into random groups as a lower-bound control. Best per axis in **bold**.

		Monosemanticity	
Model	**Ablation axis**	**C-Score (↑)**	**SIM@1 (↑)**
DINOv2 ViT-B/14	Random Baseline	34.52 ± 0.11	12.01 ± 0.08
	EOM	60.13 ± 1.48	29.70 ± 0.32
	Min Cluster Size = 100	58.01 ± 0.40	35.62 ± 0.17
	Min Cluster Size = 250	58.09 ± 0.26	34.49 ± 1.00
	Min Cluster Size = 500	61.09 ± 3.57	30.04 ± 0.44
	Min Samples = 1	55.10 ± 0.48	35.26 ± 0.36
	Min Samples = 100	58.38 ± 2.72	31.64 ± 0.47
	Min Samples = 500	61.90 ± 2.62	28.86 ± 0.23
	UMAP	54.24 ± 0.57	**39.18** ± 0.60
	DUNE *(ours)*	**62.72** ± 1.58	28.64 ± 0.52

Table 16: **Distribution of automatically selected margins ρ.** Number of the 673 split neurons selecting each candidate value on DINOv2-B/14, layer 12, with top-k = 60,000. Margins are selected independently per neuron as defined in Equation (2).

ρ	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Neurons	11	12	30	41	34	20	26	20	27	26
ρ	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1.00
Neurons	30	48	64	48	57	31	25	16	19	88

Table 17: **Ablation of the margin ρ.** C-Score on DINOv2-B/14, layer 12, when one global ρ is fixed for all neurons. Fixed-margin results use the seed-42 token dataset and top-$k = 60{,}000$; faithfulness is exact by construction in every setting ($R^2 = 100\%$). The final row gives the automatic per-neuron result across the three seeded token datasets for reference.

ρ	**C-Score** (↑)	**SIM@1** (↑)
0.05	64.03	27.86
0.10	64.28	27.84
0.15	64.28	27.91
0.20	**64.51**	27.77
0.25	64.06	27.53
0.30	63.83	27.40
0.35	63.61	27.28
0.40	63.54	27.04
0.45	63.32	27.04
0.50	62.92	27.05
0.55	63.07	26.82
0.60	62.85	26.70
0.65	62.53	26.74
0.70	62.31	26.65
0.75	62.42	26.50
0.80	62.31	26.40
0.85	62.16	26.19
0.90	61.99	26.18
0.95	61.86	26.14
1.00	50.07	26.71
Automatic per-neuron selection	62.72 ± 1.58	28.64 ± 0.52

Table 18: **Monosemanticity *vs.* faithfulness on Gemma 3 270M-IT, layer 16.** We compare DUNE (shaded) with self-trained SAEs at expansion factor 25 and published GemmaScope2 SAEs and transcoders at width 16k (expansion factor 25.6). AI denotes AutoInterp, INT Intruder, EMB Embedding, and SIM Simulation. R^2 and next-token KL measure faithfulness. We report means and standard deviations across evaluation seeds 42, 43, and 44. KL is multiplied by 100 for easier reading.

