KEEP FEATURES DENSE AND EXPLANATIONS SPARSE: ADAPTIVE INTERPRETABILITY AT TEST TIME

Anonymous authors
Paper under double-blind review

ABSTRACT

Simple data-derived linear directions can expose highly interpretable structure in neural representations, yet their activations are typically signed and dense. Sparse autoencoders instead learn non-negative sparse features, coupling feature learning with a sparsity constraint. We ask whether sparsity is needed in the representation at all. More broadly, test-time computation remains comparatively underexplored in interpretability. We introduce TTS-AE, which learns a dense, non-negative reconstructive representation from data-grounded directions and introduces sparsity only when an explanation is requested. At test time, *Test-Time Sparsification* (TTS) selects a compact, input-specific subset of features and infers the remaining activations from the covariance structure of the dense representation until a desired reconstruction fidelity is reached. Across three model–dataset pairs, the resulting dense features achieve higher semantic-quality scores than the evaluated nonlinear sparse autoencoder baselines, while TTS reaches matched fidelity with substantially fewer positively activated concepts than sparse autoencoders and sparse approximation baselines. The same dense concept space also supports covariance-aware interventions and sparse target-specific global explanations. Our results suggest that interpretability and explanation sparsity need not be learned jointly: dense representations can support sparse explanations constructed adaptively through additional computation at test time.

1 INTRODUCTION

The epistemic opacity of Deep Neural Networks (DNNs) arises in part from the difficulty of assigning meaningful semantics to their internal representations. Interpretability methods therefore seek feature spaces whose coordinates correspond to recognizable concepts and can be related to model behavior (Zednik, 2021; Duede, 2022). Sparsity is a common inductive bias for this purpose: if only a few features are active for each input, they may be easier to isolate and inspect. Sparse Autoencoders (SAEs) follow this principle by jointly learning a feature dictionary, i.e., a set of directions in representation space and imposing sparsity on its latent activations (Ng et al., 2011; Cunningham et al., 2023). Yet sparsity determines *how many* features are active, not *what those features represent*. This raises a question: is sparsity necessary for learning interpretable features, or can sparsity instead be constructed adaptively when an explanation is requested?

An alternative view is that semantic structure is already encoded in the geometry of learned representations. During training, deep networks reshape feature spaces, such that task-relevant variation becomes easier to separate and decode. From a manifold perspective, related inputs occupy structured regions of representation space whose positions, dimensions, and correlations become progressively more separable across layers (Cohen et al., 2020). Distances, neighborhoods, and directions in these spaces can therefore carry meaningful structure. This motivates feature-extraction methods grounded in the empirical representation geometry, including clustering-, prototype-, and data-anchored dictionary approaches (Fel et al., 2025). From this perspective, sparsity is not required to learn interpretable structure, but may instead determine how much of that structure is exposed in an explanation. Hence, instead of fixing explanation sparsity once for all inputs during training, we may shift sparsification to test time, when both the specific input and desired explanation fidelity are known.

Building on this idea, we introduce TTS-AE, which decouples feature learning from explanation sparsity and uses test-time computation to construct explanations adaptively. TTS-AE learns a dense,

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.

non-negative, reconstructive feature space initialized from clustering-based, data-grounded directions, without imposing sparsity during training. Sparsity is introduced only when an explanation is requested through *Test-Time Sparsification* (TTS). Given an input, TTS greedily selects reconstruction-relevant features while using the covariance structure of the dense representation to estimate the remaining activations, stopping once a chosen reconstruction fidelity is reached. Across multiple architectures and datasets, the resulting dense features are more interpretable than nonlinear sparse autoencoder features, while TTS produces substantially more compact explanations than sparse autoencoders and sparse reconstruction baselines at matched fidelity. The same dense concept space further supports targeted interventions and sparse global explanations. Our contributions are:

1. We provide evidence that sparsity is not required for learning interpretable features: directions grounded in neural representation geometry retain strong semantic structure in a dense, non-negative feature space.
2. We introduce *Test-Time Sparsification*, taking a step toward test-time computation for interpretability by constructing compact, input-specific explanations at a controllable reconstruction fidelity.
3. Across multiple architectures and datasets, we show that separating feature learning from explanation sparsity yields more interpretable features and substantially more compact explanations than nonlinear sparse autoencoders and sparse-reconstruction baselines, while the same dense representation supports targeted interventions and sparse global explanations.

2 Related Work

We focus on three lines of work most relevant to our approach: geometry-grounded feature extraction, sparse representation learning, and semantic feature validation. Broader background on mechanistic interpretability, concept-based methods, sparse decomposition, and global explanations is provided in Appendix A.

Geometry-Grounded Feature Extraction. Interpretability methods can extract meaningful structure directly from neural representations, including clustering-based concept discovery (Dreyer et al., 2024) and non-negative matrix factorization (Lee & Seung, 1999). Learned representations often organize task-relevant information geometrically, making semantic structure accessible through directions, neighborhoods, and low-dimensional subspaces (Cohen et al., 2020). Recent approaches make this connection explicit: Archetypal SAE anchors dictionary elements to observed activations (Fel et al., 2025), while recent analyses suggest that concepts may occupy structured low-dimensional manifolds rather than isolated feature directions (Bhalla et al., 2026).

Sparse Representation Learning. Sparse Autoencoders (SAEs) decompose neural representations into selective latent features (Ng et al., 2011; Cunningham et al., 2023; Bricken et al., 2023), with modern variants imposing sparsity through activation penalties, Top-K operators, or learned thresholds (Gao et al., 2025; Rajamanoharan et al., 2024). Recent work shows that SAE architecture affects the geometry of recoverable concepts (Hindupur et al., 2026), while densely activating features can retain meaningful structure (Sun et al., 2026). Closer to our setting, MP-SAE performs Matching Pursuit over an SAE-learned dictionary (Costa et al., 2025), while ITDA constructs a dictionary directly from model activations and uses Matching Pursuit to obtain sparse decompositions at inference time (Leask et al., 2025).

Feature Description and Semantic Validation. Neural features are commonly interpreted through activation-maximizing examples and automatically generated textual descriptions (Olah et al., 2017; Bykov et al., 2023b; Oikarinen & Weng, 2022). Because plausible descriptions need not faithfully characterize feature behavior, quantitative evaluation methods test whether descriptions predict or distinguish feature-associated inputs and whether the evaluation criteria themselves are reliable (Kopf et al., 2024; Oikarinen et al., 2025). We therefore evaluate semantic quality quantitatively rather than relying on qualitative inspection alone.

3 Method

TTS-AE separates two objectives that are often coupled in sparse representation learning: learning an interpretable feature space and producing sparse explanations. Given an input $x \in \mathcal{X}$, let

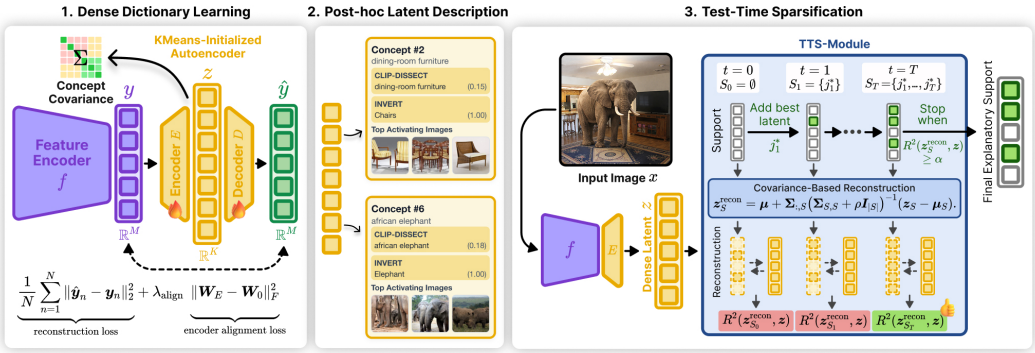

Figure 1: **Overview of TTS-AE.** TTS-AE learns a dense, information-preserving concept representation and enforces sparsity only at inference time. Step 1: A KMeans-initialized autoencoder is trained to reconstruct internal features. Step 2: Latent variables are linked to human-interpretable concept descriptions using post-hoc labeling. Step 3: At inference time, the TTS module greedily selects a small subset of active concepts and reconstructs the remaining latent state using covariance structure, yielding a sparse explanation with controlled feature-space fidelity.

$\boldsymbol{y} = f(x) \in \mathbb{R}^M$ denote an internal representation of a fixed neural network. TTS-AE first learns a dense, non-negative, reconstructive feature space initialized from representative directions obtained by clustering the observed representations. It then produces sparse explanations only at test time by selecting the features needed to reconstruct $\boldsymbol{y}$ to a user-specified fidelity. Figure 1 illustrates the full procedure.

Geometry-Grounded Dense Feature Learning. Let $\{\boldsymbol{y}_n\}_{n=1}^N$ be the standardized representations from a reference dataset. We first extract K representative directions from their empirical geometry and collect them as rows of

$$\boldsymbol{W}_0 \in \mathbb{R}^{K \times M}.$$

We obtain $\boldsymbol{W}_0$ by using spherical KMeans; however, other extractors can also be used. Although these directions can span the original representation, direct projections onto these directions are signed, assigning positive and negative coefficients to opposite sides of the same feature axis. To obtain one-sided, semantically consistent feature activations while retaining reconstructive capacity, we use these directions to initialize a non-negative autoencoder.

$$\boldsymbol{z}_n = \mathrm{ReLU}(\boldsymbol{W}_E \boldsymbol{y}_n), \qquad \hat{\boldsymbol{y}}_n = \boldsymbol{W}_D \boldsymbol{z}_n + \boldsymbol{b}_D, \tag{1}$$

where $\boldsymbol{W}_E \in \mathbb{R}^{K \times M}$ and $\boldsymbol{W}_D \in \mathbb{R}^{M \times K}$ are trainable. We optimize

$$\mathcal{L}_{\mathrm{AE}} = \frac{1}{N} \sum_{n=1}^{N} \|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2^2 + \lambda_{\mathrm{align}} \|\boldsymbol{W}_E - \boldsymbol{W}_0\|_F^2. \tag{2}$$

The reconstruction term preserves information from the original representation, while the alignment term keeps the learned encoder close to its data-grounded initialization. Importantly, Eq. 2 contains no sparsity penalty. Non-negativity constrains the activation domain, while the representation itself is deliberately allowed to remain dense. Additional extractor details are provided in Appendix G, and the effect of the alignment strength λ_{align} is analyzed in Appendix G.6.

Feature Labeling. After learning the feature space, each latent coordinate is assigned a human-readable semantic description using post-hoc labeling. Labels are obtained from the activation behavior of each feature, for example through top-activating images, label associations, CLIP-based descriptions, or VLM-based annotation. This stage is purely descriptive: the assigned labels do not affect feature learning, latent activations, or subsequent TTS selection. Our VLMGrid labeling procedure and implementation details are provided in Appendix B.

Test-Time Sparsification. Given a dense latent representation $\boldsymbol{z} = E(\boldsymbol{y}) \in \mathbb{R}^K$, a faithful explanation may require only a small subset of its features. For a desired fidelity level $\alpha \in (0, 1)$, TTS constructs an input-specific support

$$\mathbb{S}_\alpha \subseteq \{1, \ldots, K\},$$



Figure 2: **A single dense concept space supports both sparse local explanations and coherent global structure.** For a CLIP ViT-B/32 representation with 5,000 KMeansAE concepts, TTS reaches $R^2 \approx 0.8$ after selecting 617 coordinates during its search, but only two remain active after refitting; the other 615 receive zero weight. These two concepts capture the indoor scene and the elephant, while MP and the evaluated SAEs require larger active supports at approximately matched fidelity. Their semantics are supported by spatial responses, CLIP-Dissect and INVERT labels, and top-activating images. In the concept atlas (Appendix C), they also lie in coherent neighborhoods corresponding to interior spaces and animal-related concepts.

containing only the features needed to explain the representation at that fidelity.

TTS is motivated by a simple covariance-based model of the latent space. We treat $\boldsymbol{z}$ as a sample from a latent distribution with empirical mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ estimated once from the reference set of latent representations. For a given input, TTS retains the activations $\boldsymbol{z}_{\mathbb{S}}$ of the selected features, and uses the covariance with the remaining features to reconstruct the full latent representation, rather than setting the unselected features to zero:

$$\boldsymbol{z}_{\mathbb{S}}^{\text{rec}} = \boldsymbol{\mu} + \boldsymbol{\Sigma}_{:,\mathbb{S}} \left(\boldsymbol{\Sigma}_{\mathbb{S},\mathbb{S}} + \rho \boldsymbol{I}\right)^{-1} (\boldsymbol{z}_{\mathbb{S}} - \boldsymbol{\mu}_{\mathbb{S}}), \tag{3}$$

where $\rho > 0$ is a small ridge parameter. Here, the covariance structure uses deviations of the selected feature activations from their mean to estimate all latent features, yielding a full K-dimensional reconstruction $\boldsymbol{z}_{\mathbb{S}}^{\text{rec}}$. For $\rho = 0$, Eq. 3 reduces to the conditional mean $\mathbb{E}[\boldsymbol{z} \mid \boldsymbol{z}_{\mathbb{S}}]$ if the latent variables are jointly Gaussian, otherwise it is the minimum-mean-square linear estimator for a fixed support. In case of $\rho > 0$, Eq. 3 is the optimal ridge-regularized linear estimator. A derivation is provided in Appendix F.

