BACK TO THE BASICS: INTERPRETING VISION EMBEDDINGS VIA BINARY SPARSE AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse autoencoders (SAEs) have emerged as the primary framework for interpreting the internal representations of vision (and language) models, decomposing dense, entangled embeddings into sparse, more human-interpretable latent codes. Continuous activations trained under sparsity constraints have come to dominate this line of work, due to their manageable optimization dynamics. In contrast, *binary* latent codes remain scarce within the SAE setting, despite offering a natural and arguably unambiguous notion of concept presence. Binary SAEs trained with Gumbel relaxations or straight-through estimators are prone to training instability and typically underperform their continuous counterparts. In this work, we argue that binary SAEs are not intrinsically flawed: rather, it is *how* they are trained that directly and disproportionately impacts the quality of their representations. To address this, we introduce **BASICS-SAE** (**B**inary **A**tom **S**election by **I**terative **C**oordinate **S**wapping), a binary SAE that draws inspiration from classical dictionary learning by casting binary code inference as a Gauss–Seidel local search over supports, in which every accepted coordinate swap strictly decreases the reconstruction objective. We compare BASICS-SAE against Gumbel-relaxed and Local Winner-Take-All binary SAEs, and against continuous TopK and BatchTopK SAEs, on CUB and ImageNet using CLIP and DINOv3 embeddings. Beyond reconstruction fidelity, we evaluate dictionary quality using downstream few-shot classification and formal mechanistic interpretability metrics, based on recently proposed concept alignment scores. BASICS-SAE is competitive against continuous baselines in reconstruction while substantially outperforming all binary and continuous models in downstream code quality and concept alignment in most settings. Our results establish binary SAEs, when paired with sequential coordinate selection, as a strong and promising foundation for vision interpretability.

1 INTRODUCTION

Recent advances in vision representation learning have produced increasingly powerful encoders that map images into dense, high-dimensional embedding spaces, spanning distinct pretraining paradigms from language-aligned contrastive objectives, most notably Contrastive Language-Image Pre-training (CLIP) (Radford et al., 2021), to purely self-supervised visual objectives such as DINO (Oquab et al., 2023; Siméoni et al., 2025). Despite their impressive downstream performance, whether in zero-shot transfer or as general-purpose visual features, the underlying representations learned by these encoders remain highly dense, entangled, and geometrically distorted (Olah et al., 2017; Liang et al., 2022; Elhage et al., 2022). Consequently, dissecting these continuous embeddings to isolate individual, human-interpretable visual primitives, often referred to as monosemantic feature decomposition (Huben et al., 2024; Pach et al., 2025), remains a formidable challenge across vision encoders regardless of their underlying training objective.

Sparse autoencoders (SAEs) have emerged as the dominant framework for tackling this challenge, decomposing dense embeddings into a sparse, over-complete dictionary of latent features. Within this space, continuous activations trained under an explicit sparsity budget, most notably TopK (Gao et al., 2025) and BatchTopK (Bussmann et al., 2024) SAEs, have become the de facto standard, owing to their comparatively stable and well-understood optimization dynamics. Yet decomposing an embedding into sparse latents is only half of the interpretability problem: knowing *which* units are active says little about *what* those units represent. In this context, a substantial and growing

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.

body of work has therefore focused on the complementary problem of assigning semantic meaning to individual dimensions, through automated concept naming (Bau et al., 2017; Panousis & Chatzis, 2023; Yuksekgonul et al., 2023; Rao et al., 2024), feature visualization (Zaigrajew et al., 2025; Lim et al., 2025), or, most relevant to our setting, alignment against externally annotated concepts and attributes (Pach et al., 2025; Klotz et al., 2026). Among these, concept alignment methods, which directly measure the correspondence between a model's latent units and human-defined concepts, have emerged as a particularly promising and increasingly standard tool for evaluating whether a learned dictionary yields representations that are genuinely interpretable, rather than merely sparse.

It is in this context that binary latent codes become an attractive, if underexplored, alternative to their continuous counterparts. A continuous activation, however sparse, still requires a downstream interpretability method to reason about *both* which direction a unit points in *and* how strongly it is expressed. This magnitude has no clear analogue in how humans typically describe concept presence. A binary code sidesteps this entirely, since a unit is either present or not, offering an unambiguous notion of concept presence that aligns naturally with how concept alignment methods pose the evaluation problem in the first place. Despite this apparent fit, binary SAEs remain scarce in the literature; the few attempts to adopt them, primarily in the context of Large Language Models (LLMs), have reported mixed results, typically evaluated against a narrow set of metrics, and have concluded that binary codes are not well suited to this setting owing to their tendency of yielding uninterpretable high-prevalence features (Quirke et al., 2025). In this work, we revisit that conclusion in the context of vision encoders. We argue that binary SAEs are not *intrinsically* flawed: rather, it is *how* they are trained, typically via relaxed, gradient-based heuristics such as Gumbel-sigmoid or straight-through estimation, that has impacted the quality of their emerging representations, and that this is a solvable problem rather than a fundamental limitation of the binary nature of the codes.

To address this, we draw inspiration from classical dictionary learning Olshausen & Field (1997), leveraging the combinatorial machinery long used in sparse coding Mallat & Zhang (1993) within an online alternating minimization procedure (Mairal et al., 2010). Building on this rationale, we propose **BASICS-SAE** (**B**inary **A**tom **S**election by **I**terative **C**oordinate **S**wapping), a binary SAE in which the active code for every input is obtained not through a relaxed, gradient-based approximation, but through a sequential, monotone coordinate-swap refinement, paired with a closed-form, alternating dictionary update. The resulting training procedure requires no temperature annealing, no straight-through gradient estimation, and no backpropagation.

We experimentally validate BASICS-SAE across two distinct vision backbones (CLIP and DINOv3) and two datasets of differing scale and class diversity (ImageNet-1k and CUB-200-2011), comparing against two binary SAE variants: (i) one based on Gumbel-sigmoid and straight-through estimation (STE), and (ii) a Stochastic Local Winner-Takes-All (LWTA) approach, as well as against state-of-the-art continuous TopK-based SAEs. We report reconstruction fidelity, but the main metrics are in- and out-of-distribution downstream few-shot classification from the learned sparse codes and concept alignment, to directly test whether the obtained binary codes are genuinely more interpretable, not merely effective for reconstruction. Our contributions are as follows:

- **Reformulating Binary Sparse Recovery:** We recast binary sparse coding in SAEs as a discrete inference problem rather than a gradient-estimation task. By exploiting the quadratic dependence of the reconstruction loss on binary supports, candidate supports are evaluated in closed form via sequential Gauss-Seidel coordinate swaps, eliminating straight-through estimators, temperature annealing, and continuous relaxations.
- **Gradient-Free Alternating Optimization:** We pair this discrete solver with a closed-form dictionary update, yielding a fully alternating, hyperparameter-light training framework where neither the binary codes nor the dictionary weights require backpropagation or gradient descent.
- **Empirical Validation:** Across two vision backbones (CLIP and DINOv3) and datasets of varying scale, BASICS-SAE matches continuous TopK SAEs in reconstruction fidelity while outperforming both continuous and binary baselines in concept alignment and in- and out-of-distribution few-shot classification. Crucially, these findings directly challenge prior conclusions that binary codes are inherently unsuited for interpretability via sparse autoencoding.

2 Related Work

Sparse Autoencoders (SAEs) have emerged as a foundational tool for mechanistic interpretability, designed to decompose the high-dimensional internal activations of neural networks into an overcomplete dictionary of directions, each intended to correspond to a distinct, human-interpretable concept. The premise, articulated in the foundational work applying this framework to language model activations (Bricken et al., 2023; Huben et al., 2024), is that a network's true underlying features, each corresponding to a direction in the feature space, are sparsely activated and often superposed within a lower-dimensional activation space, and that an overcomplete, sparsely-activating dictionary can recover them as individually interpretable, approximately monosemantic directions. Thus, an SAE maps an activation vector $x \in \mathbb{R}^D$ to an overcomplete latent representation $z = f_\psi(x) \in \mathbb{R}^K$ with $K \gg D$, trained to reconstruct x via a linear dictionary matrix $W \in \mathbb{R}^{D\times K}$ under a sparsity constraint. This framework directly adapts classical sparse dictionary learning (Olshausen & Field, 1997) to neural feature analysis. The evolution of this field traces a clear trajectory: from early iterative sparse solvers, to amortized feedforward autoencoders, and more recently toward discrete binary representations that eliminate continuous feature scaling artifacts.