		Monosemanticity				**Faithfulness**	
Model	**Method**	**AI** (↑)	**INT** (↑)	**EMB** (↑)	**SIM** (↑)	$\mathbf{R^2}$ (↑)	**KL** (↓)
Gemma 3 270M IT	*Backbone only*	62.1 ± 0.63	20.3 ± 0.05	66.9 ± 0.17	1.5 ± 0.18	100.0 ± 0.00	0.00 ± 0.00
	GemmaScope2						
	JumpReLU (Small)	77.3 ± 0.61	29.1 ± 0.57	82.8 ± 0.25	4.5 ± 0.16	78.1 ± 0.09	10.15 ± 0.12
	JumpReLU (Medium)	74.4 ± 0.23	25.9 ± 0.27	80.2 ± 0.27	3.6 ± 0.41	83.4 ± 0.07	10.94 ± 0.06
	JumpReLU (Large)	71.7 ± 0.58	23.7 ± 0.15	75.6 ± 0.09	3.8 ± 0.37	87.7 ± 0.05	8.25 ± 0.04
	Transcoder (Small)	80.7 ± 0.40	32.3 ± 0.29	84.7 ± 0.21	4.7 ± 0.24	54.0 ± 0.21	10.97 ± 0.15
	Transcoder (Medium)	78.0 ± 0.74	29.7 ± 0.12	80.7 ± 0.10	4.9 ± 0.20	64.1 ± 0.17	7.92 ± 0.11
	Transcoder (Large)	75.9 ± 1.07	27.0 ± 0.21	76.1 ± 0.02	4.3 ± 0.04	67.5 ± 0.14	6.04 ± 0.08
	Self-Trained SAEs at EF25						
	Matching Pursuit SAE	76.4 ± 0.44	23.7 ± 0.40	78.7 ± 0.13	6.3 ± 0.27	95.9 ± 0.04	7.25 ± 0.11
	Archetypal SAE	71.4 ± 0.60	24.4 ± 0.53	71.3 ± 0.47	−1.1 ± 0.06	94.7 ± 0.04	11.28 ± 0.04
	Matryoshka SAE	80.2 ± 0.26	28.4 ± 0.09	83.0 ± 0.18	6.4 ± 0.22	94.4 ± 0.07	9.59 ± 0.09
	BatchTopK-SAE	80.7 ± 0.27	27.8 ± 0.38	83.0 ± 0.04	7.0 ± 0.41	94.3 ± 0.07	9.62 ± 0.09
	DUNE *(ours)*	75.9 ± 0.42	20.8 ± 0.05	69.5 ± 0.42	8.3 ± 0.29	100.0 ± 0.00	0.00 ± 0.00

Table 19: **Monosemanticity *vs.* faithfulness on Gemma 3 270M-IT, layer 13.** We repeat the comparison with similar numbers of features at another layer. AI denotes AutoInterp, INT Intruder, EMB Embedding, and SIM Simulation. R^2 and next-token KL measure faithfulness. We report means and standard deviations across evaluation seeds 42, 43, and 44. KL is multiplied by 100 for easier reading.

		Monosemanticity				Faithfulness	
Model	**Method**	**AI** (↑)	**INT** (↑)	**EMB** (↑)	**SIM** (↑)	**R^2** (↑)	**KL** (↓)
Gemma 3 270M IT	*Backbone only*	63.4 ± 0.34	20.5 ± 0.17	71.7 ± 0.09	2.0 ± 0.08	100.0 ± 0.00	0.00 ± 0.00
	GemmaScope2						
	JumpReLU (Small)	80.2 ± 0.08	32.1 ± 0.53	82.4 ± 0.31	6.8 ± 0.16	80.6 ± 0.05	11.99 ± 0.10
	JumpReLU (Medium)	76.3 ± 0.21	28.7 ± 0.35	80.0 ± 0.18	5.7 ± 0.54	85.2 ± 0.04	10.72 ± 0.14
	JumpReLU (Large)	73.3 ± 0.52	26.3 ± 0.21	74.4 ± 0.06	5.4 ± 0.36	88.7 ± 0.03	8.70 ± 0.12
	Transcoder (Small)	81.9 ± 0.23	34.1 ± 0.85	83.0 ± 0.28	6.4 ± 0.35	73.6 ± 0.08	25.92 ± 0.29
	Transcoder (Medium)	78.2 ± 0.28	30.9 ± 0.21	79.9 ± 0.14	5.9 ± 0.52	77.4 ± 0.05	24.42 ± 0.27
	Transcoder (Large)	75.2 ± 0.71	27.8 ± 0.08	74.8 ± 0.21	5.8 ± 0.26	84.4 ± 0.04	19.76 ± 0.19
	Self-Trained SAEs at EF25						
	Matching Pursuit SAE	77.1 ± 0.72	24.5 ± 0.03	78.8 ± 0.08	9.4 ± 0.08	94.7 ± 0.02	9.37 ± 0.08
	Archetypal SAE	80.0 ± 0.37	31.5 ± 0.21	81.2 ± 0.40	2.9 ± 0.25	85.8 ± 0.16	25.46 ± 0.32
	Matryoshka SAE	81.1 ± 0.74	29.5 ± 0.27	80.8 ± 0.23	10.0 ± 0.52	92.6 ± 0.08	13.68 ± 0.15
	BatchTopK-SAE	81.8 ± 0.18	28.9 ± 0.14	80.5 ± 0.15	10.1 ± 0.25	92.6 ± 0.10	13.49 ± 0.15
	DUNE *(ours)*	74.8 ± 0.33	21.6 ± 0.13	74.7 ± 0.34	8.1 ± 0.43	100.0 ± 0.00	0.00 ± 0.00