The support quality is measured after decoding back to the original feature space. Let $\hat{\boldsymbol{y}}_{\mathbb{S}} = D(\boldsymbol{z}_{\mathbb{S}}^{\text{rec}})$, and let $\bar{\boldsymbol{y}}_{\text{train}}$ denote the mean representation over the training set. We use the feature-space reconstruction measure

$$R^2(\mathbb{S}) = 1 - \frac{\|\boldsymbol{y} - \hat{\boldsymbol{y}}_{\mathbb{S}}\|_2^2}{\|\boldsymbol{y} - \bar{\boldsymbol{y}}_{\text{train}}\|_2^2 + \epsilon}. \tag{4}$$

TTS seeks a compact support that reaches the target fidelity α. It then builds $\mathbb{S}_\alpha$ greedily. Starting from $\mathbb{S}_0 = \varnothing$, each candidate feature $j \notin \mathbb{S}_t$ is evaluated by its improvement of the feature-space fidelity when adding it to the current support,

$$g_j^{(t)} = R^2(\mathbb{S}_t \cup \{j\}) - R^2(\mathbb{S}_t). \tag{5}$$

Input-aware sparsity induction. Among positively activated coordinates, we bias selection toward features that are more strongly expressed in the current input:

$$\text{score}_j^{(t)} = g_j^{(t)} - \lambda_{\text{sparse}} \mathbf{1}_{z_j > 0} (1 - z_j)^2. \tag{6}$$

The highest-scoring feature is added to the support, and the procedure continues until

$$R^2(\mathbb{S}_\alpha) \geq \alpha. \tag{7}$$

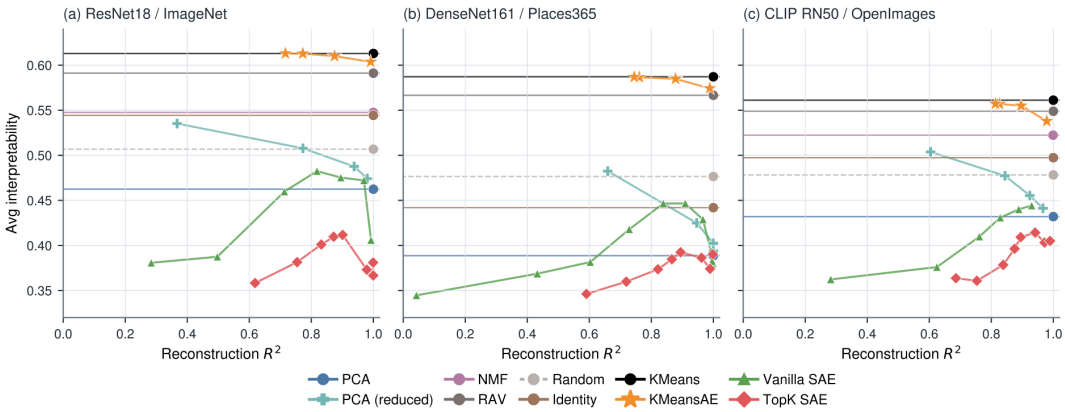

Figure 3: **KMeansAE provides a markedly stronger reconstruction–interpretability trade-off than sparse autoencoders.** Validation reconstruction R^2 versus average interpretability, defined as the mean of Clarity, INVERT, and CLIP-Dissect, for three model–dataset pairs. KMeansAE remains close to the KMeans semantic reference while achieving near-perfect reconstruction, and consistently outperforms Vanilla and TopK SAEs in the high-reconstruction regime.

We distinguish the greedy search support from the final active support after refitting; throughout, explanation sparsity refers to the latter, while total search support is reported in Appendix H.2. The final explanation consists of the selected support $\mathbb{S}_\alpha$ together with the corresponding feature activations $\boldsymbol{z}_{\mathbb{S}_\alpha}$. Varying α changes explanation sparsity without retraining the feature representation. The full TTS algorithm is provided in Appendix D and the computational complexity in Appendix E.

Illustrative explanation. Figure 2 shows an end-to-end example of the resulting explanation. For a query image containing an elephant in a bedroom, TTS reaches the target reconstruction fidelity using only two selected concepts, corresponding to the scene context and the elephant itself. The same concepts can also be located within the broader organization of the learned concept space, illustrating how the sparse input-specific explanation is related to the global concept structure of the learned dense representation.

4 Evaluation

We evaluate TTS on three model–dataset pairs spanning object recognition, scene recognition, and multimodal vision: ResNet18/ImageNet, DenseNet161/Places365, and CLIP RN50/OpenImages. We ask whether (i) dense features remain semantically interpretable, (ii) sparse explanations can be constructed at test time, and (iii) the learned concepts support targeted interventions.

We fit all extractors on frozen training representations and evaluate reconstruction and semantic quality on held-out validation data, mean-pooling spatial features. We compare PCA, Random, Identity, NMF, RAV, and KMeans with KMeansAE, JumpReLU, Archetypal SAE, TopK SAE, and Vanilla SAE; SAE hyperparameters are tuned to approximately match reconstruction ($R^2 \approx 0.8$). Semantic quality is summarized as Avg, the mean of Clarity, INVERT, and CLIP-Dissect. MS is reported separately because it provides little discrimination among dense linear extractors in our experiments; see Appendix G.4. Full details and hyperparameter sweeps are in Appendix G.

4.1 Reconstruction and Interpretability

A useful concept representation should preserve the source representation while exposing semantically coherent directions. We therefore evaluate each learned extractor across its natural control parameter and measure the trade-off between reconstruction fidelity R^2 and interpretability, measured as the average of Clarity, INVERT, and CLIP-Dissect. KMeansAE varies the geometry-alignment strength λ_{align}, Vanilla SAE varies its L_1 penalty, TopK SAE varies k, and reduced PCA varies the retained dimensionality. Fixed extractors are shown as horizontal references. All learned methods use the same cached activations, validation data, and training budget, with dictionary size $K = 5{,}000$.

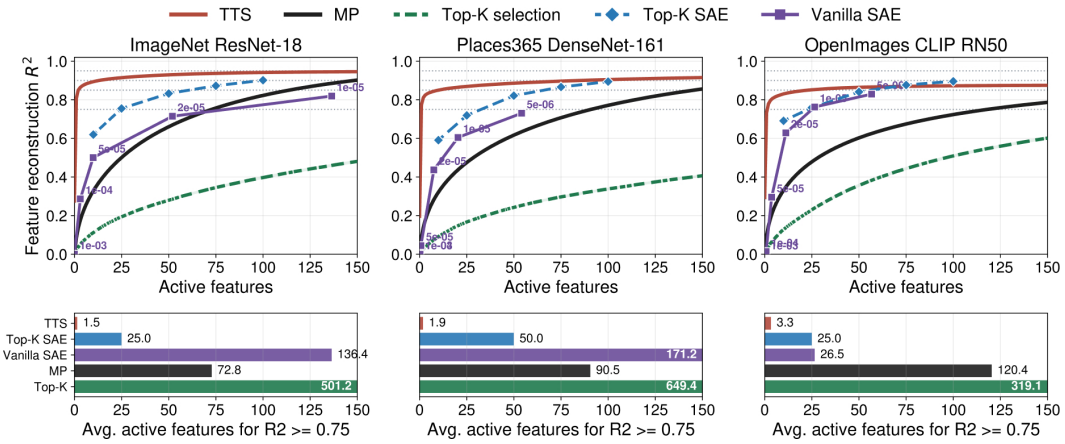

Figure 4: **Sparse reconstruction at test time.** Reconstruction R^2 versus positive active support for TTS, matching pursuit (MP), activation Top-K, TopK SAE, and Vanilla SAE. At $R^2 \geq 0.75$, TTS retains only 1.5, 1.9, and 3.3 positively weighted active concepts on ResNet18, DenseNet161, and CLIP RN50, respectively, compared with 72.8, 90.5, and 120.4 for MP and at least 25 active SAE features. Activation Top-K requires hundreds of features. Total greedy support (zero-activated selected features) and details are reported in Appendix H.2.

Figure 3 shows a consistent separation between geometry-aligned and conventional sparse representations. Across all three model–dataset pairs, KMeansAE traces the strongest reconstruction–interpretability trade-off among the learned reconstructive methods, remaining close to the KMeans reference even at high reconstruction fidelity. Vanilla and TopK SAEs achieve comparable or higher reconstruction only at substantially lower semantic scores, while reducing the PCA dimensionality does not recover the same high-reconstruction semantic quality. The metric-wise comparison, including the separate MS diagnostic, is reported in Appendix G.4.

We additionally test sensitivity to dictionary size on CLIP RN50/OpenImages. Larger dictionaries improve reconstruction but tend to reduce interpretability, while stronger geometry alignment consistently preserves higher semantic quality. This indicates that KMeansAE's advantage is not tied to a particular dictionary size and that geometry alignment is the key factor maintaining interpretable structure. Full sweeps and metric-wise results are provided in Appendix G.7.

4.2 Sparse Explanations Can Be Deferred to Test Time

We next test whether a dense feature space can yield compact explanations. We compare TTS with Matching Pursuit (MP) and activation Top-K selection over the same KMeansAE representation, and TopK and Vanilla SAEs spanning different training-time sparsity levels. For TTS, MP, and activation Top-K, we record feature-space reconstruction as support grows; each SAE point corresponds to a separately trained checkpoint. Our MP baseline also controls for the inference-time sparse-coding mechanism used by ITDA (Leask et al., 2025); we keep the dictionary fixed across MP and TTS to isolate the effect of the test-time reconstruction procedure rather than conflating it with a different dictionary-construction method.

Figure 4 shows that TTS reaches a given reconstruction fidelity with substantially fewer positively weighted active concepts than MP or the evaluated SAEs; activation Top-K requires hundreds of features to reach the same fidelity. This indicates that activation magnitude alone is insufficient for identifying reconstruction-relevant concepts, while covariance-aware reconstruction can exploit dependencies in the dense representation. Unlike a trained sparse code, TTS also allows the positive active support to adapt per input and per requested fidelity without retraining. For completeness, Appendix H.2 reports both positive active support and the total number of coordinates selected during the iterative search.

Cost of adaptivity. TTS deliberately shifts part of the sparsification effort to test time, so its improved adaptivity and compact explanations come with additional inference-time computation. In our CLIP RN50 benchmark, TTS is substantially slower than static encoders and MP. We therefore

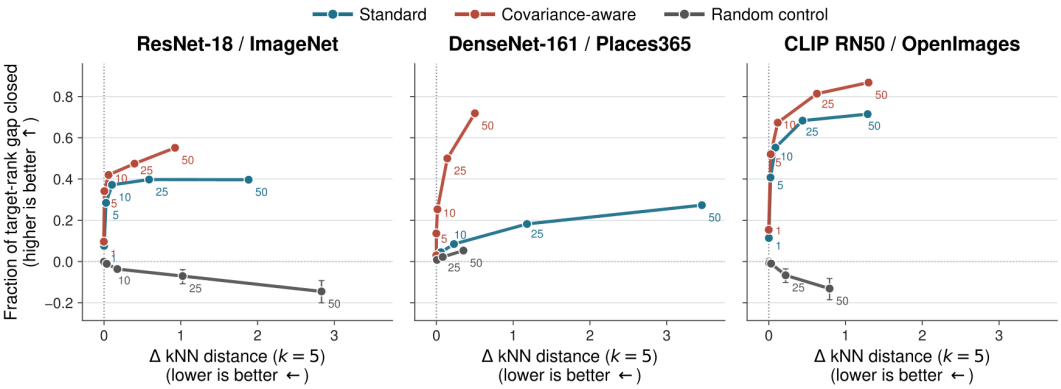

Figure 5: **Covariance-aware steering consistently outperforms standard steering.** Across all three model–dataset pairs, covariance-aware interventions close substantially more of the target-rank gap while remaining closer to real training representations, measured by ΔkNN distance ($k = 5$). All methods are matched in decoded feature-space perturbation norm; random controls produce little or negative target-directed change. Point labels denote steering strength and error bars show 95% bootstrap confidence intervals.

view this cost as the expected price of constructing input- and fidelity-specific explanations on demand, rather than fixing sparsity during training. Detailed throughput measurements and sensitivity to the fidelity and sparsity parameters are reported in Appendix H.3.

4.3 Dense Concepts Are Functionally Actionable

Semantic coherence and reconstruction alone do not reveal if learned concepts are functionally coupled to model behavior. We therefore intervene directly on TTS-AE coordinates and measure whether the downstream output moves toward the corresponding semantic target. Let $z = E(y)$ be the normalized dense representation, j the latent coordinate associated with a target concept, and $a^\star$ a high reference activation from the training set. At strength γ, the requested target-coordinate change is $\delta_j = \gamma(a^\star - z_j)$. We consider two steering directions, both expressed as an additive latent intervention $z^{(m)} = \Pi_+\big(z + \Delta z^{(m)}\big)$:

- **Standard steering.** $\Delta z^{\mathrm{std}} = \delta_j e_j$. This changes only the latent coordinate associated with the target concept, leaving all other coordinates unchanged.
- **Covariance-aware steering.** $\Delta z^{\mathrm{cov}} = \beta \delta_j d_j^{\mathrm{cov}}$, where $d_j^{\mathrm{cov}} = \Sigma_{:,j}/(\Sigma_{jj} + \rho)$. This changes the target concept together with latents that covary with it in the reference distribution. The scale β is chosen per example so that the decoded feature-space perturbation matches the norm of standard steering.

For either method, the residual $\Delta y = D(z^{\mathrm{steer}}) - D(z)$ is added to the original model representation. We also evaluate a random latent direction with zero target-coordinate component, matched to the same decoded perturbation norm. Full details are given in Appendix I.