Classical and Online Dictionary Learning. The fundamental objective of learning an overcomplete basis to represent signals compactly traces back to classical sparse dictionary learning (Olshausen & Field, 1997), formulated as a joint minimization over a dictionary matrix $W \in \mathbb{R}^{D\times K}$ and sparse representations $Z \in \mathbb{R}^{K\times N}$ for a dataset $X \in \mathbb{R}^{D\times N}$. Early classical algorithms solved this via and alternate optimization procedure: methods like the Method of Optimal Directions (MOD) (Engan et al., 1999) and K-SVD (Aharon et al., 2006) alternated between solving a full-batch sparse coding step for Z and updating the dictionary W via pseudo-inverses or singular value decompositions. To scale dictionary learning to massive or streaming data, Mairal et al. (2010) introduced *Online Dictionary Learning* (ODL), replacing full-batch iterations with streaming mini-batch updates that optimize dictionary columns via block-coordinate descent. In both full-batch and online regimes, the fidelity of the learned dictionary hinges on the inner sparse pursuit step used to infer Z. Traditional ways to tackle this sub-problem include cyclic coordinate descent (Fu, 1998; Friedman et al., 2007) or greedy pursuit methods like Orthogonal Matching Pursuit (OMP) (Pati et al., 1993). However, sequential fixed-order updates and pure greedy pursuit are susceptible to sub-optimal support selection when dictionary atoms are correlated. To overcome this greedy trap, Jain et al. (2011) introduced *Orthogonal Matching Pursuit with Replacement* (OMPR), augmenting pursuit with an iterative local exchange (swap) mechanism that dynamically replaces low-contribution active atoms with higher-gain candidates. Modern Sparse Autoencoders amortize this entire pipeline by replacing the iterative per-sample inner solver for Z with an explicit feedforward encoder network $f_\psi(x)$ trained via gradient descent. However, while this enables rapid single-pass inference, it completely forfeits the iterative support refinement of classical pursuit algorithms, leaving SAEs vulnerable to potentially sub-optimal one-shot latent selection. Building on these classical foundations, our work formulates binary sparse representation recovery as an explicit Gauss-Seidel coordinate descent over discrete binary supports, using parallel random initialization followed by OMPR-style coordinate swaps to escape greedy selection traps.

Continuous SAE Architectures and Trade-offs. Early SAEs enforced sparsity by pairing a ReLU nonlinearity with an L_1 penalty on activations (Huben et al., 2024; Bricken et al., 2023). While straightforward, penalizing activation magnitudes directly causes severe *shrinkage*, incentivizing gradient descent to systematically underestimate feature strengths (Gao et al., 2025), whle creating a tension between sparsity and reconstruction fidelity. Subsequent literature addresses shrinkage by either replacing L_1 with explicit cardinality constraints, such as TopK SAEs (Gao et al., 2025) and BatchTopK (Bussmann et al., 2024), or by architecturally decoupling feature selection from magnitude estimation via Gated SAEs (Rajamanoharan et al., 2024a) and JumpReLU SAEs (Rajamanoharan et al., 2024b). Other extensions restructure the dictionary topology, such as Matryoshka SAEs (Zaigrajew et al., 2025) for hierarchical feature organization, or MP-SAE (Costa et al., 2025), which unrolls Matching Pursuit into feedforward layers to combat atom correlation. Nevertheless, all continuous SAE variants retain continuous activation magnitudes, creating continuous degrees of freedom that remain susceptible to feature scaling variations and magnitude noise.

Binary SAEs. While continuous SAEs decompose neural representations into interpretable directions, continuous magnitudes create a key interpretability bottleneck: low-strength activations frequently conceal polysemantic noise, allowing the autoencoder to "offload" arbitrary residual in-

formation into exact scalar values (Quirke et al., 2025). To enforce strict semantic discretization, recent work proposed the Binary Sparse Autoencoder (BAE) (Quirke et al., 2025), constraining latent codes to $z \in \{0,1\}^K$. The forward pass of a binary TopK autoencoder is formulated as:

$$\hat{x} = W_{\text{dec}} \cdot \text{Binarise}\left(\text{TopK}(W_{\text{enc}}x + b_{\text{enc}})\right) + b_{\text{dec}} \tag{1}$$

where $\text{TopK}(\cdot)$ enforces k-sparsity and $\text{Binarise}(\cdot)$ maps positive pre-activations to 1. Because the step function derivative vanishes almost everywhere, backpropagating through equation 1 requires gradient approximations such as Straight-Through Estimators (STE) (Bengio et al., 2013) or continuous relaxations (Jang et al., 2017). To regulate feature allocation, Quirke et al. (2025) also introduced *GroupMax*, which select the largest pre-activation within each of k fixed groups of latents; this is the Local Winner-Take-All (LWTA) mechanism (Srivastava et al., 2013; Panousis et al., 2019). BASICS instead selects an unstructured support. Crucially, due to capacity loss and severe optimization artifacts introduced by STE gradient routing, prior works concluded that feedforward binary SAEs suffer substantial reconstruction degradation and struggle to discover high-quality features relative to continuous variants. This directly underscores the necessity for a direct, non-gradient combinatorial pursuit framework capable of bypassing backpropagation artifacts, thereby providing a realistic assessment of the representation capacity of binary codes.

3 Proposed Framework

We introduce **BASICS-SAE** (**B**inary **A**tom **S**election by **I**terative **C**oordinate **S**wapping), a binary sparse autoencoder in which the active code for each input is obtained by *explicit* combinatorial search rather than a relaxed, gradient-based approximation. Unlike standard TopK selection (Gao et al., 2025) or classical OMP (Pati et al., 1993), which commit to feature latents in a single forward pass, our selection mechanism adopts an iterative replacement paradigm inspired by Orthogonal Matching Pursuit with Replacement (OMPR) (Jain et al., 2011) and adapted to the binary setting. Given a fixed target sparsity budget k, our algorithm initializes an active set A and systematically evaluates the marginal reduction in reconstruction error achieved by swapping out the least contributing active latent $j \in A$ for a candidate latent $i \notin A$. This local search over the binary support space enables the model to recover from early selection errors, yielding significantly tighter reconstruction fidelity under strict binary constraints.

Let $x \in \mathbb{R}^d$ be an embedding from a frozen vision encoder, e.g., DINOv3. BASICS-SAE models:

$$x \approx b + W^\top z, \qquad z \in \{0,1\}^K, \quad \|z\|_0 = k, \tag{2}$$

where $W \in \mathbb{R}^{K \times d}$ is a learned dictionary of K atoms, $b \in \mathbb{R}^d$ a bias, and z a *binary* code with exactly k active units. Unlike conventional TopK SAEs, an active unit contributes its full atom, unweighted; thus, all of the model's freedom to express *how much* a concept is present lives in the dictionary, $\|W_j\|$, not in the code itself.

Rather than approximating this discrete choice via continuous relaxations, we formulate sparse code inference directly as a k-cardinality combinatorial optimization problem. Centering the target embedding around the decoder bias as $x_c = x - b$, the exact reconstruction objective for a candidate support $A \subseteq \{1, \ldots, K\}$ with $|A| = k$ is

$$\mathcal{L}(A) = \Big\| x_c - \sum_{j \in A} W_j \Big\|_2^2. \tag{3}$$

Expanding the square and collecting per-atom and per-atom-pair terms gives an *exact* quadratic pseudo-Boolean function (Boros & Hammer, 2002) of the support:

$$h_j := 2\langle W_j, x_c \rangle - \|W_j\|^2 \qquad \text{(marginal gain of atom } j\text{)}, \tag{4}$$

$$S_{ij} := 2\langle W_i, W_j \rangle \qquad \text{(pairwise overlap of atoms } i, j\text{)}, \tag{5}$$

$$\mathcal{L}(A) = \|x_c\|^2 - \sum_{j \in A} h_j + \sum_{\substack{i,j \in A \\ i<j}} S_{ij}. \tag{6}$$

Crucially, once the support A is chosen, there are no continuous activation magnitudes left to fit; selecting A completely determines the code $z = \mathbb{1}_A$. This collapses code inference entirely into

support selection, making combinatorial solvers tractable for binary SAEs. At the same time, this structural property naturally induces an alternating block-coordinate optimization scheme that decouples discrete code inference (z) from continuous parameter learning (W). The overall training procedure proceeds by iterating between two steps until convergence:

1. **Support Selection (Code Step):** Given the current dictionary W, we identify the binary support A for each input x by solving the quadratic pseudo-Boolean objective.
2. **Dictionary Update (Dictionary Step):** Holding the binary codes $z = \mathbb{1}_A$ fixed across the dataset, we update the dictionary W in closed form via a standard linear solve.

By repeatedly alternating between refining the discrete supports and updating the continuous dictionary vectors, BASICS-SAE monotonically optimizes the joint reconstruction loss without requiring backpropagation or gradient approximations. To operationalize this alternating scheme, we must define two concrete mechanisms: first, how to infer the discrete support A for a given centered input; and second, how to update the continuous dictionary W given fixed binary codes.

3.1 Support Selection and Pursuit Dynamics

A classical approach to constructing a k-sparse support $A \subset \{1, \dots, K\}$ is Matching Pursuit (MP) (Mallat & Zhang, 1993), or its binary counterpart BMP (Wen & Li, 2021), which greedily selects atoms step-by-step based on conditional marginal gains $s_j(A) = h_j - \sum_{i \in A} S_{ij}$. While BMP selects locally optimal units stepwise, it suffers from two key limitations: high serial latency (k sequential pursuit steps per input) and susceptibility to a "greedy lock", where sub-optimal early choices freeze permanently into the support. To overcome these bottlenecks while maintaining support diversity and computational efficiency, we decouple support selection into two distinct phases: (i) *support initialization* (A_0), followed by (ii) *refinement via iterative coordinate swaps*. Specifically, we restrict refinement to single-atom swaps ($i \in A \leftrightarrow j \notin A$). While multi-atom swaps scale combinatorially in computational complexity, single-atom moves represent a good tradeoff: they allow for efficiently evaluating candidate exchanges via simple scalar updates on h and S, keeping high throughput on parallel hardware while guaranteeing monotonic loss reduction.

Support Initialization. This two-phase framework raises the question of how to best initialize the candidate set A_0. Using BMP as an initializer retains its serial latency and risks freezing early sub-optimal picks that require extensive downstream swapping to undo. A cheaper alternative is to select A_0 in a single parallel step via TopK on the scores h_j. However, this approach tends to select clusters of dominant *mutually redundant* atoms, which can be disproportionately hard to revert via single-atom swaps. Random initialization carries no such structural bias: since it samples the support independently of h, redundancy in A_0 occurs only at the dictionary's base rate, rather than being actively concentrated by the scoring rule. We therefore initialize A_0 by sampling k units uniformly at random from the dictionary, and delegate all support quality entirely to the post-initialization phase: we pass this candidate support directly to an iterative Gauss–Seidel coordinate swap solver, which adapts the replacement principles of Orthogonal Matching Pursuit with Replacement (OMPR) (Jain et al., 2011) to efficiently resolve feature redundancies in the binary support space.