Table 20: **Monosemanticity *vs.* faithfulness on Qwen3 0.6B, layer 21.** We repeat the comparison with similar numbers of features at another layer. AI denotes AutoInterp, INT Intruder, EMB Embedding, and SIM Simulation. R^2 and next-token KL measure faithfulness. We report means and standard deviations across evaluation seeds 42, 43, and 44. KL is multiplied by 100 for easier reading.

		Monosemanticity				Faithfulness	
Model	**Method**	**AI** (↑)	**INT** (↑)	**EMB** (↑)	**SIM** (↑)	**R^2** (↑)	**KL** (↓)
Qwen3 0.6B	*Backbone only*	64.5 ± 0.53	21.6 ± 0.10	69.8 ± 0.10	0.0 ± 0.08	100.0 ± 0.00	0.00 ± 0.00
	Self-Trained SAEs at EF25						
	Matching Pursuit SAE	78.3 ± 0.27	25.8 ± 0.24	58.4 ± 0.23	7.1 ± 0.51	80.4 ± 0.08	2.72 ± 0.02
	Archetypal SAE	69.8 ± 0.26	24.3 ± 0.10	58.3 ± 0.26	−0.2 ± 0.38	54.7 ± 1.09	7.71 ± 0.06
	Matryoshka SAE	73.4 ± 0.37	25.9 ± 0.49	60.1 ± 0.10	1.2 ± 0.27	72.5 ± 0.36	4.08 ± 0.06
	BatchTopK-SAE	73.2 ± 0.40	25.4 ± 0.19	59.1 ± 0.27	1.8 ± 0.14	72.9 ± 0.37	3.98 ± 0.06
	DUNE *(ours)*	74.4 ± 0.55	22.6 ± 0.16	68.5 ± 0.58	4.9 ± 0.42	100.0 ± 0.00	0.00 ± 0.00

Table 21: **Qualitative examples for Gemma layer-16 neuron 139.** The neuron description comes from the backbone-only AutoInterp evaluation; subunits descriptions come from the DUNE evaluation. We show all eight subunits with saved descriptions and activating passages, out of 24 subunits in the checkpoint. Passages are shortened for readability; . . . mark omitted context.

Neuron ID	Generated feature description	Activating passage excerpt
Neuron 139	text describing software development and troubleshooting	`display: flex; align-items: center` "Look, this is how it goes. . . You attack me one at a time" "That's just how it was. There were no secrets on Main Street."
Subunit 3742	discussions of identity and social concepts like whiteness	"whiteness becomes an object through the expression of anxiety" "Calling for whiteness to be seen. . . " "whiteness studies as a political project"

Subunit 3745	programming and scripting related terms and file paths	`#!/usr/bin/expect -f`
		`#!/usr/bin/env python`
		`#!/usr/bin/env ruby`

Subunit 3748	discussions of the Middle East and related political or social issues	"Egypt is key for a stable Middle East" "public safety initiatives in Latin America and the Middle East" "ISIL. . . seized large parts of both Middle East countries"
Subunit 3749	words related to biological processes and scientific terminology	"cytokines (e.g. granulocyte-macrophage colony-stimulating factor)" "Hermaphrodites were considered gravid if they retained an embryonic. . . " "reproductive output, fecundity and embryo volume"
Subunit 3751	the concept of sequential or consecutive items or events	"row-consecutive $p \times p$-minors" "a sequence of integers" "a mandatory consecutive term of 120 months"