To compare efficacy across models, we measure the fraction of the available target-rank gap that is closed,

$$G_{\mathrm{rank}} = \frac{r_{\mathrm{before}} - r_{\mathrm{after}}}{r_{\mathrm{before}} - 1}. \tag{8}$$

Thus, $G_{\mathrm{rank}} = 0$ indicates no improvement, $G_{\mathrm{rank}} = 1$ means the target reaches rank one, and negative values indicate movement away from the target.

Figure 5 shows that both steering methods move the intended target upward in the output ranking, whereas matched random directions have little systematic target-specific effect. Covariance-aware steering is consistently more favorable, closing a larger fraction of the target-rank gap while producing a smaller increase in distance to nearby real training representations. Importantly, this kNN diagnostic does not use the covariance employed to construct the intervention. Thus, the covariance structure

exploited by TTS provides a useful intervention direction that better preserves empirical representation support.

Open-ended VLM steering. We additionally test whether the same intervention principle extends to generative multimodal behavior. For Qwen2-VL-2B-Instruct, we mean-pool projected visual tokens, steer the corresponding KMeansAE latent representation, decode the residual, and add it back to every visual token before generation. Figure 6 shows a qualitative example for *car*: covariance-aware steering introduces the target concept into the generated description while largely preserving the surrounding scene content.

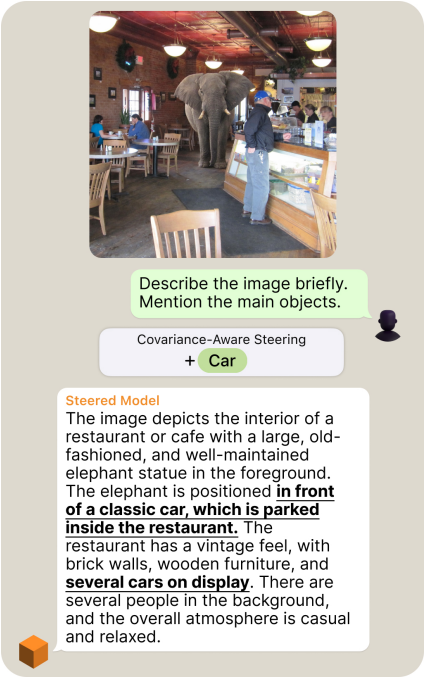

Figure 6: **Steering an open-ended VLM.** Covariance-aware injection of the concept *car* into Qwen2-VL-2B causes the generated description to incorporate the target concept while preserving the surrounding scene description. See Appendix I.3.

5 From Local to Global Explanations

TTS constructs sparse explanations locally: for a particular input, it selects the concepts needed to reconstruct that input's internal representation. Keeping the underlying concept representation dense also enables a complementary form of sparsity. Instead of fixing an input and asking which concepts explain its representation, we can fix a model target and ask which concepts consistently explain that target across a data distribution. Thus, the same learned concept space can support both *input-specific* and *target-specific* sparse explanations, without retraining the concept extractor.

Sparse global readout. Let $g_c(x)$ denote a scalar output of the frozen model, such as a class logit, and let $\widetilde{z}(x)$ denote the TTS-AE concept activations standardized on the fitting distribution. We approximate the original model output directly with a sparse linear readout

$$\widehat{g}_c(x) = b_c + \sum_{j \in S_c} w_{cj} \widetilde{z}_j(x), \tag{9}$$

where S_c contains a small target-specific subset of concepts. Unlike the local TTS support $S_\alpha(x)$, which changes with the input and is chosen to preserve feature reconstruction, S_c is shared across inputs and is chosen to predict one particular model output. Positive and negative coefficients identify concepts associated with increasing or decreasing that output conditional on the other selected concepts. This yields complementary sparse views of the same dense representation: local TTS asks which concepts explain a particular input, while the global readout identifies concepts that systematically predict a model output across images. Faithfulness is measured by held-out output-space R^2; full fitting details are given in Appendix J. Figure 7 illustrates this for the ResNet18 *tennis ball* logit: a four-concept readout achieves held-out $R^2 = 0.54$, selecting the target concept together with dog-related contextual concepts whose semantics are supported by their top-activating images.

Prospective validation with generated evidence. The coefficients of a global readout are predictive associations, so we additionally test whether the selected concepts remain behaviorally informative outside the fitting data. For 20 predeclared targets, we use up to three positively weighted concept descriptions as prompts to FLUX and score the resulting images with frozen ResNet18, CLIP RN50, and Qwen2-VL-2B evaluators. Target names, synonyms, and direct hypernyms are removed from the primary positive prompts, and we compare against matched random concepts, negatively weighted concepts, and a direct target-name positive control.

Selected positive concepts raise the standardized target response relative to matched random and negative controls, with substantially stronger effects when the top concepts are generated jointly. This

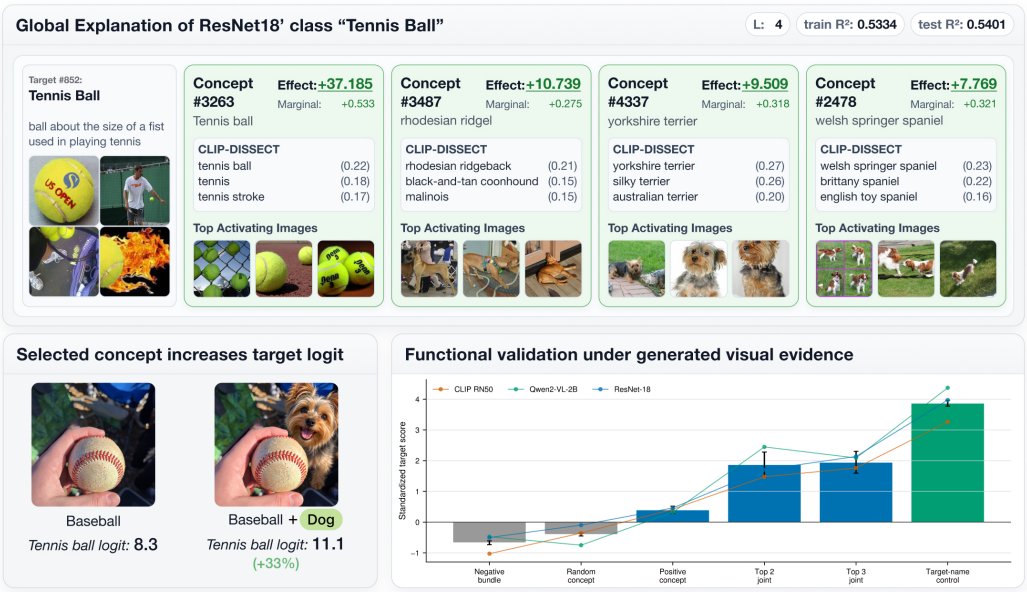

Figure 7: **Dense concepts support sparse global explanations and behavioral validation. Top:** a four-concept readout of the ResNet18 *tennis ball* logit achieves held-out $R^2 = 0.54$, selecting the target concept together with dog-related contextual predictors. **Bottom-left:** adding a selected dog concept to a baseball image increases the *tennis ball* logit from 8.3 to 11.1. **Bottom-right:** across 20 targets and three frozen evaluators, images generated from positively weighted concepts receive higher standardized target scores than matched random or negative controls, with stronger responses when selected concepts are combined. The target-name condition serves as a positive control. Full protocol and controls are provided in Appendix J.

provides prospective behavioral support for the sparse global readouts, although generated images may also alter correlated visual attributes. Together, the local, steering, and global analyses show how the same dense representation can support different sparse explanations depending on the question. Full generation protocol, control construction, scoring, aggregation, and limitations are provided in Appendix J.

6 Conclusion

In this work, we introduced TTS-AE to decouple feature learning from explanation sparsity, making sparsity a property of the explanation rather than a constraint on the representation. Our evaluation demonstrates the benefits of this separation in three ways. First, dense concept representations learned without sparsity constraints achieve stronger semantic interpretability than the evaluated nonlinear sparse autoencoders while maintaining high reconstruction fidelity. Second, TTS constructs compact, input-specific explanations at test time, using substantially fewer positively weighted concepts than sparse autoencoders and sparse reconstruction baselines at matched fidelity. Third, the resulting concepts support targeted interventions, with covariance-aware steering producing stronger target-directed changes while remaining closer to observed representations than coordinate-wise steering. Beyond interventions, the same dense concept space can be used to construct sparse, target-specific global explanations without retraining the concept representation. Overall, our results indicate that sparse explanations do not require sparse representations: sparsity can instead be introduced on user demand and adapted to the input, target, and desired fidelity.

AI USE STATEMENT

Generative AI tools were used as assistive tools during the preparation of this work. Specifically, they were used to support coding and debugging, assist with language editing of author-written text, assist with literature and related-work searches, and provide limited assistance in reviewing mathematical derivations and proofs. Mathematical arguments, claims, and proofs were independently verified and finalized by the authors.

Generative AI was not used to generate the research idea, formulate the central hypotheses, design the proposed method, or determine the experimental conclusions. The scientific contributions, methodological decisions, experiments, analysis, and interpretation of results are the work of the authors. All AI-assisted outputs were reviewed, edited, and validated by the authors, who take full responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies methods for interpreting and intervening on the internal representations of pretrained neural networks. We use existing models and benchmark datasets and do not collect personal data or conduct experiments on human subjects. While our primary goal is to improve model interpretability, the demonstrated steering mechanisms can also modify model behavior and could potentially be used in unintended ways. We therefore present these interventions primarily as tools for analyzing and validating learned concepts rather than as mechanisms for unconstrained model control. The generative models used in our experiments are employed only to construct controlled evaluation evidence.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of TTS-AE and Test-Time Sparsification in the main paper, including the reconstruction objective, covariance-based reconstruction, and feature-selection procedure. The appendices provide additional algorithmic details, derivations, computational complexity, model and dataset configurations, hyperparameters, baseline specifications, evaluation protocols, steering experiments, and the global-explanation and generated-evidence protocols. We will provide the implementation and experiment configurations as supplementary material to facilitate reproduction of the reported results.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, volume 31, pp. 9505–9515, 2018.

Sebastian Bach, Alexander Binder, Grégoire on, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research*, 11:1803–1831, 2010.

Dilyara Bareeva, Galip Ümit Yolcu, Anna Hedström, Niklas Schmolenski, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. Quanda: An interpretability toolkit for training data attribution evaluation and beyond. *arXiv preprint arXiv:2410.07158*, 2024.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6541–6549, 2017.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Jack Merullo, et al. Do sparse autoencoders capture concept manifolds? *arXiv preprint arXiv:2604.28119*, 2026.

Umang Bhatt, Adrian Weller, and José M. F. Moura. Evaluating and aggregating feature-based model explanations, 2020.

Alexander Binder, Leander Weber, Sebastian Lapuschkin, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Shortcomings of top-down randomization-based sanity checks for evaluations of deep neural network explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16143–16152, 2023.

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. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Kirill Bykov, Mayukh Deb, Dennis Grinwald, Klaus Robert Muller, and Marina MC Höhne. DORA: Exploring outlier representations in deep neural networks. *Transactions on Machine Learning Research*, 2023a. ISSN 2835-8856. URL https://openreview.net/forum?id=nfYwRIezvg.

Kirill Bykov, Laura Kopf, Shinichi Nakajima, Marius Kloft, and Marina Höhne. Labeling neural representations with inverse recognition. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 24804–24828. Curran Associates, Inc., 2023b.

Uri Cohen, SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature communications*, 11(1):746, 2020.

Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba E. Ba. From flat to hierarchical: Extracting sparse representations with matching pursuit. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Ll5miDx8KB.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

Ann-Kathrin Dombrowski, Jan E Gerken, Klaus-Robert Müller, and Pan Kessel. Diffeomorphic counterfactuals with generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3257–3274, 2023.

Maximilian Dreyer, Erblina Purelku, Johanna Vielhaben, Wojciech Samek, and Sebastian Lapuschkin. Pure: Turning polysemantic neurons into pure features by identifying relevant circuits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8212–8217, 2024.

Eamon Duede. Deep learning opacity in scientific discovery. *Philosophy of Science*, pp. 1–13, 2022.

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.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 01 2009.

Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9v1eW8HgMU.

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 *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403, 2024.

Tessa Han, Suraj Srinivas, and Himabindu Lakkaraju. Which explanation should i choose? a function approximation perspective to characterizing post hoc explanations. *Advances in Neural Information Processing Systems*, 35:5256–5268, 2022.

Anna Hedström, Philine Bommer, Kristoffer K Wickstrøm, Wojciech Samek, Sebastian Lapuschkin, and Marina M-C Höhne. The meta-evaluation problem in explainable ai: identifying reliable estimators with metaquantus. *arXiv preprint arXiv:2302.07265*, 2023.

Anna Hedström, Leander Weber, Sebastian Lapuschkin, and Marina Höhne. Sanity checks revisited: An exploration to repair the model parameter randomisation test. *arXiv preprint arXiv:2401.06465*, 2024.

Sai Sumedh R Hindupur, Ekdeep S Lubana, Thomas Fel, and Demba Ba. Projecting assumptions: The duality between sparse autoencoders and concept geometry. *Advances in Neural Information Processing Systems*, 38:11649–11699, 2026.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International Conference on Machine Learning*, pp. 2668–2677. PMLR, 2018.

Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

Laura Kopf, Philine L Bommer, Anna Hedström, Sebastian Lapuschkin, Marina Höhne, and Kirill Bykov. Cosy: Evaluating textual explanations of neurons. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 34656–34685. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/3d4c0a618d0acd7921493e4f30395c22-Paper-Conference.pdf.