Iterative Refinement. Given an initial support A, we refine it by repeatedly identifying the weakest active atom $i \in A$ and the best available replacement $j \notin A$, executing a swap $A' = (A \setminus \{i\}) \cup \{j\}$ whenever doing so strictly decreases $\mathcal{L}(A)$. To evaluate whether a candidate swap $i \to j$ is beneficial, we compute the net change in reconstruction loss. Let $i \in A$ be a removal candidate and $j \notin A$ a replacement candidate, $A' = (A \setminus \{i\}) \cup \{j\}$. The value of i within A is

$$v_i(A) = h_i - \sum_{\substack{l \in A \\ l \neq i}} S_{il}. \tag{7}$$

In this context, the gain of replacement candidate j must be evaluated against $A \setminus \{i\}$, i.e., the actual set j is being added to, rather than the full set A:

$$g_j(A \setminus \{i\}) = \underbrace{h_j - \sum_{l \in A} S_{lj}}_{=: g_j(A)} + S_{ij}. \tag{8}$$

The true net gain of the swap is therefore

$$\Delta_{ij} = g_j(A \setminus \{i\}) - v_i(A) = \big(g_j(A) + S_{ij}\big) - v_i(A), \tag{9}$$

and a swap is accepted iff $\Delta_{ij} > 0$. Crucially, omitting the pairwise overlap S_{ij} (i.e., testing $g_j(A) > v_i(A)$) is mathematically unsound: it is equivalent to testing $\Delta_{ij} > S_{ij}$, which erroneously accepts loss-increasing swaps when $S_{ij} < 0$ and rejects valid loss-decreasing swaps when $S_{ij} > 0$. Because accepting a swap with $\Delta_{ij} > 0$ strictly guarantees a decrease in loss ($\mathcal{L}(A') < \mathcal{L}(A)$), the sequence of objective values strictly monotonically decreases. Since A ranges over a finite set of candidate supports, this greedy search must terminate after finitely many accepted swaps. In practice, however, we cap the local search at a budget of maxSwaps refinements per representation (see Algo. 2). Unlike BMP, this dynamic is non-myopic: atoms selected early can be displaced later as the support evolves, and unlike soft continuous approximations (e.g., Gumbel-Softmax or LWTA), every evaluation operates directly on the exact combinatorial objective without temperature schedules or forward-backward mismatches.

3.2 Dictionary Update

With codes $z_n \in \{0,1\}^K$ fixed for a batch, minimizing $\sum_n \|x_n - b - W^\top z_n\|^2$ over W is ordinary least squares. Following standard online dictionary learning formulations (Mairal et al., 2010), and writing Z (rows $z_n^\top$) and X_c (rows $x_n - b$), we maintain exponentially-decayed sufficient statistics across batches using a decay rate γ, s.t.,

$$A^{(t)} = \gamma A^{(t-1)} + Z_t^\top Z_t, \qquad B^{(t)} = \gamma B^{(t-1)} + Z_t^\top X_{c,t}, \tag{10}$$

and solve $W^{(t)} = \big(A^{(t)} + \lambda I\big)^{-1} B^{(t)}$ at every step, with ridge λ guarding against atoms with little accumulated activity. The bias is updated by an exponential moving average of the residual left after decoding. Because the code step and the dictionary step alternate against a shared exact objective, the overall approach in Algorithm 1 is an instance of block-coordinate descent: each step is individually non-increasing in the objective it optimizes.

Algorithm 1 BASICS-SAE: training/inference step

$x_c \leftarrow x - b$; $A \leftarrow \textsc{InitCodes}(x_c, W)$
for alt $= 1$ **to** n_{alt} **do**
  $A \leftarrow \textsc{IterativeSwap}(x_c, A)$
  if training **then** update W via ridge on $\mathbb{1}_A$
  end if
end for
$A \leftarrow \textsc{IterativeSwap}(x_c, A)$; $z \leftarrow \mathbb{1}_A$
if training **then** update b via residual EMA
end if
return z, $x_{\text{reconstruct}} = W^\top z + b$

Algorithm 2 $\textsc{IterativeSwap}(x_c, A_{\text{init}})$

for $t = 1$ **to** maxSwaps **do**
  $i \leftarrow \arg\min_{i \in A} v_i(A)$
  $j \leftarrow \arg\max_{j \notin A} \big(g_j(A) + S_{ij}\big)$
  if $\Delta_{ij} \le 0$ **then break** ▷ converged
  end if
  swap $i \to j$ in A
end for
return A

At inference, only Algorithm 2 is executed, once, against the frozen dictionary; the training-only branches in Algorithm 1 are skipped and no alternation occurs. The dictionary and bias are never updated on evaluation data.

4 Experimental Evaluation

4.1 Experimental Setup

Reconstruction Fidelity and Downstream Semantic Utility. To evaluate the representation capacity, robustness, and interpretability of our binary sparse codes, we benchmark our framework using CUB (Wah et al., 2011) and ImageNet-1k (Russakovsky et al., 2015). Because reconstruction fidelity serves merely as a proxy, comprehensively evaluating learned sparse features requires assessing downstream utility and semantic code quality. We thus evaluate discrete code quality via few-shot classification and, to test whether the learned sparse representations generalize beyond in-distribution splits, we also evaluate few-shot classification under distribution shift, using ImageNet-A (Hendrycks et al., 2021b) and ImageNet-R (Hendrycks et al., 2021a).

Table 1: **NMSE and Few-Shot Classification on CUB-200-2011 and ImageNet-1k.** Comparison across CLIP (ViT-L/14) and DINOv3 (ViT-L/16) embeddings with $k = 32$ and dictionary sizes 3072 and 4096, respectively. **BASICS-SAE** consistently outperforms binary baselines and performs competitively with continuous models across feature geometries. Best results in **bold**.

Method	CLIP Embeddings				DINOv3 Embeddings			
	NMSE ↓	1-Shot↑	10-Shot↑	50-Shot↑	NMSE ↓	1-Shot↑	10-Shot↑	50-Shot↑
CUB-200-2011								
Raw feature vector	-	44.6%	70.2%	75.9%	-	73.3%	83.5%	85.7%
TopK (Gao et al., 2025)	0.073	47.7%	66.9%	71.9%	0.213	74.4%	84.8%	87.1%
BatchTopK (Bussmann et al., 2024)	0.069	45.8%	66.9%	72.5%	0.192	**75.0%**	85.0%	87.4%
Gumbel (Quirke et al., 2025)	0.196	45.9%	63.4%	66.7%	0.337	74.2%	81.9%	83.6%
LWTA (Quirke et al., 2025)	0.133	44.9%	66.6%	69.4%	0.309	74.1%	84.7%	86.4%
BASICS	**0.057**	**48.5%**	**74.9%**	**78.9%**	**0.172**	72.4%	**85.5%**	**87.5%**
ImageNet-1k								
Baseline	-	35.8%	57.0%	62.7%	-	58.7%	77.5%	**79.4%**
TopK (Gao et al., 2025)	0.097	**45.8%**	66.3%	69.3%	0.239	60.3%	**77.8%**	79.1%
BatchTopK (Bussmann et al., 2024)	**0.095**	45.5%	65.5%	68.5%	**0.236**	60.1%	**77.8%**	79.2%
Gumbel (Quirke et al., 2025)	0.176	37.6%	57.0%	60.8%	0.423	60.4%	75.6%	76.9%
LWTA (Quirke et al., 2025)	0.177	38.2%	61.2%	65.2%	0.452	55.1%	75.1%	76.7%
BASICS	0.100	41.8%	**67.3%**	**71.0%**	0.297	**60.5%**	**77.8%**	**79.4%**

Vision Backbones and Feature Extraction. We extract embeddings from two Vision Transformer backbones spanning distinct pretraining paradigms: a multimodal, language-aligned CLIP `ViT-L/14` (Radford et al., 2021) and a self-supervised DINOv3 `ViT-L/16` (Siméoni et al., 2025). While CLIP provides language-grounded semantic features, DINOv3 captures fine-grained spatial and geometric representations without textual supervision; DINOv2 (Oquab et al., 2023) results are included in Appendix A.3). Evaluating across both spaces verifies whether our learned binary latents generalize across multimodal and strictly visual domains.

Concept Alignment Score (*TAPAScore*). Beyond downstream utility, a primary goal of SAEs is to extract visual features that faithfully align with human-understandable concepts. To evaluate this, we benchmark our learned representations on CUB-200-2011 (Wah et al., 2011) using the evaluation framework proposed by Klotz et al. (2026). Their proposed *TAPAScore* evaluates interventional fidelity by testing whether targeted feature perturbations induce selective and causally consistent changes in downstream latent activations. More precisely, it first performs a many-to-one latent-concept matching (cf. Appendix A.2), where a ground-truth concept is matched to a coalition of a few latents combined (here we use 1, 3, 5 and 10 latents), then verifies if concepts added (resp. removed) in a controlled way cause matched latents to fire (resp. unfire) accordingly.