Subunit 3755	code elements like HTML tags and CSS classes within web development contexts	`<div class="navbar-collapse...">`
		`<div class="collapse navbar-collapse"...>`
		`<div id="synopsis"...>`

Subunit 3756	text related to font styles sizes and colors in web design or programming	`font: 10pt "Tw Cen MT Condensed Extra Bold"`
		`font: 12pt "Niagara Engraved"`
		`font: 14pt "Onyx"`

Subunit 3760	dates and times often associated with error messages or system logs	`Aug 30 06:36 backdoored-whois.deb`
		`Mon Oct 17 09:44:39 UTC`
		`[Tue Aug 16 16:26:11 2011] [error...`

Table 22: **Qualitative examples for Gemma layer-16 neuron 331.** The neuron description comes from the backbone-only AutoInterp evaluation; subunit descriptions come from the DUNE evaluation. We show all four subunits with saved descriptions and activating passages, out of 19 subunits in the checkpoint. Passages are shortened for readability; . . . mark omitted context.

Neuron ID	Generated feature description	Activating passage excerpt
Neuron 331	text describing chemical processes and production	"methanol vapors. Said vapors are toxic. . . " "induction of the senescent physiological state. . . knocking-down lncRNA-OIS1" "Response time can be an indicator of whether SQL query case is correct"

Subunit 8573	technical text related to software development and error messages	`self::IF_TYPE_EPLRS`
		`self::IF_TYPE_EON`
		`ImportError: Could not import settings`

Subunit 8575	text related to XML encoding and UTF-8 character encoding	`<?xml version="1.0" encoding="UTF-8"?>`
		`<xs:schema xmlns:...`
		`encoding='utf-8'`

Subunit 8581	musical artists, bands, albums, and songs, particularly debut releases and covers	"Bikini Kill's self-titled Kill Rock Stars debut EP will be reissued" "debut album for BMG/RCA" "Madness later recorded the song for their cover album"
Subunit 8583	scientific text describing cell types and biological processes	"lncRNA-OIS1 in primary BJ cells" "embryonic fibroblast. . . and keratinocyte" "senescence in WI-38 cells"

Table 23: **Qualitative examples for Gemma layer- neuron 543.** The neuron description comes from the backbone-only AutoInterp evaluation; subunit descriptions come from the DUNE evaluation. We show all six subunits with saved descriptions and activating passages, out of 24 subunits in the checkpoint. Passages are shortened for readability; . . . mark omitted context.

Neuron ID	Generated feature description	Activating passage excerpt
Neuron 543	mathematical limits and infinity	"$\ln 2 = \lim_{x \to 1^-} \ln(1 + \cdots)$" "Roscoe's House of Chicken and Waffles and Dirty Bird's cornbreads. . . " "Simpson ended up on the federal no-fly list"
Subunit 14088	words related to distress or sounds of discomfort	"her horrific, ear-splitting wail" "a loud, wailing howl"
Subunit 14101	sports statistics and scores often involving hyphenated numbers	"produced eight points during the run, with two long range triples" "completed just 52.5 percent of his passes" "4-for-4 from the charity stripe"
Subunit 14104	measurements and dimensions expressed as numbers and units	"dimensions of `3.875" x 2.5`. . . " "an average width of 35 metres" "4 1/4 inches wide"

Subunit 14105	code related to public class definitions and methods	`public class Problem_16 { public long sumOfDigits(...) }`
		`public class Hey<T>`
		`public class Travels`

Subunit 14107	copyright notices and associated dates	`Copyright 2018-2020 Polyaxon`
		`Copyright 2013 Google Inc.`
		`Copyright 2015 go-termios`

Subunit 14108	mathematical notation and terminology	"$\ldots^{d_1(j)} \times \mathbb{RP}^{d_2(j)}$ of the subspace" "$H^{2k-2}(N^{n-2}; \mathbb{Z}/2)$" "$\Sigma_h^{\text{reg}}$"

Table 24: **Models where we apply DUNE.** We list the upstream checkpoint identifier and loading source for each model, together with its stated license. The DINOv2 entry covers the Small, Base, and Large variants.