Satyapriya Krishna, Tessa Han, Alex Gu, Steven Wu, Shahin Jabbari, and Himabindu Lakkaraju. The disagreement problem in explainable machine learning: A practitioner's perspective. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=jESY2WTZCe.

Patrick Leask, Neel Nanda, and Noura Al Moubayed. Inference-time decomposition of activations (ITDA): A scalable approach to interpreting large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=4WMZqAoYi4.

Victor Lecomte, Kushal Thaman, Rylan Schaeffer, Naomi Bashkansky, Trevor Chow, and Sanmi Koyejo. What causes polysemanticity? an alternative origin story of mixed selectivity from incidental causes. *arXiv preprint arXiv:2312.03096*, 2023.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4765–4774, 2017.

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

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3:861, 09 2018. doi: 10.21105/joss.00861.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Deepdream-a code example for visualizing neural networks. *Google Research*, 2(5):67, 2015.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, 33:17153–17163, 2020.

Neel Nanda, Lawrence Chan, Tom Liberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

Angus Nicolson, Lisa Schut, Alison Noble, and Yarin Gal. Explaining explainability: Recommendations for effective use of concept activation vectors. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=7CUluLpLxV.

Tuomas Oikarinen and Tsui-Wei Weng. Clip-dissect: Automatic description of neuron representations in deep vision networks. *arXiv preprint arXiv:2204.10965*, 2022.

Tuomas Oikarinen, Subhro Das, Lam Nguyen, and Lily Weng. Label-free concept bottleneck models. In *International Conference on Learning Representations*, 2023.

Tuomas Oikarinen, Ge Yan, and Tsui-Wei Weng. Evaluating neuron explanations: A unified framework with sanity checks. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=8unyWZ14mf.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020. doi: 10.23915/distill.00024.001. URL https://distill.pub/2020/circuits/zoom-in.

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 preprint arXiv:2407.14435*, 2024.

Wojciech Samek, Alexander Binder, Grégoire on, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2016.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2017.

Anav Sood and Trevor Hastie. A statistical view of column subset selection. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 87(5):1382–1403, 2025.

Xiaoqing Sun, Alessandro Stolfo, Joshua Engels, Ben Wu, Senthooran Rajamanoharan, Mrinmaya Sachan, and Max Tegmark. Dense sae latents are features, not bugs. *Advances in Neural Information Processing Systems*, 38:140252–140282, 2026.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=nA5AZ8CEyow`.

Carlos Zednik. Solving the black box problem: A normative framework for explainable artificial intelligence. *Philosophy & technology*, 34(2):265–288, 2021.

APPENDIX

A RELATED WORKS (EXTENDED)

A.1 LOCAL AND GLOBAL SURROGATE EXPLANATIONS

Local explainability methods aim to explain the behavior of a model for a particular input (Guidotti et al., 2018). Common families include counterfactual explanations (Wachter et al., 2017; Dombrowski et al., 2023), example-based explanations such as influence functions and training-data attribution (Koh & Liang, 2017; Hammoudeh & Lowd, 2024; Bareeva et al., 2024), and feature-attribution methods that assign relevance to input dimensions (Baehrens et al., 2010; Bach et al., 2015; Kindermans et al., 2017; Montavon et al., 2018; Lundberg & Lee, 2017). In vision, such attributions are commonly represented as pixel- or region-level heatmaps, which provide spatial localization but do not by themselves supply concept-level semantics.

A useful unifying view is *Local Function Approximation* (LFA) (Han et al., 2022). LFA characterizes many post-hoc explanation methods as fitting a simple surrogate in a neighborhood of the query input. Different choices of neighborhood distribution, fidelity objective, and surrogate family can lead to substantially different explanations, contributing to the disagreement observed between attribution methods (Krishna et al., 2024). Other attribution procedures, including LRP (Bach et al., 2015), Grad-CAM (Selvaraju et al., 2017), and DeepLIFT (Shrikumar et al., 2017), instead use specialized propagation or architectural rules.

Global explanations seek an interpretable approximation of model behavior across a data distribution rather than for one individual input. Such explanations include global surrogate models, concept bottlenecks, and methods that aggregate local explanations into compact global descriptions (Guidotti et al., 2018; Koh et al., 2020; Yuksekgonul et al., 2023). Sparse linear or rule-based surrogates are particularly common because they provide explicit control over explanation complexity while allowing fidelity to the original model to be measured on held-out data.

A.2 CONCEPT EXTRACTION AND REPRESENTATION GEOMETRY

Mechanistic Interpretability studies internal representations and computations with the aim of identifying meaningful units, directions, circuits, or other structures inside neural networks (Elhage et al., 2022; Lecomte et al., 2023; Bereska & Gavves, 2024; Nanda et al., 2023). A central difficulty is polysemanticity, where an individual unit or direction responds to multiple factors (Scherlis et al., 2022; Olah et al., 2020). This has motivated methods that transform model activations into alternative coordinate systems whose components are intended to be easier to interpret.

Supervised concept methods define semantic variables using labeled examples or predefined concept sets. Concept Activation Vectors represent concepts as separating directions in representation space (Kim et al., 2018; Nicolson et al., 2025), while Concept Bottleneck Models expose an explicit concept layer that can be inspected or intervened upon (Koh et al., 2020). Label-free and post-hoc variants use pretrained representations or surrogate models to construct concept bottlenecks without retraining the original model from scratch (Oikarinen et al., 2023; Yuksekgonul et al., 2023).

Unsupervised approaches instead derive structure directly from model activations. Matrix factorization methods such as NMF seek additive non-negative components (Lee & Seung, 1999), while CRAFT uses recursive factorization to extract concepts from intermediate representations (Fel et al., 2023). Clustering methods identify recurring regions or prototypical directions in activation space (Dreyer et al., 2024). Archetypal SAE similarly constrains learned dictionary elements through their relationship to observed activations, explicitly connecting dictionary learning to the empirical representation distribution (Fel et al., 2025).

The geometric organization of neural representations provides additional motivation for such approaches. Deep networks can progressively reshape object manifolds so that task-relevant structure becomes more linearly separable, with changes in manifold dimension, extent, and correlations across layers (Cohen et al., 2020). Recent work further argues that individual semantic concepts may themselves be represented as low-dimensional manifolds rather than isolated directions. Bhalla et al. (Bhalla et al., 2026) show that SAEs can represent such structures either through groups of atoms spanning a manifold globally or through localized atoms that tile different regions of it.

A.3 Sparse Autoencoders and Concept Geometry

Sparse Autoencoders learn an overcomplete dictionary while constraining each input to activate only a small subset of latent variables (Ng et al., 2011; Cunningham et al., 2023; Bricken et al., 2023). Classical formulations use an ℓ_1 penalty on activations, creating a continuous trade-off between reconstruction and sparsity. More recent variants enforce sparsity more directly. Top-K SAEs retain a fixed number of the largest activations for each input (Gao et al., 2025), while JumpReLU SAEs learn activation thresholds and directly optimize an L_0-style sparsity objective (Rajamanoharan et al., 2024).

The learned features depend not only on the data but also on structural assumptions imposed by the SAE architecture. Hindupur et al. (Hindupur et al., 2026) formalize this dependence and show that different SAE formulations can recover different concepts because their architectures encode assumptions about properties such as intrinsic dimensionality and separability. Their analysis motivates geometry-aware SAE designs rather than treating different sparse autoencoders as interchangeable estimators of a single underlying dictionary.

Sparsity also does not imply that every meaningful SAE feature activates rarely. Sun et al. (Sun et al., 2026) study densely activating SAE latents in language models and show that many correspond to systematic and functionally meaningful directions, including structural, semantic, and output-related signals. Their results indicate that dense latents can reflect genuine properties of the underlying representation rather than merely optimization artifacts.

Concept structure can also be distributed across several SAE atoms. Bhalla et al. (Bhalla et al., 2026) analyze continuous concept manifolds and show that existing SAEs can split such structure across multiple atoms, sometimes producing fragmented mixtures of global subspace and local tiling solutions. This motivates evaluating semantic structure beyond one-latent-to-one-concept correspondence.

A.4 Sparse Approximation at Inference Time

Sparse approximation can be performed after a dictionary has been fixed. Matching Pursuit greedily constructs a sparse approximation by repeatedly selecting the dictionary atom most correlated with the current reconstruction residual (Mallat & Zhang, 1993). The resulting support can be viewed as a sequence of increasingly accurate approximations, with sparsity controlled through the number of pursuit steps or a reconstruction-based stopping condition.

Recent interpretability methods incorporate this principle into neural representation decomposition. MP-SAE uses a Matching-Pursuit-style encoder over an SAE-learned dictionary, selecting features sequentially according to their contribution to reconstruction (Costa et al., 2025). Because the support is constructed iteratively, different prefix lengths provide different sparsity levels at inference time while retaining a dictionary learned within the SAE framework.

Inference-Time Decomposition of Activations (ITDA) removes SAE training altogether (Leask et al., 2025). ITDA constructs its dictionary directly from observed model activations. During dictionary construction, a new activation is added when its Matching Pursuit reconstruction under the existing dictionary exceeds a specified error threshold. At inference time, a new activation is decomposed over this activation dictionary using Matching Pursuit with a pre-specified sparsity level L_0. Each dictionary element remains associated with the input context that generated the corresponding model activation, providing an exemplar-based interpretation of the resulting decomposition.

Top-K selection provides a simpler inference-time alternative when a latent representation has already been obtained. Coordinates are ranked by activation magnitude, but the largest K are retained. Unlike residual-based pursuit, activation ranking does not directly optimize reconstruction error and does not account for dependencies among latent coordinates.

A.5 Feature Description and Semantic Validation

Extracted features must be connected to recognizable semantic patterns before they can serve as human-readable explanations. Feature Visualization and activation maximization synthesize or retrieve inputs that strongly activate a unit or direction (Erhan et al., 2009; Mordvintsev et al., 2015;

Olah et al., 2017). Network Dissection associates internal units with labeled visual concepts (Bau et al., 2017), while compositional and language-based methods assign textual descriptions to neural representations (Mu & Andreas, 2020; Oikarinen & Weng, 2022; Bykov et al., 2023b).

CLIP-Dissect uses vision-language similarity to assign open-vocabulary textual descriptions to visual neurons (Oikarinen & Weng, 2022), while INVERT derives labels through inverse recognition of activation patterns (Bykov et al., 2023b). Vision-language models further enable descriptions to be generated directly from collections of highly activating examples, making it possible to assign compact natural-language labels without requiring a fixed vocabulary.

A plausible textual label does not necessarily provide a faithful description of feature behavior. CoSy evaluates textual neuron explanations by comparing feature responses on examples synthesized from a proposed description with responses on controls (Kopf et al., 2024). Oikarinen et al. (Oikarinen et al., 2025) provide a unified framework for evaluating neuron explanations and show through sanity checks that different evaluation metrics can lead to substantially different conclusions about explanation quality.

More generally, explanation evaluation often relies on proxy measures because ground-truth explanations are rarely available. Faithfulness tests quantify changes in model behavior after modifying features judged important (Samek et al., 2016; Bhatt et al., 2020); robustness tests measure stability to small perturbations (Montavon et al., 2018); and randomization tests assess dependence on learned model parameters (Adebayo et al., 2018). Meta-evaluation studies show that the reliability of explanation metrics must itself be tested (Binder et al., 2023; Hedström et al., 2023; 2024).

A.6 Subset Selection and Covariance-Based Reconstruction

Selecting a small set of observed variables that preserves information about a larger representation has a long history in statistics and numerical linear algebra. Column Subset Selection (CSS) chooses a subset of columns that provides a good linear reconstruction of the complete data matrix, while the related statistical literature studies *principal variables*, subsets of observed variables that preserve information about the remaining coordinates.

Sood and Hastie (Sood & Hastie, 2025) show that CSS and principal-variable selection can be expressed through the same population objective and characterized using only covariance information. For a fixed subset S, the optimal linear prediction of the centered full vector from its selected coordinates takes the form

$$\hat{\boldsymbol{z}} = \boldsymbol{\Sigma}_{:,S}\boldsymbol{\Sigma}_{S,S}^{-1}\boldsymbol{z}_S, \tag{10}$$

when the relevant covariance block is nonsingular. Their analysis also gives CSS a probabilistic interpretation in which unselected variables are generated conditionally from the selected variables through a linear model.

For jointly Gaussian variables, the same expression arises from standard conditional Gaussian inference:

$$\mathbb{E}[\boldsymbol{z} \mid \boldsymbol{z}_S] = \boldsymbol{\mu} + \boldsymbol{\Sigma}_{:,S}\boldsymbol{\Sigma}_{S,S}^{-1}(\boldsymbol{z}_S - \boldsymbol{\mu}_S). \tag{11}$$

Thus, subset selection can be interpreted not only as sparse approximation, but also as choosing which coordinates must be observed so that the remaining coordinates can be inferred from statistical dependencies. Regularized variants replace $\boldsymbol{\Sigma}_{S,S}^{-1}$ with $(\boldsymbol{\Sigma}_{S,S} + \rho\boldsymbol{I})^{-1}$ when the selected covariance block is ill-conditioned.

B VLMGrid Concept Descriptor

The VLMGrid descriptor assigns a concise natural-language name to each learned concept by presenting a vision-language model with the images that most strongly activate that concept. The resulting phrase provides a human-readable semantic handle for a latent coordinate and is used both for concept inspection and for downstream atlas labeling.

Let $\mathcal{D} = \{x_n\}_{n=1}^{N}$ be the image collection used for concept description, and let

$$\boldsymbol{z}_n = E(f(x_n)) \in \mathbb{R}^K$$

denote the dense latent representation of image x_n. We write $a_{nj} = z_{n,j}$ for the activation of concept j on image x_n.