Baselines. We benchmark against a representative suite of continuous and binary sparse autoencoder (SAE) architectures: (i) standard *TopK* (Gao et al., 2025), enforcing strict sample-level k-sparsity; (ii) *BatchTopK* (Bussmann et al., 2024), relaxing this to a batch-level budget; (iii) *TopK Gumbel-Sigmoid* (Gumbel) binary SAE with a straight-through estimator (STE); and (iv) Stochastic Local Winner-Takes-All (*LWTA*) binary SAE (Srivastava et al., 2013; Panousis et al., 2019), which partitions neurons into competitive groups matching the GroupMax setup of (Quirke et al., 2025). All models share identical target sparsity budgets. Full training details appear in Appendix A.

4.2 Results and Discussion

Reconstruction Fidelity and Downstream Semantic Utility. We evaluate representation quality through a dual lens: signal reconstruction fidelity (NMSE) and downstream semantic utility via an N-shot prototype-based classification rationale, on contrastive (CLIP) and self-supervised (DINOv3) features. The obtained comparative results are depicted in Table 1. Therein, we observe that, on the contrastively trained CLIP representations, BASICS-SAE achieves remarkable reconstruction fidelity. It significantly reduces NMSE on CUB-200 (0.057 vs. 0.069 for BatchTopK) and remains almost on par with continuous models on ImageNet-1k (0.100 vs. 0.095). Crucially, this signal preservation translates into substantial downstream gains: on CUB-200, BASICS-SAE outperforms BatchTopK by $+6.4\%$ at 50-shot (78.9% vs. 72.5%), and leads all models on ImageNet-1k in both 10-shot (67.3%) and 50-shot (71.0%) regimes. A summary of the 10-shot performance on CUB for varying dictionary sizes and sparsity ratios is illustrated in Figure 1.

Our results on DINOv3 highlight that reconstruction NMSE alone does not dictate the semantic utility of the learned latents. Because DINOv3 feature distributions exhibit high capacity and unnormalized variance, binary quantization inherently incurs a higher reconstruction penalty on broad-domain datasets like ImageNet-1k (0.297 vs. 0.236 for BatchTopK). Yet, despite this higher NMSE, BASICS-SAE retains the semantics, reaching 79.4% 50-shot accuracy, matching or surpassing continuous variants (79.2% BatchTopK, 79.1% TopK) as well as the raw feature baseline (79.4%). Furthermore, on fine-grained CUB-200 features under DINOv3, our approach achieves the lowest NMSE (0.172) and highest classification accuracies in two settings (85.5% 10-shot, 87.5% 50-shot). However, continuous representations remain relatively competitive in the 1-Shot setting, with BASICS-SAE only coming on top in two of the four tested configurations (CUB with CLIP and ImageNet with DINOv3). We hypothesize that the semantic information potentially lost via binary encoding is then recovered by pooling information from several samples.

Across all configurations, BASICS-SAE highlights the capabilities of binary sparse representations relative to prior continuous relaxation techniques. Existing soft-binarization approaches (Gumbel and LWTA) suffer from signal degradation (e.g., NMSE up to 0.452 on DINOv3 ImageNet-1k) and degraded downstream transfer (60.8% and 65.2% 50-shot on CLIP ImageNet-1k). In contrast, a combinatorial coordinate-swap solver reliably closes the performance gap to continuous SAEs.

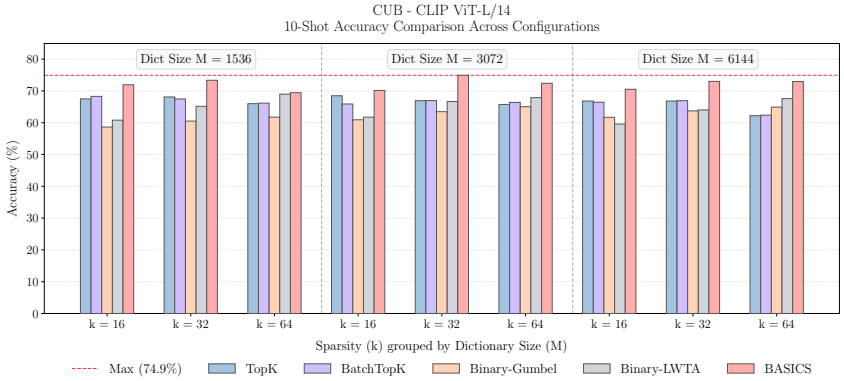

Figure 1: **10-shot accuracy on CUB across dictionary and sparsity budgets.** Evaluated on CLIP embeddings across dimensions $M \in \{1536, 3072, 6144\}$ and sparsity levels $k \in \{16, 32, 64\}$. BASICS-SAE consistently outperforms all baselines, demonstrating that direct combinatorial pursuit extracts more discriminative features without relying on continuous magnitude scaling.

Out-of-Distribution Robustness. As shown in Table 2, BASICS-SAE demonstrates strong out-of-distribution (OOD) robustness, outperforming prior binary formulations while matching or exceeding state-of-the-art continuous SAEs. Under CLIP embeddings, BASICS-SAE achieves top 50-shot accuracy across both OOD benchmarks, reaching 62.5% on ImageNet-A (+1.4% over TopK; +9.9% over Gumbel) and 80.7% on ImageNet-R (+1.2% over TopK; +8.3% over Gumbel). Under DINOv3 features, BASICS-SAE retains its edge on adversarial samples (75.8% vs. 74.8% TopK at 50-shot) while performing on par with continuous baselines on artistic renditions (87.3%). These results confirm that hard binary constraints do not compromise OOD robustness. While discrete coding could intuitively risk discarding subtle signals under domain shift, BASICS-SAE consistently matches or improves continuous baselines (+1.2% to +2.8% on CLIP; on par on DINOv3).

These empirical findings demonstrate that binary SAEs are not an inherently lossy compromise. Rather, BASICS-SAE establishes binary representations as a competitive alternative to continuous models across in- and out-of-distribution benchmarks. Having confirmed their functional viability, we next turn to their structural properties to evaluate concept alignment and interpretability.

Concept Alignment Score. As reported in Table 3, BASICS-SAE sets a new state-of-the-art across both CLIP and DINOv3 feature spaces, demonstrating particularly decisive gains on interventional steerability (TAPAScore). With CLIP embeddings, BASICS-SAE outperforms both continuous SAEs by over 8.8 percentage points at Top-10 (0.6208 vs. 0.5319) and the binary Gumbel (0.4739) by 14.7 points. Although LWTA achieves a strong Top-1 score, BASICS-SAE leads across the

Table 2: **Few-Shot Classification under Distribution Shift.** Out-of-Distribution (OOD) semantic evaluation via few-shot accuracy on ImageNet-A/ImageNet-R using CLIP/DINOv3 representations. Dictionary sizes are 3072 and 4096, respectively, with $k = 32$. Best results are shown in **bold**.

Method	ImageNet-A				ImageNet-R			
	CLIP		DINOv3		CLIP		DINOv3	
	10-Shot	50-Shot	10-Shot	50-Shot	10-Shot	50-Shot	10-Shot	50-Shot
Raw feature vector	53.7%	57.3%	75.0%	76.8%	68.6%	72.8%	87.4%	88.1%
TopK (Gao et al., 2025)	59.4%	61.1%	73.6%	74.8%	78.3%	79.5%	**86.8%**	**87.5%**
BatchTopK (Bussmann et al., 2024)	58.6%	59.7%	73.4%	74.6%	77.5%	79.2%	86.7%	87.3%
Gumbel (Quirke et al., 2025)	51.2%	52.6%	70.3%	71.1%	70.4%	72.4%	83.7%	84.2%
LWTA (Quirke et al., 2025)	51.0%	53.2%	65.8%	67.5%	70.4%	72.7%	81.9%	82.7%
BASICS	**60.8%**	**62.5%**	**73.9%**	**75.8%**	**78.7%**	**80.7%**	**86.8%**	87.3%

rest of metrics. With DINOv3, BASICS-SAE obtains a Top-10 TAPAScore of 0.6822, a $+15.5\%$ improvement over continuous TopK (0.5274) and $+9.2\%$ over LWTA (0.5902).

Table 3: **TAPAScores on CUB-200-2011 across CLIP and DINOv3.** Comparisons with $k = 32$ and dictionary sizes 3072 (CLIP) and 4096 (DINOv3). Best results are highlighted in **bold**.

Method	CLIP				DINOv3			
	Top-1	Top-3	Top-5	Top-10	Top-1	Top-3	Top-5	Top-10
TopK (Gao et al., 2025)	0.3532	0.5128	0.5240	0.5319	0.2233	0.3583	0.4153	0.5274
BatchTopK (Bussmann et al., 2024)	0.3979	0.5080	0.5332	0.5312	0.2380	0.3845	0.4350	0.5278
Gumbel (Quirke et al., 2025)	0.2022	0.3539	0.3740	0.4739	0.1377	0.2857	0.3645	0.4337
LWTA (Quirke et al., 2025)	**0.5112**	0.5590	0.5891	0.5513	0.2874	0.4780	0.5588	0.5902
BASICS	0.3949	**0.6117**	**0.6159**	**0.6208**	**0.3116**	**0.5728**	**0.6290**	**0.6822**

5 Limitations & Conclusions

While BASICS-SAE yields significant gains in feature quality and interpretability, it has four primary limitations. First, *computational latency*: because Gauss-Seidel sweeps update atom swaps sequentially, refinement cannot be fully parallelized within a single solve. Individual swaps are cheap, but this sequential dependency still introduces an inference overhead absent in feedforward SAEs (a $1.6\times$-$1.9\times$ per-batch latency, cf. Appendix A).Second, *sub-optimal swap dynamics*: evaluating single-atom replacements can trap the solver in sub-optimal configurations when escaping requires jointly displacing multiple redundant atoms (Sec. 3.1). Supporting small block-swaps or parallelizing decoupled swaps is a promising direction for improving convergence speed and solution quality. Third, *hyperparameter dependencies*: while our hyperparameters behave intuitively and avoid the training instabilities typical of gradient estimation, automated data-driven tuning could enable fully hands-off deployment. Fourth, *scope of validation*: our evaluation is limited to vision encoders, whereas prior negative results on binary SAEs were reported in language models (Quirke et al., 2025). Although we do not directly claim to overturn those domain-specific findings, the core mechanisms driving our gains, support selection procedure, balanced dictionary usage (Appendix A), and improved interventional alignment, are domain-agnostic. We expect these benefits to transfer to language models and leave direct evaluation in that domain to future work.