Model	Identifier	Source	License
DINOv2-S/B/L (Oquab et al., 2024)	`facebookresearch/dinov2` (`vits14`, `vitb14`, `vitl14`)	`torch.hub`	Apache-2.0
AugReg ViT-B/16 (Dosovitskiy et al., 2021; Steiner et al., 2022)	`vit_base_patch16_224.augreg_in21k_ft_in1k`	`timm`	Apache-2.0
ConvNeXtV2-Base (Woo et al., 2023)	`convnextv2_base.fcmae_ft_in22k_in1k`	`timm`	CC BY-NC 4.0
SigLIP2-B/16 (Tschannen et al., 2025)	`google/siglip2-base-patch16-224`	Hugging Face	Apache-2.0
LLaVA-OneVision 0.5B (Li et al., 2025)	`llava-hf/llava-onevision-qwen2-0.5b-ov-hf`	Hugging Face	Apache-2.0
Gemma 3 270M-IT (Team, 2025a)	`google/gemma-3-270m-it`	Hugging Face	Gemma terms
Qwen3 0.6B (Team, 2025b)	`Qwen/Qwen3-0.6B`	Hugging Face	Apache-2.0

Table 25: **Published baselines and models used to evaluate.** We list checkpoint identifiers, loading sources, and stated licenses. For CLIP, the license follows the upstream project repository; for InternVL3-14B, it follows the model card's License section.

Model	Identifier	Source	License
Gemma Scope 2 270M-IT (McDougall et al., 2025)	`google/gemma-scope-2-270m-it`	Hugging Face	CC BY 4.0
CLIP ViT-B/16 (Radford et al., 2021)	`openai/clip-vit-base-patch16`	Hugging Face	MIT
SigLIP2-SO400M/14 (384 px) (Tschannen et al., 2025)	`google/siglip2-so400m-patch14-384`	Hugging Face	Apache-2.0
Qwen3-VL-30B-A3B-Instruct (Team, 2025c)	`Qwen/Qwen3-VL-30B-A3B-Instruct`	Hugging Face	Apache-2.0
Gemma 3 27B-IT (Team, 2025a)	`google/gemma-3-27b-it`	Hugging Face	Gemma terms
InternVL3-14B (Zhu et al., 2025)	`OpenGVLab/InternVL3-14B`	Hugging Face	MIT
Gemma 3 12B-IT (Team, 2025a)	`google/gemma-3-12b-it`	Hugging Face	Gemma terms
Llama 3.1 8B (Team, 2024)	`meta-llama/Llama-3.1-8B`	Hugging Face	Llama 3.1 license
all-MiniLM-L6-v2 (Reimers & Gurevych, 2019)	`sentence-transformers/all-MiniLM-L6-v2`	Hugging Face	Apache-2.0

Table 26: **Per-backbone DUNE configurations.** Default clustering means `min_cluster_size` $= 1000$ with all other settings as in Section D.8. Effective EF is the per-neuron cluster count summed across neurons divided by the layer width, averaged over the three seeded token datasets.

Backbone	Layer	TopK	Clustering	Eff. EF
AugReg ViT-B/16	12	100k	default	7.49
DINOv2-B/14	12	400k	default	8
DINOv2-B/14	10	400k	default	8.01
DINOv2-B/14	11	200k	default	5.91
DINOv2-S/14	12	400k	default	8.1
DINOv2-L/14	23	400k	`min_cluster_size` $= 500$	7.90
SigLIP2-B/16	12	200k	default	7.51
ConvNeXtV2-B	last linear layer of the fourth stage	500k	default	7.58
DINOv2-B/14 (EF4)	12	60k	default	3.96
DINOv2-B/14 (EF16)	12	400k	`min_cluster_size` $= 750$	11.41
Gemma 3 270M-IT	13	100k	`min_cluster_size`$= 250$, `eom`	26.56
Gemma 3 270M-IT	16	100k	`min_cluster_size`$= 250$, `eom`	25.66
Qwen3 0.6B	21	100k	`min_cluster_size`$= 250$, `eom`	28.13