B.1 Top-Activating Image Selection

For a positive concept description, VLMGrid selects the m images with the largest activation values for concept j:

$$\mathcal{I}_j^{(m)} = \mathrm{TopM}_{n\in\{1,\dots,N\}}\, a_{nj}. \tag{12}$$

In the experiments reported here, we use a 3×3 grid and therefore set $m = 9$.

The selected images are intended to expose the dominant visual pattern associated with the concept. Rather than describing one individual exemplar, the descriptor asks the vision-language model to identify the common visual regularity shared across the activation-maximizing examples.

B.2 Grid Construction And VLM Prompting

The selected images are converted into a single image grid. Each cell contains one activation-maximizing image, resized to a common visual scale so that the model can compare the examples side by side. The grid is then passed to a vision-language model together with a restrictive prompt that asks for a single short visual concept phrase.

Formally, the descriptor is

$$\tau_j = \mathrm{VLMGrid}\left(\{x_n : n \in \mathcal{I}_j^{(m)}\}\right), \tag{13}$$

where τ_j is a concise human-readable phrase describing the common visual pattern in the grid. Depending on the concept, such phrases may refer to objects, scenes, textures, materials, colors, or part-level visual motifs.

The prompting scheme is intentionally restrictive. The model is instructed to return only one short concept phrase, use no more than five words, and avoid explanations, lists, or additional commentary. This encourages labels that are compact enough to be displayed directly in concept tables, visualizations, and atlas summaries.

B.3 Output Normalization

The generated phrase is normalized before being stored. We retain only the first response line, remove common answer prefixes, discard punctuation-heavy or malformed outputs, and truncate phrases that exceed the word budget. If the model does not return a usable phrase, the concept receives a conservative fallback label.

This post-processing step ensures that VLMGrid outputs remain uniform in format and suitable for downstream aggregation. In particular, the resulting labels are later used as linguistic evidence when generating cluster names in the concept atlas.

B.4 Implementation Details

In the experiments reported here, the default descriptor uses Qwen/Qwen3-VL-8B-Instruct as the vision-language model.

The VLMGrid descriptor therefore converts an abstract activation coordinate into a compact semantic description by combining activation-maximizing visual evidence with controlled vision-language summarization.

C Concept Atlas Construction

We use a *concept atlas* to visualize the global organization of the latent space learned by TTS-AE. Each point in the atlas corresponds to one learned concept. Concepts that behave similarly across the dataset are placed nearby, and groups of neighboring concepts are assigned concise semantic labels. The atlas construction has three main steps: pairwise concept comparison through Extreme-Activation distance, two-dimensional visualization with UMAP, and semantic clustering with LLM-generated region labels.

C.1 Extreme-Activation Distance Between Concepts

Let $\mathcal{D} = \{x_n\}_{n=1}^N$ be the image collection used to build the atlas, and let

$$\boldsymbol{z}_n = E(f(x_n)) \in \mathbb{R}^K$$

denote the dense latent representation of image x_n. We write $a_{nj} = z_{n,j}$ for the activation of concept j on image x_n.

To compare concepts, we use the Extreme-Activation distance introduced by (Bykov et al., 2023a). The central idea is that two concepts should be considered similar when images that strongly activate one concept also induce similar activation behavior in the other. Thus, concept similarity is defined through their responses on activation-maximizing examples rather than through their coordinate values alone.

For each concept j, we identify a set of images with the strongest positive activations,

$$\mathcal{I}_j^{(m)} = \mathrm{TopM}_{n\in\{1,\dots,N\}}\, a_{nj}, \tag{14}$$

where m is the number of extreme-activation images used to characterize the concept. The Extreme-Activation distance compares concepts by evaluating their responses on such top-activating image sets, yielding a pairwise dissimilarity matrix

$$\boldsymbol{D}^{\mathrm{EA}} = \left[d_{ij}^{\mathrm{EA}}\right]_{i,j=1}^{K}. \tag{15}$$

This matrix serves as the quantitative backbone of the atlas. Concepts with small Extreme-Activation distance tend to respond to visually related image patterns and are therefore treated as neighbors in the subsequent visualization.

C.2 Two-Dimensional Projection With UMAP

We embed the concept-distance matrix $\boldsymbol{D}^{\mathrm{EA}}$ into two dimensions using UMAP (McInnes et al., 2018). Let

$$\boldsymbol{u}_j \in \mathbb{R}^2$$

denote the resulting atlas coordinate of concept j. UMAP is used purely as a visualization tool: its role is to preserve local neighborhoods implied by the Extreme-Activation distance as well as possible in a planar map.

The two axes of the atlas are not interpreted individually. Instead, the atlas is read through relative geometry: concepts placed close together are concepts with similar extreme-activation behavior, while broader spatial regions indicate larger semantic neighborhoods in the learned latent space. This projection provides a compact global view of the concept organization and makes it possible to inspect whether the learned representation contains coherent visual regions, such as object families, scene types, materials, or texture groups.

C.3 Hierarchical Clustering And Semantic Region Labels

To make the atlas easier to interpret, we organize the projected concepts into semantic regions. For several target resolutions

$$\mathcal{K} = \{8, 16, 32, 64, 128, 256\},$$

we perform agglomerative clustering with average linkage on the UMAP coordinates $\{\boldsymbol{u}_j\}_{j=1}^K$. For each resolution $k \in \mathcal{K}$, this produces a partition

$$\mathcal{P}_k = \{C_{k,1}, \dots, C_{k,k}\},$$

where each cluster $C_{k,r}$ is a set of concept indices. Coarse clusterings summarize broad semantic regions of the atlas, while finer clusterings reveal more specific local substructures.

To assign an interpretable label to each region, we reuse the concept-level descriptions produced by the VLMGrid descriptor described in Appendix B. Let τ_j denote the textual description of concept j. For a cluster $C_{k,r}$, we collect the concept names

$$T_{k,r} = \{\tau_j : j \in C_{k,r}\}, \tag{16}$$

and ask a language model to summarize them into a concise region label,

$$\ell_{k,r} = \mathrm{LLM}_{\mathrm{cluster}}(T_{k,r}). \tag{17}$$

The generated label is intended to describe the dominant shared visual theme of the cluster, such as a scene category, object family, color pattern, texture, or material property. These region labels make the atlas readable at multiple scales: low-resolution clusters provide a global overview of the latent space, while high-resolution clusters expose finer semantic distinctions.

The final concept atlas, therefore, combines three pieces of information in a single visualization: the Extreme-Activation geometry of the learned concepts, a two-dimensional neighborhood-preserving layout, and interpretable cluster labels derived from concept descriptions. It serves as a global complement to the local sparse explanations produced by the TTS module.

D Detailed Construction Of The TTS Module

This appendix provides the full specification of the Test-Time Sparsification (TTS) procedure used in the main paper. The TTS module consists of two components:

$$\tau_E^{\alpha} : (\boldsymbol{y}, \boldsymbol{z}) \mapsto (S_\alpha, \boldsymbol{z}_{S_\alpha}), \qquad \tau_D : (S, \boldsymbol{z}_S) \mapsto \boldsymbol{z}_S^{\mathrm{recon}}. \tag{18}$$

The TTS encoder τ_E^{α} selects a support of latent coordinates that reaches a prescribed feature-space fidelity threshold α, while the TTS decoder τ_D reconstructs a dense latent state from the selected coordinates using the empirical covariance structure of the latent space.

D.1 Reference Statistics In The Latent Space

Let $\{\boldsymbol{z}_n\}_{n=1}^N$ denote latent vectors obtained from a reference dataset, where $\boldsymbol{z}_n = E(\boldsymbol{y}_n)$. We compute the empirical latent mean and covariance,

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{z}_n \in \mathbb{R}^K, \qquad \boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N} (\boldsymbol{z}_n - \boldsymbol{\mu})(\boldsymbol{z}_n - \boldsymbol{\mu})^\top \in \mathbb{R}^{K\times K}. \tag{19}$$

These statistics are computed once and then kept fixed at inference time. For a query input with latent state $\boldsymbol{z} \in \mathbb{R}^K$, we define the centered representation $\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{\mu} \in \mathbb{R}^K$. For any support $S \subseteq \{1, \dots, K\}$, the selected centered activations are denoted by $\tilde{\boldsymbol{z}}_S \in \mathbb{R}^{|S|}$.

D.2 Construction Of The TTS Decoder

The TTS decoder τ_D reconstructs a dense latent state from a support S and the observed activations $\boldsymbol{z}_S$. Importantly, coordinates outside S are not interpreted as zero. They remain unobserved and are inferred from the selected activations through the empirical covariance structure.

For a nonempty support S, the centered reconstruction is

$$\hat{\tilde{\boldsymbol{z}}}_S = \boldsymbol{\Sigma}_{:,S}\left(\boldsymbol{\Sigma}_{S,S} + \rho \boldsymbol{I}_{|S|}\right)^{-1} \tilde{\boldsymbol{z}}_S \in \mathbb{R}^K, \qquad \boldsymbol{z}_S^{\mathrm{recon}} = \boldsymbol{\mu} + \hat{\tilde{\boldsymbol{z}}}_S \in \mathbb{R}^K, \tag{20}$$

where $\rho > 0$ is a ridge parameter used for numerical stability and $\boldsymbol{I}_{|S|} \in \mathbb{R}^{|S|\times|S|}$ is the identity matrix. Equivalently, the TTS decoder can be written as

$$\tau_D(S, \boldsymbol{z}_S) = \boldsymbol{\mu} + \boldsymbol{\Sigma}_{:,S}\left(\boldsymbol{\Sigma}_{S,S} + \rho \boldsymbol{I}_{|S|}\right)^{-1} (\boldsymbol{z}_S - \boldsymbol{\mu}_S). \tag{21}$$

For the empty support $S = \emptyset$, no coordinates are observed. In this case, the reconstruction defaults to the latent mean,

$$\tau_D(\emptyset, \emptyset) = \boldsymbol{\mu}, \qquad \boldsymbol{y}_0 = D(\boldsymbol{\mu}). \tag{22}$$

The optimality properties of the reconstruction rule in equation 20 are established in Appendix F.

D.3 Feature-Space Fidelity Measure

For a support S, let

$$\boldsymbol{z}_S^{\mathrm{recon}} = \tau_D(S, \boldsymbol{z}_S), \qquad \hat{\boldsymbol{y}}_S = D(\boldsymbol{z}_S^{\mathrm{recon}}) \in \mathbb{R}^M. \tag{23}$$

Let

$$\bar{\boldsymbol{y}}_{\mathrm{train}} = \frac{1}{N_{\mathrm{train}}} \sum_{n \in \mathcal{D}_{\mathrm{train}}} \boldsymbol{y}_n \tag{24}$$

denotes the mean representation over the complete training-feature set. We use the same feature-space reconstruction measure throughout the paper:

$$R^2(S) = 1 - \frac{\|\boldsymbol{y} - \hat{\boldsymbol{y}}_S\|_2^2}{\|\boldsymbol{y} - \bar{\boldsymbol{y}}_{\mathrm{train}}\|_2^2 + \varepsilon}. \tag{25}$$

The threshold $\alpha \in (0, 1)$ defines the target fidelity, and the TTS encoder terminates once it finds a support satisfying

$$R^2(S) \geq \alpha. \tag{26}$$

D.4 Construction Of The TTS Encoder

The TTS encoder τ_E^α constructs a support greedily. Starting from $S_0 = \emptyset$, iteration t evaluates every candidate latent coordinate $j \notin S_t$ by considering the enlarged support $S_t^{(j)} = S_t \cup \{j\}$. Using the TTS decoder, this candidate support produces

$$\boldsymbol{z}_{S_t^{(j)}}^{\mathrm{recon}} = \tau_D(S_t^{(j)}, \boldsymbol{z}_{S_t^{(j)}}), \qquad \hat{\boldsymbol{y}}_{S_t^{(j)}} = D(\boldsymbol{z}_{S_t^{(j)}}^{\mathrm{recon}}). \tag{27}$$

The reconstruction improvement produced by candidate j at iteration t is

$$g_j^t = R^2(S_t^{(j)}) - R^2(S_t). \tag{28}$$

This quantity measures how much candidate j improves feature-space reconstruction when added to the current support.

D.5 Activation-Aware Sparsity Term

A purely reconstruction-based rule does not distinguish among candidates based on their activation magnitude in the current input. We therefore introduce an activation-dependent term that, among positively activated coordinates, favors features with larger normalized activations:

$$p_j^{\mathrm{sparse},t} = \lambda_{\mathrm{sparse}} \mathbf{1}(z_j > 0)\,(1 - z_j)^2, \qquad \mathrm{score}^{\,t}(j) = g_j^t - p_j^{\mathrm{sparse},t}. \tag{29}$$

Since $z_j \in [0, 1]$, the penalty decreases as a positive activation approaches one. Coordinates with exactly zero activation are unpenalized and are therefore scored solely by their reconstruction gain.