We introduced BASICS-SAE, a binary sparse autoencoder that derives active latent codes via a discrete coordinate-swap procedure rather than continuous gradient relaxations. Framing binary sparse coding as quadratic pseudo-Boolean optimization, our monotone Gauss-Seidel swap rule guarantees strict objective decrease at every step, bypassing temperature annealing, straight-through estimators, and training instabilities. Across diverse vision backbones and datasets, changing *how* binary codes are selected closes, and frequently reverses, the performance gap with continuous SAEs. BASICS-SAE matches or exceeds continuous TopK baselines in reconstruction fidelity while outperforming existing binary and continuous SAEs in few-shot classification and concept grounding. These results indicate that binary SAEs are not intrinsically flawed: prior performance bottlenecks stem primarily from optimization choices rather than fundamental limitations of binary representations.

AI USE STATEMENT

In this work, we used generative AI tools solely for editing and polishing aspects of text flow, grammar, and conciseness. We have not used generative AI tools for generating ideas, producing experimental results, writing code, or analyzing data. Additionally, all AI interaction was restricted to post-hoc text refinement. We have reviewed and manually verified all AI-assisted textual suggestions. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure full reproducibility, we provide complete, anonymized source code in the supplementary material, including dependencies, environment configurations, and a single-command script to reproduce our primary experimental results. Complete algorithmic formulations for our coordinate-swap procedure are detailed in Section 3 (Algorithms 2 and 1). Furthermore, all hyperparameter choices, dataset preprocessing protocols, backbone configurations, and evaluation setups are fully documented in Section 4 and Appendix A.

REFERENCES

Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11), 2006.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proc. CVPR*, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Endre Boros and Peter L Hammer. Pseudo-boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002.

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, et al. Towards monosemanticity: Decomposing language models with dictionary learning. `https://transformer-circuits.pub/2023/monosemantic-features/index.html`, 2023. Transformer Circuits Thread.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. In *Proc. NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024.

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

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Kjersti Engan, Sven Ole Aase, and John Håkon Husøy. Method of optimal directions for frame design. In *Proc. ICASSP*, 1999.

Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302 – 332, 2007. doi: 10.1214/07-AOAS131. URL `https://doi.org/10.1214/07-AOAS131`.

Wenjiang J Fu. Penalized regressions: the bridge versus the lasso. *Journal of computational and graphical statistics*, 7(3):397–416, 1998.

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

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 *Proc. ICCV*, 2021a.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proc. CVPR*, 2021b.

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

Prateek Jain, Ambuj Tewari, and Inderjit S. Dhillon. Orthogonal matching pursuit with replacement. In *Proc. NeurIPS*, 2011.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proc. ICLR*, 2017.

Jonas Klotz, Cassio Fraga Dantas, Pallavi Jain, Diego Marcos, and Begüm Demir. Evaluating the interpretability of sparse autoencoders with concept annotations. In *Proc. ECCV*, 2026.

Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. In *Proc. NeurIPS*, 2022.

Hyesu Lim, Jinho Choi, Jaegul Choo, and Steffen Schneider. Sparse autoencoders reveal selective remapping of visual concepts during adaptation. In *Proc. ICLR*, 2025.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of machine learning research*, 11(1), 2010.

Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23), 1997.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Hervé, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, Mathilde Caron, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2023.

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

Konstantinos P. Panousis and Sotirios Chatzis. Discover: Making vision networks interpretable via competition and dissection. In *Proc. NeurIPS*, 2023.

Konstantinos P. Panousis, Sotirios Chatzis, and Sergios Theodoridis. Nonparametric bayesian deep networks with local competition. In *Proc. ICML*, 2019.

Yagyensh C. Pati, Ramin Rezaiifar, and Perinkulam S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proc. Asilomar Conference on Signals, Systems and Computers*, 1993.

Lucia Quirke, Stepan Shabalin, and Nora Belrose. Binary sparse coding for interpretability. *arXiv preprint arXiv:2509.25596*, 2025.

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 *Proc. ICML*, 2021.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, Janos Kramar, Rohin Shah, and Neel Nanda. Improving sparse decomposition of language model activations with gated sparse autoencoders. In *Proc. NeurIPS*, 2024a.

Senthooran Rajamanoharan, Tom Lieberum, Nicholas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramar, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024b.

Sukrut Rao, Sweta Mahajan, Moritz Böhle, and Bernt Schiele. Discover-then-name: Task-agnostic concept bottlenecks via automated concept discovery. In *Proc. ECCV*, 2024.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. In *Proc. IJCV*, 2015.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. URL https://arxiv.org/abs/2508.10104.

Rupesh Kumar Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino J. Gomez, and Jürgen Schmidhuber. Compete to compute. In *Proc. NeurIPS*, 2013.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge J. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Jinming Wen and Haifeng Li. Binary sparse signal recovery with binary matching pursuit. *Inverse Problems*, 37(6):065014, 2021.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *Proc. ICLR*, 2023.

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

A Appendix

A.1 Experimental Details and Training Setup

In this section, we provide full implementation details, training hyperparameter configurations, baseline-specific training heuristics, and hardware specifications for all experiments.

Datasets and Dictionary Configurations

CUB-200-2011 Evaluation Suite: For fine-grained evaluations on CUB-200-2011, models were trained using a batch size of 256. We evaluated feature representations across two backbone feature extractors (CLIP ViT-L/14 and DINOv2) under three dictionary expansion factors relative to the embedding dimension d: $2\times$ ($M = 1536$), $4\times$ ($M = 3072$), and $8\times$ ($M = 6144$). For each dictionary scale, we tested target sparsity budgets of $k \in \{16, 32, 64\}$.

ImageNet-1K Scale Benchmark: For large-scale validation on ImageNet-1k, we adopted the widely established canonical benchmark configuration found in the sparse autoencoder literature: a dictionary expansion factor of $4\times$ the embedding dimension ($M = 3072$ for CLIP and $M = 4096$

for DINO) paired with a sparsity constraint of $k = 32$. Models on ImageNet-1k were trained with a mini-batch size of 4096 for 100 epochs. While we swept the learning rate across standard log-spaced intervals, downstream reconstruction and classification performance showed minimal sensitivity to learning rate variations on this dataset.

All models, baselines, and downstream few-shot evaluations were executed on a single workstation node equipped with an NVIDIA GPU with 48 GB VRAM (e.g., RTX A6000). All embeddings are pre-computed offline to minimize training overhead.

GRADIENT-BASED BASELINE TRAINING PROTOCOLS

All standard gradient-based baselines were trained using the AdamW optimizer.

- **TopK and BatchTopK SAEs:** Models were trained for up to 250 epochs with early stopping based on validation reconstruction loss. To mitigate dictionary unit inactivation (dead features), both variants were trained with the standard auxiliary dead-unit recovery loss $\mathcal{L}_{\text{aux}}$ weighted by $\beta = 0.03125$ $(1/32)$.
- **Binary Gumbel and LWTA:** Local Winner-Take-All (LWTA) and Binary Gumbel SAEs required twice the training duration for CUB (500 epochs) due to significantly slower optimization convergence. Binary Gumbel exhibited substantial optimization instability throughout training regardless of learning rate tuning, temperature annealing or warm-up schedules.

BASICS-SAE HYPERPARAMETER CONFIGURATION

For our proposed method, we train the models with the same number of epochs as the continuous baselines. **BASICS-SAE**, hyperparameter tuning demonstrated strong robustness across dataset domains and backbone encoders. While optimal hyperparameters can be fine-tuned per architecture, we establish a unified default parameter set that performs consistently across all benchmarks:

- Alternating pursuit steps: $n_{\text{alts}} = 1$
- Running statistics exponential decay: $\gamma = 0.95$
- Max Number of Swaps: *maxSwaps*$= 128$
- Ridge regularization penalty: $\lambda = 5 \times 10^{-3}$
- Decoder learning rate: $\eta_{\text{dec}} = 0.01$

A comprehensive sensitivity analysis and one-factor-at-a-time (OAT) ablation study isolating the individual impact of each hyperparameter is provided next.

HYPERPARAMETER ABLATION STUDY

Our hyperparameter ablation on CUB-200-2011 isolates each design component against our updated baseline configuration ($\gamma = 0.95, \lambda = 5 \times 10^{-3}$, *maxSwaps*$= 128, n_{\text{alts}} = 1$, *Random initialization*):

Impact of Refinement *maxSwaps*: The number of swaps is the primary factor preventing optimization failure. Restricting the solver to a purely myopic single-candidate swap (*maxSwaps*$= 1$) causes severe degradation: reconstruction NMSE rises to 0.2458, and 10-shot accuracy drops from 74.89% to 10.98%. Increasing *maxSwaps* to 128 enables the coordinate-swap mechanism to resolve feature co-dependencies across high-dimensional support candidates.