The next coordinate added to the support is

$$j_t^\star = \arg\max_{j \notin S_t} \mathrm{score}^{\,t}(j), \qquad S_{t+1} = S_t \cup \{j_t^\star\}. \tag{30}$$

The TTS encoder τ_E^α is therefore defined by repeated application of this greedy update until the target fidelity is reached:

$$\tau_E^\alpha(\boldsymbol{y}, \boldsymbol{z}) = (S_\alpha, \boldsymbol{z}_{S_\alpha}), \qquad R^2(S_\alpha) \geq \alpha. \tag{31}$$

D.6 Algorithm For Test-Time Sparsification

Algorithm 1 summarizes the complete inference-time sparsification procedure. The algorithm takes as input the original feature vector $\boldsymbol{y}$, its dense latent representation $\boldsymbol{z}$, the precomputed latent statistics $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and the desired fidelity threshold α. It then constructs the explanatory support greedily, beginning from the empty set. At each iteration, every currently unselected latent coordinate is evaluated by temporarily adding it to the support, reconstructing the dense latent state with the covariance-based TTS decoder, and measuring the resulting improvement in feature-space fidelity.

The candidate score combines reconstruction gain with the activation-dependent term above, which favors more strongly activated coordinates among candidates with nonzero activation; exactly inactive coordinates are scored solely by reconstruction gain. The support is enlarged until the reconstruction reaches the target fidelity level α. The final output consists of the selected support S_α, its observed activations $\boldsymbol{z}_{S_\alpha}$, and the reconstructed dense latent state $\boldsymbol{z}_{S_\alpha}^{\mathrm{recon}}$.

Algorithm 1 Test-Time Sparsification

Input: Feature vector $\boldsymbol{y}$, latent vector $\boldsymbol{z}$, training-feature mean $\bar{\boldsymbol{y}}_{\text{train}}$, latent mean $\boldsymbol{\mu}$, covariance $\boldsymbol{\Sigma}$, decoder D, fidelity threshold α, ridge parameter ρ, sparsity weight λ_{sparse}
$\tilde{\boldsymbol{z}} \leftarrow \boldsymbol{z} - \boldsymbol{\mu}$
$S_0 \leftarrow \emptyset, \quad t \leftarrow 0$
$\boldsymbol{z}_{S_0}^{\text{recon}} \leftarrow \boldsymbol{\mu}$
$\hat{\boldsymbol{y}}_{S_0} \leftarrow D(\boldsymbol{z}_{S_0}^{\text{recon}})$
while $R^2(S_t) < \alpha$ **do**
  for each $j \notin S_t$ **do**
    $S_t^{(j)} \leftarrow S_t \cup \{j\}$
    $\hat{\tilde{\boldsymbol{z}}}_{S_t^{(j)}} \leftarrow \boldsymbol{\Sigma}_{:,S_t^{(j)}} \left(\boldsymbol{\Sigma}_{S_t^{(j)},S_t^{(j)}} + \rho \boldsymbol{I}_{|S_t^{(j)}|} \right)^{-1} \tilde{\boldsymbol{z}}_{S_t^{(j)}}$
    $\boldsymbol{z}_{S_t^{(j)}}^{\text{recon}} \leftarrow \boldsymbol{\mu} + \hat{\tilde{\boldsymbol{z}}}_{S_t^{(j)}}$
    $\hat{\boldsymbol{y}}_{S_t^{(j)}} \leftarrow D(\boldsymbol{z}_{S_t^{(j)}}^{\text{recon}})$
    $g_j^t \leftarrow R^2(S_t^{(j)}) - R^2(S_t)$
    $p_j^{\text{sparse},t} \leftarrow \lambda_{\text{sparse}} \mathbf{1}(z_j > 0)\,(1 - z_j)^2$
    $\text{score}^{\,t}(j) \leftarrow g_j^t - p_j^{\text{sparse},t}$
  end for
  $j_t^\star \leftarrow \arg\max_{j \notin S_t} \text{score}^{\,t}(j)$
  $S_{t+1} \leftarrow S_t \cup \{j_t^\star\}$
  $\boldsymbol{z}_{S_{t+1}}^{\text{recon}} \leftarrow \tau_D(S_{t+1}, \boldsymbol{z}_{S_{t+1}})$
  $\hat{\boldsymbol{y}}_{S_{t+1}} \leftarrow D(\boldsymbol{z}_{S_{t+1}}^{\text{recon}})$
  $t \leftarrow t + 1$
end while
Output: $S_\alpha \leftarrow S_t$, selected activations $\boldsymbol{z}_{S_\alpha}$, reconstructed latent state $\boldsymbol{z}_{S_\alpha}^{\text{recon}}$

E Computational Complexity Of Test-Time Sparsification

This section summarizes the computational cost of the Test-Time Sparsification procedure in Algorithm 1. Let K denote the latent dimensionality, M the dimensionality of the original feature vector, N the number of reference samples used to estimate latent statistics, and

$$T = |S_\alpha| \tag{32}$$

the final support size selected for a given input.

The complexity has two components: a one-time preprocessing cost for estimating latent statistics, and an input-dependent inference cost for constructing the sparse support.

E.1 One-Time Cost Of Latent Statistics

Before inference, TTS computes the empirical latent mean and covariance from reference latent vectors $\{\boldsymbol{z}_n\}_{n=1}^N$. The mean requires

$$\mathcal{O}(NK) \tag{33}$$

time, while the covariance matrix requires

$$\mathcal{O}(NK^2) \tag{34}$$

time and

$$\mathcal{O}(K^2) \tag{35}$$

memory. These costs are incurred once and are independent of the number of test inputs.

E.2 Cost Of One Greedy Expansion Step

At iteration t, the current support contains t selected coordinates, i.e. $|S_t| = t$. The TTS encoder must decide which additional coordinate should be added next. It therefore evaluates every remaining candidate $j \notin S_t$, of which there are $K - t$.

For each candidate, the algorithm evaluates the enlarged support $S_t^{(j)} = S_t \cup \{j\}$, computes the covariance-based latent reconstruction, decodes it into feature space, and evaluates the corresponding score. With cached support-level linear algebra and the linear-decoder precomputation

$$\boldsymbol{G} = \boldsymbol{W}_D \boldsymbol{\Sigma}, \tag{36}$$

the score of one candidate can be evaluated in

$$\mathcal{O}(t^2 + Mt) \tag{37}$$

time. The t^2 term comes from updating the support-level covariance solve, and the Mt term comes from computing the candidate feature reconstruction in the original feature space.

Since $K - t$ candidates are considered, selecting the next coordinate has a cost

$$\boxed{\mathcal{O}\left((K-t)(t^2+Mt)\right)}. \tag{38}$$

After the best candidate has been identified, updating the support itself from S_t to S_{t+1} is inexpensive relative to the search over candidates and can be handled with the same cached linear-algebra update.

This expression highlights the relevant scaling behavior of one greedy step. For a small current support $t \ll K$, the dependence on the latent dimension is linear in K, up to factors determined by the current support size and feature dimension.

E.3 Overall Inference Complexity

The support is constructed over T greedy expansion steps. Summing the single-step cost in Eq. equation 38 over $t = 0, \ldots, T-1$ gives

$$\sum_{t=0}^{T-1} (K-t)(t^2 + Mt). \tag{39}$$

Using $K - t \leq K$ and standard summation bounds,

$$\sum_{t=0}^{T-1} (K-t)t^2 = \mathcal{O}(KT^3), \qquad \sum_{t=0}^{T-1} (K-t)Mt = \mathcal{O}(MKT^2). \tag{40}$$

Therefore, the total per-input complexity of TTS support selection is

$$\boxed{\mathcal{O}\left(KT^3 + MKT^2\right)}. \tag{41}$$

The algorithm is therefore particularly attractive in the sparse regime targeted by TTS, where the final support size satisfies $T \ll K$. In this regime, the cost grows linearly with the latent dimensionality K for each support-expansion step, while the higher-order dependence is governed by the number of selected explanatory coordinates rather than by the full latent dimension.

F Optimality of the covariance reconstruction for a fixed support

This appendix establishes the optimality of the covariance-based reconstruction step used in the TTS module for a *fixed* support set. The result applies to the reconstruction stage only, not to the full greedy support-selection procedure.

Let $\boldsymbol{z} \in \mathbb{R}^K$ be a random concept vector with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and define the centered concept vector

$$\tilde{\boldsymbol{z}} = \boldsymbol{z} - \boldsymbol{\mu}. \tag{42}$$

Fix a support set $S \subseteq \{1, \ldots, K\}$, and let $\tilde{\boldsymbol{z}}_S \in \mathbb{R}^{|S|}$ denote the corresponding subvector.

The reconstruction problem considered here is the following: among all linear estimators of the form

$$\hat{\tilde{\boldsymbol{z}}} = A\tilde{\boldsymbol{z}}_S, \qquad A \in \mathbb{R}^{K \times |S|}, \tag{43}$$

find the one that minimizes the expected squared reconstruction error

$$\mathcal{L}(A) = \mathbb{E}\left[\|\tilde{\boldsymbol{z}} - A\tilde{\boldsymbol{z}}_S\|_2^2\right]. \tag{44}$$

Lemma 1. *For any matrix $A \in \mathbb{R}^{K\times|S|}$, the objective in equation 44 admits the expansion*

$$\mathcal{L}(A) = \operatorname{tr}(\mathbf{\Sigma}) - 2\operatorname{tr}(A\mathbf{\Sigma}_{S,:}) + \operatorname{tr}(A\mathbf{\Sigma}_{S,S}A^\top). \tag{45}$$

Proof. Starting from equation 44,

$$\begin{aligned}\mathcal{L}(A) &= \mathbb{E}\big[(\tilde{\boldsymbol{z}} - A\tilde{\boldsymbol{z}}_S)^\top(\tilde{\boldsymbol{z}} - A\tilde{\boldsymbol{z}}_S)\big] \\ &= \mathbb{E}\big[\tilde{\boldsymbol{z}}^\top\tilde{\boldsymbol{z}}\big] - 2\mathbb{E}\big[\tilde{\boldsymbol{z}}^\top A\tilde{\boldsymbol{z}}_S\big] + \mathbb{E}\big[\tilde{\boldsymbol{z}}_S^\top A^\top A\tilde{\boldsymbol{z}}_S\big].\end{aligned} \tag{46}$$

Each term is rewritten using trace identities:

$$\mathbb{E}\big[\tilde{\boldsymbol{z}}^\top\tilde{\boldsymbol{z}}\big] = \operatorname{tr}\big(\mathbb{E}[\tilde{\boldsymbol{z}}\tilde{\boldsymbol{z}}^\top]\big) = \operatorname{tr}(\mathbf{\Sigma}), \tag{47}$$

$$\mathbb{E}\big[\tilde{\boldsymbol{z}}^\top A\tilde{\boldsymbol{z}}_S\big] = \operatorname{tr}\big(A\,\mathbb{E}[\tilde{\boldsymbol{z}}_S\tilde{\boldsymbol{z}}^\top]\big) = \operatorname{tr}(A\mathbf{\Sigma}_{S,:}), \tag{48}$$

$$\mathbb{E}\big[\tilde{\boldsymbol{z}}_S^\top A^\top A\tilde{\boldsymbol{z}}_S\big] = \operatorname{tr}\big(A\,\mathbb{E}[\tilde{\boldsymbol{z}}_S\tilde{\boldsymbol{z}}_S^\top]\,A^\top\big) = \operatorname{tr}(A\mathbf{\Sigma}_{S,S}A^\top). \tag{49}$$

Substituting these identities proves equation 45. □

Theorem 1 (Optimal linear reconstruction for a fixed support)**.** *Assume that $\mathbf{\Sigma}_{S,S}$ is invertible. Then the unique minimizer of equation 44 is*

$$A^\star = \mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}. \tag{50}$$

Consequently, the reconstruction

$$\hat{\tilde{\boldsymbol{z}}}_S = \mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}\tilde{\boldsymbol{z}}_S \tag{51}$$

is the minimum-mean-square linear estimator of the full centered concept vector from the selected coordinates.

Proof. By Lemma 1,

$$\mathcal{L}(A) = \operatorname{tr}(\mathbf{\Sigma}) - 2\operatorname{tr}(A\mathbf{\Sigma}_{S,:}) + \operatorname{tr}(A\mathbf{\Sigma}_{S,S}A^\top). \tag{52}$$

This is a strictly convex quadratic function of A because $\mathbf{\Sigma}_{S,S}$ is positive definite. Differentiating with respect to A yields

$$\nabla_A\mathcal{L}(A) = -2\mathbf{\Sigma}_{:,S} + 2A\mathbf{\Sigma}_{S,S}. \tag{53}$$

Setting the gradient to zero gives

$$A\mathbf{\Sigma}_{S,S} = \mathbf{\Sigma}_{:,S}. \tag{54}$$

Since $\mathbf{\Sigma}_{S,S}$ is invertible, the unique solution is

$$A^\star = \mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}. \tag{55}$$

Substituting this matrix into equation 43 yields equation 51. □

Corollary 1 (Expected reconstruction error)**.** *Under the assumptions of Theorem 1, the minimum expected reconstruction error is*

$$\mathbb{E}\Big[\|\tilde{\boldsymbol{z}} - \hat{\tilde{\boldsymbol{z}}}_S\|_2^2\Big] = \operatorname{tr}\Big(\mathbf{\Sigma} - \mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}\mathbf{\Sigma}_{S,:}\Big). \tag{56}$$

Proof. Substituting $A^\star = \mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}$ into equation 45 gives

$$\begin{aligned}\mathcal{L}(A^\star) = \operatorname{tr}(\mathbf{\Sigma}) &- 2\operatorname{tr}\Big(\mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}\mathbf{\Sigma}_{S,:}\Big) \\ &+ \operatorname{tr}\Big(\mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}\mathbf{\Sigma}_{S,S}\mathbf{\Sigma}_{S,S}^{-1}\mathbf{\Sigma}_{S,:}\Big).\end{aligned} \tag{57}$$

The last two terms simplify to

$$-\operatorname{tr}\Big(\mathbf{\Sigma}_{:,S}\mathbf{\Sigma}_{S,S}^{-1}\mathbf{\Sigma}_{S,:}\Big), \tag{58}$$

which proves equation 56. □

Corollary 2 (Gaussian conditional reconstruction)**.** *If, in addition, $\tilde{\boldsymbol{z}} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, then*

$$\hat{\tilde{\boldsymbol{z}}}_S = \mathbb{E}[\tilde{\boldsymbol{z}} \mid \tilde{\boldsymbol{z}}_S]\,. \tag{59}$$

Thus, for a fixed support S, the covariance-based reconstruction coincides with the Bayes-optimal estimator under squared loss.

Proof. For a jointly Gaussian random vector, the conditional mean of the full vector given a subvector is affine and is determined by the standard Gaussian conditioning formula. Since $\tilde{\boldsymbol{z}}$ is zero-mean,

$$\mathbb{E}[\tilde{\boldsymbol{z}} \mid \tilde{\boldsymbol{z}}_S] = \boldsymbol{\Sigma}_{:,S}\boldsymbol{\Sigma}_{S,S}^{-1}\tilde{\boldsymbol{z}}_S, \tag{60}$$

which is exactly equation 51. □

In practice, the submatrix $\boldsymbol{\Sigma}_{S,S}$ may become ill-conditioned, especially in highly correlated concept spaces. For numerical stability, the following ridge-stabilized reconstruction is used in the main text:

$$\hat{\tilde{\boldsymbol{z}}}_S = \boldsymbol{\Sigma}_{:,S}\left(\boldsymbol{\Sigma}_{S,S} + \rho\boldsymbol{I}\right)^{-1}\tilde{\boldsymbol{z}}_S, \qquad \rho > 0. \tag{61}$$

The next theorem shows that this is the unique minimizer of a regularized linear reconstruction objective.

Theorem 2 (Optimal ridge-regularized linear reconstruction)**.** *For $\rho > 0$, consider the regularized objective*

$$\mathcal{L}_\rho(A) = \mathbb{E}\big[\|\tilde{\boldsymbol{z}} - A\tilde{\boldsymbol{z}}_S\|_2^2\big] + \rho\|A\|_F^2. \tag{62}$$

Then the unique minimizer is

$$A_\rho^\star = \boldsymbol{\Sigma}_{:,S}\left(\boldsymbol{\Sigma}_{S,S} + \rho\boldsymbol{I}\right)^{-1}. \tag{63}$$

Consequently, the ridge-stabilized reconstruction rule in equation 61 is the optimal regularized linear estimator.

Proof. By Lemma 1,

$$\mathcal{L}_\rho(A) = \operatorname{tr}(\boldsymbol{\Sigma}) - 2\operatorname{tr}(A\boldsymbol{\Sigma}_{S,:}) + \operatorname{tr}(A\boldsymbol{\Sigma}_{S,S}A^\top) + \rho\|A\|_F^2. \tag{64}$$

Differentiating with respect to A gives

$$\nabla_A\mathcal{L}_\rho(A) = -2\boldsymbol{\Sigma}_{:,S} + 2A\boldsymbol{\Sigma}_{S,S} + 2\rho A. \tag{65}$$

Setting the gradient to zero yields

$$A(\boldsymbol{\Sigma}_{S,S} + \rho\boldsymbol{I}) = \boldsymbol{\Sigma}_{:,S}. \tag{66}$$

Since $\rho > 0$, the matrix $\boldsymbol{\Sigma}_{S,S} + \rho\boldsymbol{I}$ is positive definite and therefore invertible. Hence,

$$A_\rho^\star = \boldsymbol{\Sigma}_{:,S}\left(\boldsymbol{\Sigma}_{S,S} + \rho\boldsymbol{I}\right)^{-1}, \tag{67}$$

which proves the claim. □

The preceding results establish that, for any fixed support, the reconstruction stage used by the TTS module is optimal within the class of linear estimators under the empirical second-order model. In the unregularized case, it is the minimum-mean-square linear estimator; in the ridge-stabilized case used in practice, it is the optimal regularized linear estimator. These statements justify the reconstruction rule itself, while the support-selection stage remains a greedy approximation to the underlying combinatorial subset-selection problem.

G Additional Details For Extractor Evaluation

This appendix describes the evaluation protocol used to compare concept extractors in Section 4. The goal is to measure whether an extractor produces semantically coherent concept activations while retaining enough information to reconstruct the original hidden representation.

G.1 General Evaluation Procedure

All evaluations use frozen vision models. For each model–dataset pair, we choose a target layer, collect hidden features from training images, fit each extractor on these features, and evaluate the resulting concept activations on validation images. Each extractor defines an encoder and decoder,

$$\boldsymbol{z} = E(\boldsymbol{y}), \qquad \hat{\boldsymbol{y}} = D(\boldsymbol{z}), \tag{68}$$

where $\boldsymbol{y}$ is the frozen hidden representation, $\boldsymbol{z}$ is the extracted concept representation, and $\hat{\boldsymbol{y}}$ is the reconstructed feature vector.

The main evaluation uses three model–dataset pairs. For Places365, we evaluate DenseNet161 and aggregate spatial activations by averaging. For ImageNet, we evaluate ResNet18 using the flattened average-pooling representation. For OpenImages, we evaluate CLIP RN50 at the final visual residual block and mean-pool its spatial representation. These settings provide scene-centric, object-centric, and multimodal visual representations.

For each extractor, we store the fitted encoder and decoder, validation concept activations, reconstructed features, and per-concept evaluation scores. All final learned-extractor comparisons use the same cached features and training budget, with dictionary size $K = 5{,}000$ unless stated otherwise.

G.2 Compared Extractors

We compare linear and nonlinear concept extractors. The linear group contains Identity, PCA, Random, NMF, RAV, and KMeans. Identity acts as a reconstruction reference, PCA captures variance-aligned directions, Random provides a non-semantic reference, NMF provides a non-negative factorization baseline, RAV represents activation-derived directions, and KMeans uses cluster centroids as concept directions.

The nonlinear group contains KMeansAE, Vanilla SAE, TopK SAE, JumpReLU SAE, and Archetypal SAE. KMeansAE initializes a non-negative autoencoder from KMeans directions and trains for reconstruction with a geometry-alignment penalty. Vanilla SAE uses an ℓ_1 penalty on latent activations, TopK SAE explicitly retains a fixed number of activations, JumpReLU SAE learns activation thresholds, and Archetypal SAE constrains its learned dictionary through observed representations.

For sparse autoencoders, sparsity strength is tuned before the full evaluation to target validation reconstruction $R^2 \approx 0.8$. This avoids comparing semantic quality at arbitrarily different reconstruction fidelities.

G.3 Evaluation Metrics

We evaluate concept quality using complementary semantic metrics. **Clarity** measures whether the images that most strongly activate a concept are visually and semantically coherent. For each concept, we select the top-activating validation images, obtain their CLIP image embeddings, normalize them, and compute their average pairwise cosine similarity.

INVERT measures how strongly concept activations recover dataset labels. For each concept and label, we estimate how well the activation distinguishes positive from negative examples using an approximate ROC-AUC, and report the strongest positive label association.

CLIP-Dissect provides an open-vocabulary semantic measure by comparing the visual evidence associated with a concept against CLIP text embeddings from a candidate vocabulary. We report its top-1 softWPMI score.

Our primary aggregate,

$$\text{Avg} = \frac{\text{Clarity} + \text{INVERT} + \text{CLIP-Dissect}}{3}, \tag{69}$$

therefore combines top-image coherence, dataset-label alignment, and open-vocabulary semantic alignment.

We additionally report the **MS** monosemanticity score as a separate diagnostic. MS uses normalized concept activations as weights over CLIP image embeddings and measures weighted semantic

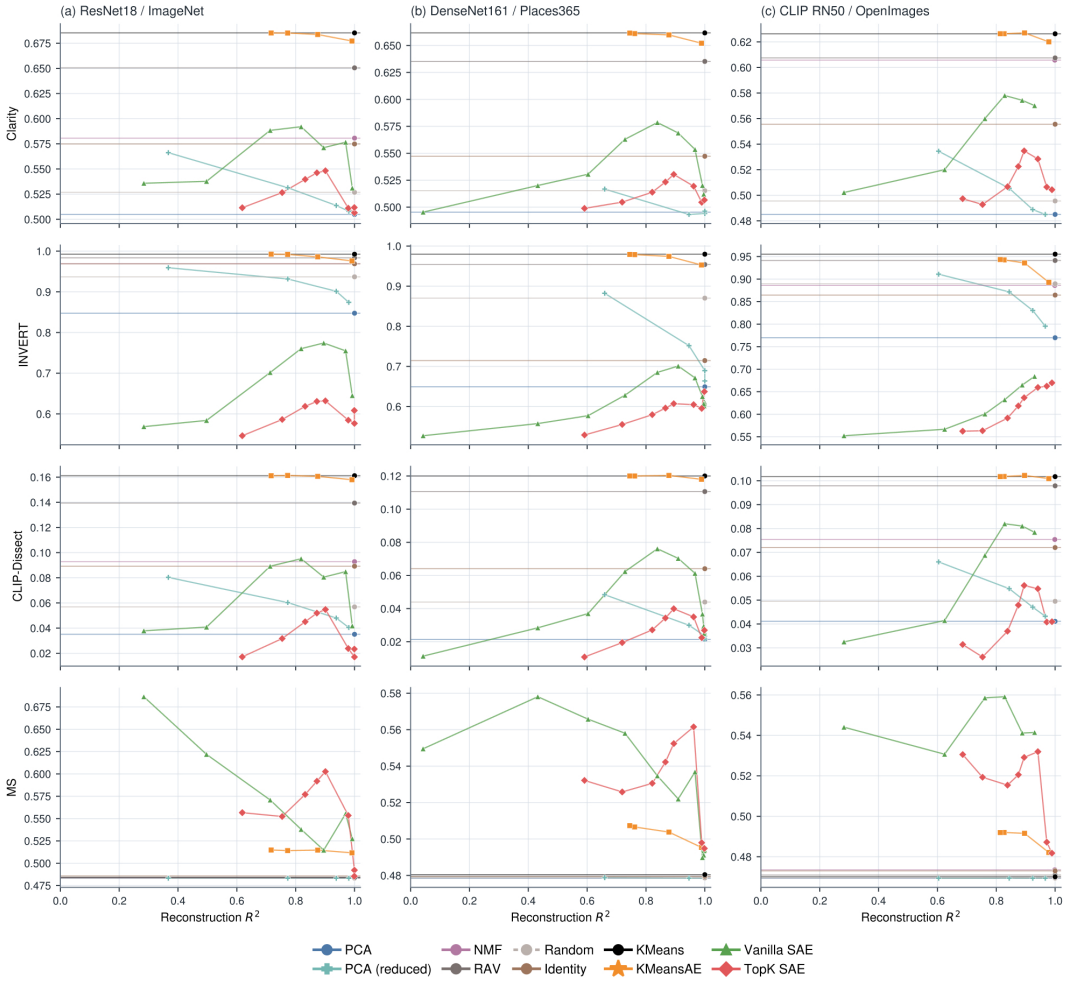

Figure 8: **Individual concept-quality metrics across the reconstruction frontier.** Clarity, INVERT, CLIP-Dissect, and MS are shown separately as functions of validation reconstruction R^2 for ResNet18/ImageNet, DenseNet161/Places365, and CLIP RN50/OpenImages. The first three rows are the metrics averaged to obtain the Avg score used in Figure 3. They consistently distinguish the high-semantic-quality KMeans and KMeansAE representations from the evaluated SAE trajectories. In contrast, the MS scores of the dense linear extractors are tightly clustered despite substantial differences under the other three metrics, indicating that MS provides limited discrimination for these representations. We therefore report MS separately and do not include it in Avg.

concentration across the full validation set. We do not include MS in Avg because it has limited discriminative power for the dense linear extractors in this evaluation: KMeans, Identity, RAV, Random, PCA, and NMF receive tightly clustered MS scores even though they differ substantially under Clarity, INVERT, and CLIP-Dissect. We therefore retain MS for completeness but base the main aggregate on the three metrics that more clearly distinguish semantic quality in this setting.

Reconstruction quality uses the same feature-space R^2 as in the main paper,

$$R^2_{\mathrm{AE}} = 1 - \frac{\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2}{\|\boldsymbol{y} - \bar{\boldsymbol{y}}_{\mathrm{train}}\|_2^2 + \varepsilon}, \tag{70}$$

where $\bar{\boldsymbol{y}}_{\mathrm{train}}$ is computed once from the corresponding complete training-feature set.