Initialization Strategy: Random dictionary initialization substantially outperforms the greedy TopK-of-h heuristic. TopK-of-h leads to a 6.81% drop in 10-shot accuracy (68.08% vs. 74.89%) and a higher reconstruction error (0.0643 vs. 0.0568), showing that greedy deterministic initialization introduces directional bias and early correlation traps.

Ridge Regularization Penalty (λ): Our default ridge penalty of $\lambda = 5 \times 10^{-3}$ delivers the highest downstream classification performance (48.53% 1-shot, 74.89% 10-shot). While a weaker penalty ($\lambda = 10^{-3}$) achieves a lower reconstruction error (0.0530 vs. 0.0568), it yields lower 1-shot accuracy (45.34%). Conversely, increasing λ to 10^{-2} leads to excessive regularization, degrading both reconstruction NMSE (0.0631) and downstream classification (74.25%).

Table 4: **Hyperparameter Ablation Study on CUB-200-2011** ($K = 3072, k = 32$). Each design parameter is isolated against the default reference configuration (†). The maximum number of swaps and support initialization are the most impactful factors, while a ridge penalty of $\lambda = 5 \times 10^{-3}$ with decay $\gamma = 0.95$ yields the best downstream classification metrics.

Ablated Parameter	Value	NMSE (↓)	1-Shot (%)	10-Shot (%)	50-Shot (%)
Default Baseline	*Default Set*	*0.0568*	*48.53*	*74.89*	*78.93*
Initialization	Random†	0.0568	48.53	74.89	78.93
	TopK-of-h	0.0643	36.29	68.08	75.20
Alternating Steps	1†	0.0568	48.53	74.89	78.93
	3	0.0604	46.55	73.77	77.51
maxSwaps	1	0.2458	2.69	10.98	11.31
	16	0.0777	42.63	71.62	75.41
	128†	0.0568	48.53	74.89	78.93
Ridge Penalty (λ)	1e-3	0.0530	45.34	74.47	78.13
	5e-3†	0.0568	48.53	74.89	78.93
	1e-2	0.0631	46.80	74.25	78.55
Statistics Decay (γ)	0.90	0.0578	47.51	74.68	79.24
	0.95†	0.0568	48.53	74.89	78.93
	0.99	0.0655	39.48	71.07	76.87

Online Statistics Decay Rate (γ): An exponential decay rate of $\gamma = 0.95$ yields the strongest overall balance. Slowing down the decay to $\gamma = 0.99$ severely reduces 1-shot (39.48%) and 10-shot accuracy (71.07%) due to overly rigid historical covariance estimates, while a faster decay ($\gamma = 0.90$) slightly degrades reconstruction (0.0578).

Alternating Optimization Steps (n_{alts}): Increasing inner alternating steps from $n_{\text{alts}} = 1$ to $n_{\text{alts}} = 3$ does not improve feature quality. Reconstruction NMSE increases slightly ($0.0568 \rightarrow 0.0604$), and 10-shot accuracy declines to 73.77%. A single alternating pass ($n_{\text{alts}} = 1$) is therefore both computationally optimal and effective for representation learning.

Statistical Variance Across Random Seeds

To verify that model performance is robust to initial dictionary support selections, we evaluate BASICS-SAE across six distinct random seeds ($N = 6$). As shown in Table 5, the model exhibits tight consistency across all runs, yielding an average reconstruction NMSE of 0.05692 ± 0.00015 and a 10-shot accuracy of $75.17\% \pm 0.28\%$. The minimal variance ($\sigma \leq 0.45\%$) demonstrates that our closed-form dictionary updates and online statistics decay reliably guide optimization toward consistent representation space regardless of initial random support draws.

Table 5: **Performance Stability Across Random Seeds on CUB-200-2011** ($K = 3072, k = 32$). We evaluate BASICS-SAE across six random seed initializations under the default hyperparameter setup ($\gamma = 0.95, \lambda = 5 \times 10^{-3}$, *MaxSwaps*$= 128, n_{\text{alts}} = 1$). The low standard deviation confirms stability and robustness to random support initialization.

Random Seed	NMSE (↓)	1-Shot (%)	10-Shot (%)	50-Shot (%)
Seed 0	0.05694	47.46	75.41	78.38
Seed 42	0.05685	48.05	75.39	79.26
Seed 54	0.05720	48.64	75.35	79.22
Seed 112	0.05677	47.65	75.23	78.89
Seed 256	0.05685	47.86	74.77	79.00
Seed 1234	0.05692	47.46	74.85	78.88
Mean ± Std	**0.05692 ± 0.00015**	**47.85 ± 0.45**	**75.17 ± 0.28**	**78.94 ± 0.32**

Inference Latency Analysis

To evaluate practical deployability, we benchmark the per-batch inference latency of BASICS-SAE against continuous and soft-binary baselines under identical hardware and batch settings (Table 6). Because BASICS-SAE employs a sequential coordinate-swap procedure to optimize the discrete k-cardinality support, it introduces a modest runtime overhead compared to single-pass feedforward encoders, requiring $40.10\,\text{ms}$ on CUB-200 ($K = 8192, k = 64$) and $48.25\,\text{ms}$ on ImageNet-1k ($K = 3072, k = 32$), representing an approximate $1.6\times$ to $1.9\times$ latency increase over standard TopK. Nevertheless, this runtime increase represents an acceptable trade-off considering the empirical gains across both paradigm classes. When compared to feedforward binary SAEs, BASICS-SAE eliminates unit hyper-activity and feature collapse; when benchmarked against continuous SAEs, it provides superior feature disentanglement and interventional steerability.

Table 6: **Inference Batch Latency Benchmarks.** Comparison of per-batch inference latency (in ms) across vision backbones and datasets. All benchmarks were run under identical conditions.

Method	**CUB-200 (DINOv3)** ($K = 8192, k = 64$)	**ImageNet-1k (CLIP)** ($K = 3072, k = 32$)
TopK	23.91 ms	25.14 ms
BatchTopK	24.13 ms	24.89 ms
Gumbel-STE	24.16 ms	25.55 ms
Binary LWTA (GroupMax)	24.61 ms	25.37 ms
BASICS	40.10 ms	48.25 ms

Feature Prevalence

Figure 2 shows the distribution of per-unit activation prevalence for each method, on ImageNet-1k with CLIP ViT-L/14 ($K = 3072$, $k = 32$, left) and on CUB with DINOv3 ViT-L/16 ($K = 4096$, $k = 32$, right). In both settings, BASICS produces the narrowest, most sharply peaked distribution of any method, concentrated close to the value a perfectly evenly-used dictionary would produce (k/K, roughly 1.0% for the CLIP/ImageNet setting and 0.8% for the DINOv3/CUB setting). This indicates that the vast majority of BASICS units fire at close to their fair share of examples, rather than a small subset dominating while the rest lie mostly dormant. TopK and BatchTopK, by contrast, exhibit substantially broader, skewed distributions in both settings: a large mass of units sits below the fair-share rate, while a long right tail extends out to prevalence values an order of magnitude or more above it, a signature of a small number of generically-firing "hub" units alongside a much larger population of under-used ones, consistent with feature-absorption effects previously reported for continuous TopK SAEs. The relaxation-based binary baselines fall between these extremes: Binary-Gumbel tracks BASICS closely on ImageNet but is noticeably broader and right-shifted on CUB, while Binary-LWTA is consistently broader than BASICS in both settings and, on CUB, shifts its peak past the fair-share value entirely. BASICS maintains the tightest concentration across two different backbones and two datasets of very different scale and class structure suggests this is a structural property of exact, swap-based code selection rather than an artifact of any one dataset or embedding space.

A.2 Concept-Latent Matching

The *Matching Score* Measures the observational correspondence between latent activations and ground-truth concept annotations across samples in a test set to associate to each annotated concept the best-matching latents. A many-to-one matching procedure is used, where each ground-truth concept is matched to a coalition of a few latents combined (here we use 1, 3, 5 and 10 latents). The coalitions are formed using the sequential FBMP (Fully-Binary Matching Pursuit) procedure introduced in Klotz et al. (2026). A F_β-score is used as the similarity criterion for latent selection, with $\beta = 0.5$ as recommended by Klotz et al. (2026). The reported values in Table 7 are the average $F_{0.5}$ score between selected latent coalitions and the corresponding ground-truth concept. The matched latents are then used to compute the interventional consistency metric *TAPAScore* reported in the main paper.

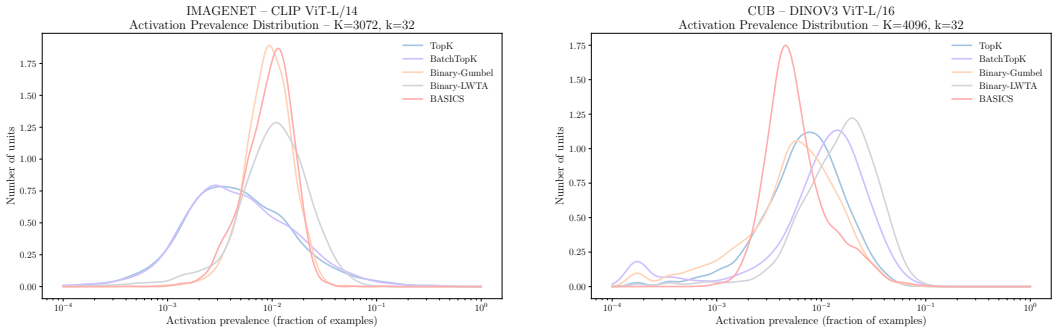

Figure 2: Distribution of per-unit activation prevalence (fraction of examples on which a unit fires) for ImageNet-1k with CLIP ViT-L/14, on ImageNet-1k with CLIP ViT-L/14 ($K = 3072$, $k = 32$, left) and on CUB with DINOv3 ViT-L/16 ($K = 4096$, $k = 32$, right). A narrower distribution centered near k/K ($\approx 10^{-2}$ on the left plot and $8{\times}10^{-3}$ on the right) indicates more evenly-distributed dictionary usage across units; a broader, heavier-tailed distribution indicates usage concentrated in a subset of frequently-firing units while others remain comparatively idle. BASICS yields the most concentrated usage distribution of all methods compared, in both settings.

Table 7: **FBMP ($F_{0.5}$) evaluation scores on CUB-200-2011 across CLIP and DINOv3.** Comparisons across embeddings with $k = 32$ and dictionary sizes 3072 and 4096 respectively. Best results per backbone in **bold**.

		CLIP				DINOv3			
Metric	**Method**	**Top-1**	**Top-3**	**Top-5**	**Top-10**	**Top-1**	**Top-3**	**Top-5**	**Top-10**
Matching	TopK (Gao et al., 2025)	0.2613	0.3362	0.3510	0.3571	0.2164	0.3297	0.3639	0.3894
	BatchTopK (Bussmann et al., 2024)	0.2596	0.3295	0.3439	0.3502	0.2211	0.3242	0.3507	0.3685
	Gumbel (Quirke et al., 2025)	0.2024	0.3071	0.3420	**0.3727**	0.2182	0.3358	**0.3752**	**0.4067**
	LWTA (Quirke et al., 2025)	0.2406	0.3195	0.3340	0.3406	0.2168	0.3229	0.3522	0.3743
	BASICS	**0.2640**	**0.3380**	**0.3512**	0.3573	**0.2486**	**0.3434**	0.3668	0.3830

On static *Matching*, BASICS-SAE outperforms all baselines across almost all ranks on CLIP features, while securing the highest Top-1 (0.2486) and Top-3 (0.3434) matching scores on DINOv3.

A.3 Additional Experiments: DINOv2 ViT-L/14 Backbone

To complement our evaluations on contrastively trained vision-language models (CLIP) and recent self-supervised architectures (DINOv3), we extend our empirical evaluation to DINOv2 (ViT-L/14) (Oquab et al., 2023). DINOv2 features exhibit distinct visual semantic clustering formed purely through self-supervised learning without textual supervision. Evaluating on DINOv2 serves as a stringent test of completeness, verifying whether binary sparse autoencoder representations generalize effectively across structurally diverse feature spaces.

On reconstruction and few-shot classification (Table 8), BASICS-SAE reproduces the strong performance patterns observed on CLIP and DINOv3. On fine-grained evaluation (CUB), NMSE (0.139) remains virtually on par with the strongest continuous baseline, BatchTopK (0.134), while few-shot accuracy leads across all five methods at every shot count, exceeding the top competing baseline by $+0.8\%$, $+2.5\%$, and $+2.8\%$ at 1-, 10-, and 50-shot respectively, and outperforming Gumbel by up to 9.8 percentage points at 50-shot.

This trend extends to broad-domain visual representations on ImageNet-1k (Table 8): BASICS-SAE achieves the highest 10-shot (74.8%) and 50-shot (75.6%) classification performance among all dictionary models, matching or exceeding even the uncompressed raw backbone baseline (75.3%) and continuous SAEs (75.5%). While continuous variants achieve lower reconstruction error on ImageNet due to unrestricted continuous activation magnitudes, BASICS-SAE achieves a substantially lower NMSE (0.291) than other binary SAE approaches (Gumbel at 0.369 and LWTA at 0.417). This demonstrates that discrete coordinate pursuit preserves complex, high-dimensional visual manifolds significantly better than relaxed soft-binarization techniques.

On concept alignment (Table 9), the two scores diverge. Klotz et al. (2026) deliberately separate the matching step, i.e., the correspondence between latents and annotated concepts used to identify *which* latent stands for which concept, from TAPAScore, which in turn tests whether the matched latent actually responds selectively and in the correct direction under targeted attribute perturbations; the authors explicitly caution that the former is a necessary prerequisite for computing the latter, not a proxy for it, and that a high matching score need not imply high perturbation alignment. Our DINOv2 results echo exactly this dissociation: under the Matching score, LWTA and Gumbel attain the best values, and BASICS-SAE trails most binary and continuous baselines, edging out only Gumbel at Top-1. Yet on TAPAScore, the metric the authors identify as the actual interpretability measure, BASICS-SAE reverses this ordering entirely, achieving the best score at every rank and improving Top-10 alignment to 0.4163 from BatchTopK's 0.3519 and LWTA's 0.2506. To a lesser degree, a similar dissociation between matching and interventional alignement (TAPAS) happens on CLIP and DINOv3 results: while Gumbel (on DINOv3) or LWTA (on DINOv2) occasionally achieve higher matching scores by fitting superficial latent-concept correlations, their representations degrade when subjected to targeted attribute perturbations.

Table 8: **NMSE and Few-Shot Classification on CUB-200-2011 and ImageNet-1k.** Comparison across DINOv2 (ViT-L/14) embeddings with $k = 32$ and dictionary size 4096. **BASICS-SAE** consistently matches or even outperforms binary baselines and continuous models in few-shot performance. Best results in **bold**.

Method	CUB				ImageNet-1k			
	NMSE ↓	1-Shot↑	10-Shot↑	50-Shot↑	NMSE ↓	1-Shot↑	10-Shot↑	50-Shot↑
Raw feature vector	-	69.2%	76.7%	78.4%	-	59.6%	74.2%	75.3%
TopK (Gao et al., 2025)	0.148	70.8%	80.7%	82.6%	0.206	**60.2%**	74.5%	75.4%
BatchTopK (Bussmann et al., 2024)	**0.134**	70.5%	81.0%	83.0%	**0.200**	**60.2%**	74.4%	75.5%
Gumbel (Quirke et al., 2025)	0.270	67.4%	73.8%	76.2%	0.369	61.0%	72.7%	73.4%
LWTA (Quirke et al., 2025)	0.225	60.5%	80.3%	83.2%	0.417	53.0%	71.2%	72.9%
BASICS	0.139	**71.6%**	**83.5%**	**86.0%**	0.291	**60.2%**	**74.8%**	**75.6%**

Table 9: **FBMP ($F_{0.5}$) evaluation scores on CUB-200-2011 with DINOv2 backbone.** Comparisons across embeddings with $k = 32$ and dictionary size 4096. Best results per backbone in **bold**.

Metric	Method	DINOv2			
		Top-1	Top-3	Top-5	Top-10
Matching	TopK (Gao et al., 2025)	0.2133	0.3288	0.3651	0.3924
	BatchTopK (Bussmann et al., 2024)	0.2166	0.3257	0.3583	0.3820
	Gumbel (Quirke et al., 2025)	0.2071	0.3339	0.3746	0.4057
	LWTA (Quirke et al., 2025)	**0.2333**	**0.3377**	**0.3616**	**0.3727**
	BASICS	0.2119	0.3172	0.3474	0.3699
TAPAS	TopK (Gao et al., 2025)	0.1391	0.2216	0.2618	0.3164
	BatchTopK (Bussmann et al., 2024)	0.1384	0.2363	0.2888	0.3519
	Gumbel (Quirke et al., 2025)	0.0743	0.1255	0.1735	0.2284
	LWTA (Quirke et al., 2025)	0.1119	0.2554	0.2574	0.2506
	BASICS	**0.1971**	**0.3716**	**0.3863**	**0.4163**

A.4 Matching Pursuit SAE (MPSAE) Baseline Details and Limitations

For completeness, we include empirical comparisons against the Matching Pursuit Sparse Autoencoder (MPSAE) (Costa et al., 2025). However, key experimental settings and hyperparameter specifications were not fully detailed in the original work, and their evaluation setup does not directly correspond to our benchmarking protocols.

To ensure a fair evaluation within our framework, we implemented a standardized PyTorch version of MPSAE using the official implementation (Costa et al., 2025); we set the maximum pursuit iterations $T_{\max}$ to the desired sparsity rate according to each experiment. The remaining training

parameters (optimizer, learning rate schedule, batch size, and total steps) were kept identical to our primary models to maintain consistent compute budgets across all baselines.

Table 10: **NMSE and Few-Shot Classification on CUB-200-2011 and ImageNet-1k.** Comparison of MP-SAE and BASICS across CLIP and DINOv3 backbones on CUB-200 and ImageNet for $k = 32$ and dictionary size 3072 and 4096 respectively.

				Few-Shot Accuracy (%) ↑		
Dataset	**Backbone**	**Model**	**NMSE ↓**	**1-shot**	**10-shot**	**50-shot**
CUB-200	CLIP	MP-SAE (Costa et al., 2025)	0.068	40.2%	52.8%	57.1%
		BASICS	**0.057**	**48.5%**	**74.9%**	**78.9%**
	DINOv3	MP-SAE (Costa et al., 2025)	0.198	69.7%	75.3%	77.8%
		BASICS	**0.172**	**72.4%**	**85.5%**	**87.5%**
ImageNet	CLIP	MP-SAE (Costa et al., 2025)	**0.084**	38.8%	55.6%	59.4%
		BASICS	0.100	**41.8%**	**67.3%**	**71.0%**
	DINOv3	MP-SAE (Costa et al., 2025)	**0.225**	**62.5%**	77.0%	78.3%
		BASICS	0.297	60.5%	**77.8%**	**79.4%**

Across both fine-grained (CUB-200) and large-scale (ImageNet-1k) visual representations, BASICS demonstrates significant advantages over greedy search via MP-SAE. On CUB-200, BASICS consistently outperforms MP-SAE in both reconstruction fidelity (NMSE) and downstream few-shot classification across CLIP and DINOv3 backbones. This suggests that for fine-grained feature spaces with subtle semantic distinctions, discrete coordinate search via single-swap local moves extracts more discriminative representations than sequential matching pursuit.

Reconstruction versus downstream utility. On ImageNet-1k, the picture differs meaningfully between backbones. On CLIP embeddings, BASICS-SAE wins every few-shot metric, including 1-shot (41.8% vs. 38.8%), despite a higher NMSE (0.100 vs. 0.084). On DINOv3 embeddings, MP-SAE achieves the best reconstruction of any method we evaluate, including the continuous baselines in Table 1, and edges out BASICS-SAE (and other continuous SAEs) at 1-shot (62.5% vs. 60.5%); BASICS-SAE recovers the advantage at 10- and 50-shot (77.8% vs. 77.0%, 79.4% vs. 78.3%). Notably, while MP-SAE predicts continuous feature coefficients, BASICS-SAE operates under a strict binary constraint. BASICS-SAE regaining the lead in higher-shot regimes reinforces our main hypothesis: while continuous activations retain a slight edge in 1-shot in some cases evaluation due to magnitude information, binary representations fully recover this gap, and surpass continuous models, once semantic information is pooled across multiple training samples.

B Binary SAEs Formulations

B.1 Binary TopK Gumbel SAE

The Binary TopK Gumbel Sparse Autoencoder (SAE) learns a sparse binary representation $\mathbf{z} \in \{0,1\}^K$ of an input vector $x \in \mathbb{R}^d$. It combines a stochastic continuous relaxation based on Logistic noise (the Gumbel-Sigmoid trick) with a hard Top-K operator, using a Straight-Through Estimator (STE) to enable end-to-end gradient backpropagation.

Given an input vector x, we first center the input relative to the decoder bias $\mathbf{b}_{\text{dec}} \in \mathbb{R}^d$ and compute the pre-activation logits $\mathbf{z} \in \mathbb{R}^K$:

$$\mathbf{z} = (x - \mathbf{b}_{\text{dec}})\mathbf{W}_{\text{enc}} + \mathbf{b}_{\text{enc}} \tag{11}$$

where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{d \times K}$ is the encoder weight matrix and $\mathbf{b}_{\text{enc}} \in \mathbb{R}^K$ is the encoder bias.

To allow backpropagation through stochastic discrete decisions during training, we inject Logistic noise into the logits. Let $u_i \sim \text{Uniform}(\epsilon, 1-\epsilon)$ with a small stabilization constant $\epsilon = 10^{-10}$. The

noisy, temperature-scaled logits $\tilde{\mathbf{z}}$ and continuous soft activations $\mathbf{z}^{\text{soft}} \in (0,1)^K$ are computed as:

$$\tilde{z}_i = \begin{cases} \dfrac{z_i + \text{logit}(u_i)}{\tau} & \text{during training} \\ \dfrac{z_i}{\tau} & \text{during evaluation} \end{cases} \tag{12}$$

$$\mathbf{z}^{\text{soft}} = \sigma(\tilde{\mathbf{z}}) \tag{13}$$

where $\sigma(\cdot)$ denotes the element-wise sigmoid function and $\tau > 0$ is the temperature hyperparameter controlling the sharpness of the relaxation.

Hard Top-K selection and straight-through estimation. To enforce strict sparsity, a hard binary mask $\mathbf{z}^{\text{hard}} \in \{0,1\}^K$ is constructed by selecting the K largest entries of $\tilde{\mathbf{z}}$:

$$z_i^{\text{hard}} = \begin{cases} 1 & \text{if } \tilde{z}_i \in \text{Top-}K(\tilde{\mathbf{z}}) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The final activation vector combines the hard mask and the soft relaxation via a straight-through estimator, so that the forward pass is exactly binary while gradients flow through $\mathbf{y}_{\text{soft}}$:

$$\mathbf{z}^{\text{out}} = \mathbf{z}^{\text{hard}} - \text{stopgrad}(\mathbf{z}^{\text{soft}}) + \mathbf{z}^{\text{soft}} \tag{15}$$

Temperature annealing. The temperature τ is annealed geometrically over training, from $\tau_{\text{start}} = 2.0$ at the start of training to $\tau_{\text{end}} = 0.1$ at the final epoch, following $\tau_t = \tau_{\text{start}} \cdot (\tau_{\text{end}}/\tau_{\text{start}})^{t/T}$ for epoch t of T total epochs. Higher temperatures early in training smooth the relaxation and provide a more informative gradient signal while the encoder is far from convergence; annealing τ toward 0.1 sharpens $\mathbf{y}_{\text{soft}}$ toward the hard mask $\mathbf{m}$ by the end of training, reducing the forward-backward mismatch inherent to the straight-through estimator.

Reconstruction and objective. The reconstructed input $\hat{x} \in \mathbb{R}^d$ is decoded as $\hat{x} = \mathbf{W}_{\text{dec}}^\top \mathbf{z} + \mathbf{b}_{\text{dec}}$, and the total loss, averaged over a batch of N examples, is

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \hat{x}_n\|_2^2 \tag{16}$$

Note that $\|\mathbf{z}\|_0$ can be strictly less than K in practice, since $\mathbf{z}^{\text{hard}}$ selects the K largest entries of $\tilde{\mathbf{z}}$ regardless of their sign.

B.2 Binary Local Winner-Take-All (LWTA) SAE

The Local Winner-Take-All (LWTA) Gumbel Sparse Autoencoder learns a sparse binary representation $\mathbf{z}^{\text{out}} \in \{0,1\}^K$ of an input vector $x \in \mathbb{R}^d$. It partitions the dictionary space of dimension K into k disjoint feature groups of size $M = K/k$, selecting exactly one active binary feature per group via a categorical Gumbel-Softmax distribution. This guarantees an exact $L_0 = k$ activation constraint across the dictionary, matching the exact sparsity constraint of a TopK SAE.

Grouping and logit computation. Given an input vector x, we first center the input relative to the decoder bias $\mathbf{b}_{\text{dec}} \in \mathbb{R}^d$ and compute the pre-activation logits $\mathbf{z} \in \mathbb{R}^K$:

$$\mathbf{z} = (x - \mathbf{b}_{\text{dec}})\mathbf{W}_{\text{enc}} + \mathbf{b}_{\text{enc}} \tag{17}$$

where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{d \times K}$ is the encoder weight matrix and $\mathbf{b}_{\text{enc}} \in \mathbb{R}^K$ is the encoder bias.

The logits vector $\mathbf{z}$ is reshaped into k distinct group vectors $\mathbf{z}^{(g)} \in \mathbb{R}^M$ for $g \in \{1, 2, \ldots, k\}$:

$$\mathbf{z} = \left[\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(k)}\right] \tag{18}$$

Categorical Gumbel-Softmax sampling per group. To allow backpropagation through stochastic discrete decisions during training, a categorical decision is modeled within each group g using a Gumbel-Softmax relaxation. Let $g_i = -\log(-\log(u_i + \epsilon) + \epsilon)$ with $u_i \sim \text{Uniform}(0, 1)$ and stabilization constant $\epsilon = 10^{-10}$. The noisy, temperature-scaled logits $\tilde{\mathbf{z}}^{(g)}$ and continuous soft group activations $\mathbf{z}^{\text{soft},(g)} \in (0, 1)^M$ are computed as:

$$\tilde{z}_i^{(g)} = \begin{cases} \dfrac{z_i^{(g)} + g_i}{\tau} & \text{during training} \\ \dfrac{z_i^{(g)}}{\tau} & \text{during evaluation} \end{cases} \tag{19}$$

$$z_{\text{soft},i}^{(g)} = \frac{\exp\left(\tilde{z}_i^{(g)}\right)}{\sum_{j=1}^{M} \exp\left(\tilde{z}_j^{(g)}\right)} \tag{20}$$

where $\tau > 0$ is the temperature hyperparameter controlling the sharpness of the relaxation.

Hard selection and straight-through estimation. A hard one-hot mask $\mathbf{z}^{\text{hard},(g)} \in \{0, 1\}^M$ is constructed by selecting the argmax within each group:

$$z_i^{\text{hard},(g)} = \begin{cases} 1 & \text{if } i = \arg\max_j \left(\tilde{z}_j^{(g)}\right) \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

The group feature activation vector combines the hard mask and soft relaxation via a straight-through estimator:

$$\mathbf{z}^{\text{out},(g)} = \mathbf{z}^{\text{hard},(g)} - \text{stopgrad}\left(\mathbf{z}^{\text{soft},(g)}\right) + \mathbf{z}^{\text{soft},(g)} \tag{22}$$

Concatenating all group vectors yields the global binary feature activation vector $\mathbf{z}^{\text{out}} \in \{0, 1\}^K$:

$$\mathbf{z}^{\text{out}} = \left[\mathbf{z}^{\text{out},(1)}, \mathbf{z}^{\text{out},(2)}, \ldots, \mathbf{z}^{\text{out},(k)}\right] \tag{23}$$

Because exactly one feature is active in each of the k groups, the active feature count is strictly deterministic ($L_0 = k$).

Reconstruction and objective. The reconstructed input $\hat{x} \in \mathbb{R}^d$ is decoded as $\hat{x} = \mathbf{W}_{\text{dec}}^\top \mathbf{z}^{\text{out}} + \mathbf{b}_{\text{dec}}$, and the total loss, averaged over a batch of N examples, is

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \hat{x}_n\|_2^2 \tag{24}$$