G.4 Metric-Wise Reconstruction–Interpretability Results

Figure 8 decomposes the aggregate result from Figure 3. The main pattern is not driven by a single constituent metric: KMeans and strongly aligned KMeansAE configurations perform favorably

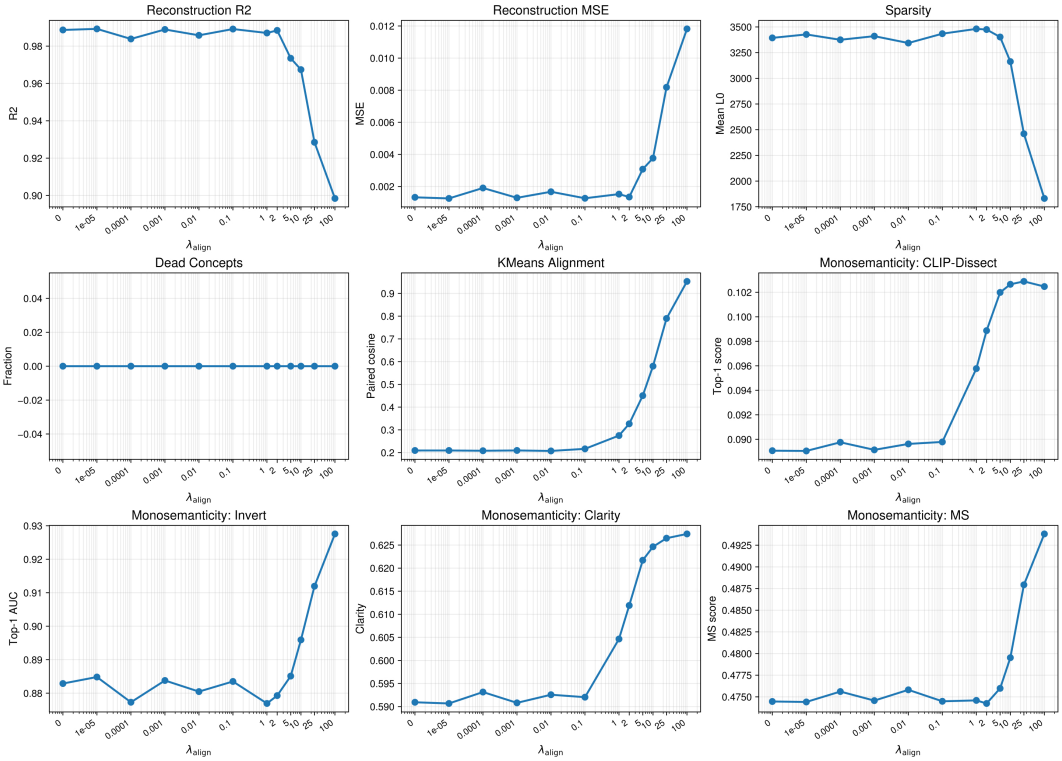

Figure 9: **Effect of KMeans alignment.** Sweep over the KMeansAE alignment coefficient λ_{align} on CLIP RN50/OpenImages. Stronger alignment generally improves the primary semantic metrics, including INVERT, CLIP-Dissect, and Clarity, while reducing reconstruction fidelity. MS is shown as a separate supplementary diagnostic.

across Clarity, INVERT, and CLIP-Dissect. Vanilla and TopK SAE trajectories generally improve semantically as their reconstruction regime changes, but remain below the geometry-grounded representations over much of the evaluated frontier.

The MS panel behaves differently. In particular, several dense linear extractors that are clearly separated by the other semantic metrics receive nearly identical MS values. This explains why MS is useful as a supplementary diagnostic but is not included in the primary average.

G.5 SAE Strength Tuning

Sparse autoencoders have a reconstruction–sparsity trade-off controlled by their main sparsity hyperparameter. For TopK and Archetypal SAE, this parameter is the number of retained activations. For Vanilla SAE and JumpReLU SAE, it is the activation penalty strength. We tune these parameters on a validation subset before the full evaluation.

The tuning procedure trains candidate models over a predefined search grid and evaluates each candidate by validation reconstruction R^2 and average latent L_0. The selected candidate is the sparsest model that reaches the target reconstruction level, with target $R^2 \approx 0.8$. If no candidate reaches the target region, the selection falls back to the candidate closest to the target reconstruction, preferring lower L_0 when reconstruction is similar.

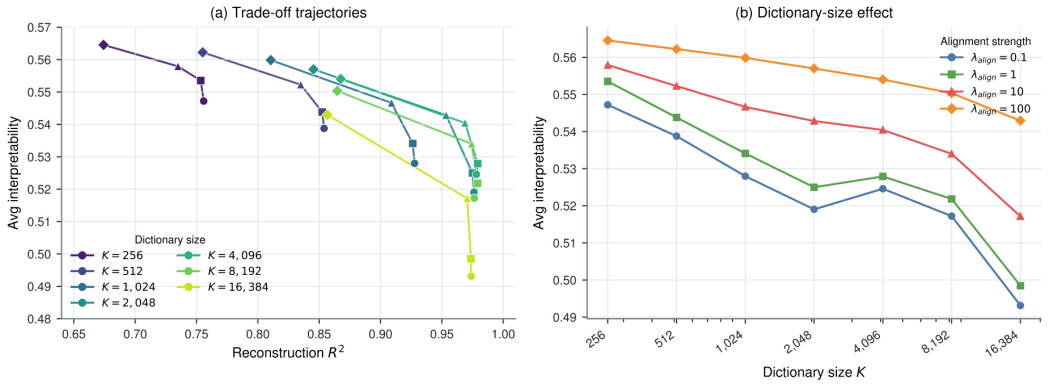

Figure 10: **Reconstruction–interpretability trade-off across dictionary sizes. Left:** empirical reconstruction–interpretability trajectories for dictionary sizes $K = 256$ to 16,384, with points along each trajectory corresponding to different values of λ_{align}. **Right:** average interpretability as a function of dictionary size for fixed alignment strengths. Increasing dictionary size generally improves reconstruction while reducing semantic quality, whereas stronger alignment consistently preserves higher interpretability. Even across the full $64\times$ range in dictionary size, strong alignment retains comparatively stable semantic quality.

G.6 KMeansAE Alignment Sweep

KMeansAE uses an alignment term that encourages the learned encoder to remain close to the KMeans initialization:

$$\mathcal{L}_{\text{AE}} = \frac{1}{N}\sum_{n=1}^{N} \|\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n\|_2^2 + \lambda_{\text{align}} \|\boldsymbol{W}_E - \boldsymbol{W}_0\|_F^2. \tag{71}$$

Small values of λ_{align} allow the encoder to adapt more freely for reconstruction, while large values keep the learned directions closer to the original KMeans geometry.

Figure 9 shows the resulting reconstruction–interpretability trade-off. Stronger alignment improves Clarity, INVERT, and CLIP-Dissect, but eventually reduces reconstruction R^2 because the encoder has less freedom to adapt. This supports the use of geometry alignment as the mechanism that preserves the semantic structure of the KMeans initialization while still allowing the autoencoder to optimize reconstruction.

G.7 Effect of Dictionary Size

To test whether the reconstruction–interpretability behavior depends on a particular dictionary capacity, we repeat the KMeansAE experiment on CLIP RN50/OpenImages for

$$K \in \{256, 512, 1024, 2048, 4096, 8192, 16384\}$$

and

$$\lambda_{\text{align}} \in \{0.1, 1, 10, 100\}.$$

All configurations use the same optimization protocol; each dictionary size shares one KMeans initialization across its four alignment values.

Figure 10 shows that dictionary capacity and geometry alignment control distinct aspects of the representation. Increasing K generally improves reconstruction, while stronger alignment shifts the semantic-quality frontier upward. At $\lambda_{\text{align}} = 100$, Avg decreases only from approximately 0.565 at $K = 256$ to 0.543 at $K = 16{,}384$, despite the $64\times$ increase in dictionary size.

The individual metrics in Figure 11 support the aggregate trend. Reconstruction increases rapidly with dictionary size before saturating, while the fraction of active coordinates decreases. Clarity, INVERT, and CLIP-Dissect generally decline as capacity increases, particularly under weak alignment, whereas

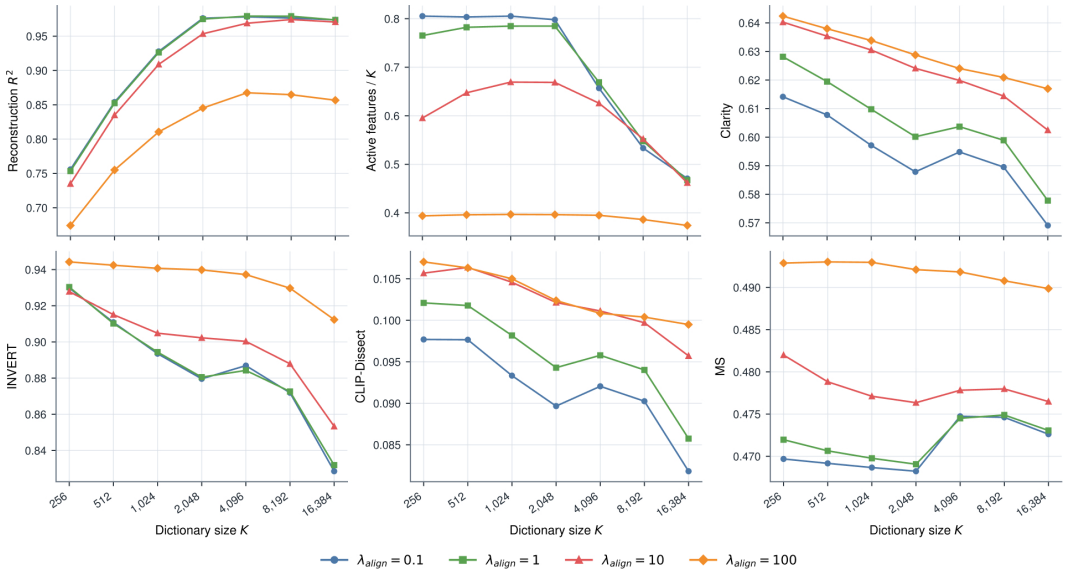

Figure 11: **Detailed effect of KMeansAE dictionary size.** Reconstruction R^2, fraction of active features, Clarity, INVERT, CLIP-Dissect, and MS as functions of dictionary size on CLIP RN50/OpenImages for four alignment strengths. Larger dictionaries substantially improve reconstruction and reduce the fraction of active coordinates. The primary semantic metrics generally decrease with increasing K, but stronger alignment preserves substantially higher semantic quality throughout the sweep. MS is again shown separately as a supplementary diagnostic.

large λ_{align} substantially mitigates this loss. These results indicate that the main reconstruction–interpretability behavior is not specific to the $K = 5{,}000$ operating point used elsewhere in the paper.

G.8 Layerwise CLIP ViT-B/32 Evaluation

The main extractor comparison evaluates one representation per model. To test whether the observed behavior depends on the chosen layer, we perform a layerwise evaluation on CLIP ViT-B/32 using OpenImages. We compare KMeans, KMeansAE, and Vanilla SAE across visual transformer residual blocks and the final embedding representation.

Figure 12 shows that KMeans and KMeansAE follow similar trends across depth. Their Clarity and CLIP-Dissect scores generally improve in later layers and in the final embedding. KMeansAE remains close to KMeans on these semantic metrics while maintaining high reconstruction quality. Vanilla SAE is substantially sparser, but its reconstruction is less stable and its Clarity and CLIP-Dissect scores remain below KMeans and KMeansAE across most layers. Vanilla SAE obtains higher MS values in some later layers, but we treat this metric as supplementary for the reasons discussed in Appendix G.4.

G.9 PCA Analysis

Although PCA is weaker than KMeans in the main extractor comparison, its components are naturally ordered by explained variance. Figure 13 shows that leading PCA components are generally more interpretable than later components, particularly under INVERT and CLIP-Dissect. Clarity shows a similar but weaker pattern, while MS is comparatively stable across components. This stability is consistent with the limited discrimination of MS among dense linear representations observed in Figure 8.

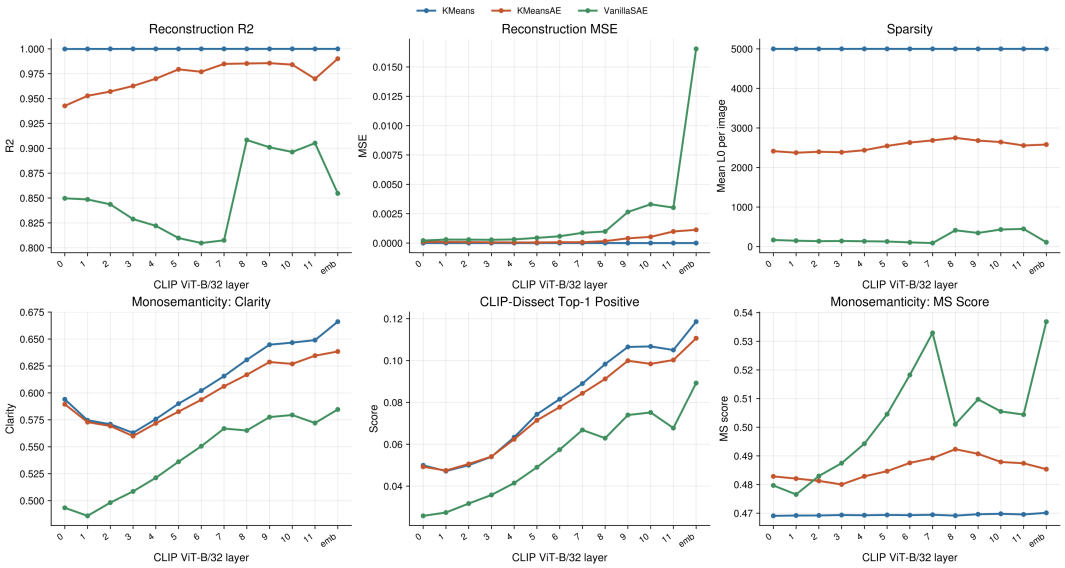

Figure 12: **Layerwise extractor behavior.** Layerwise evaluation of KMeans, KMeansAE, and Vanilla SAE on CLIP ViT-B/32 using OpenImages. KMeans and KMeansAE show similar depth-dependent concept-quality trends, with semantic scores generally improving in later layers and the final embedding. KMeansAE preserves strong reconstruction quality while closely tracking KMeans on Clarity and CLIP-Dissect. Vanilla SAE is substantially sparser but has weaker Clarity and CLIP-Dissect scores and less stable reconstruction behavior.

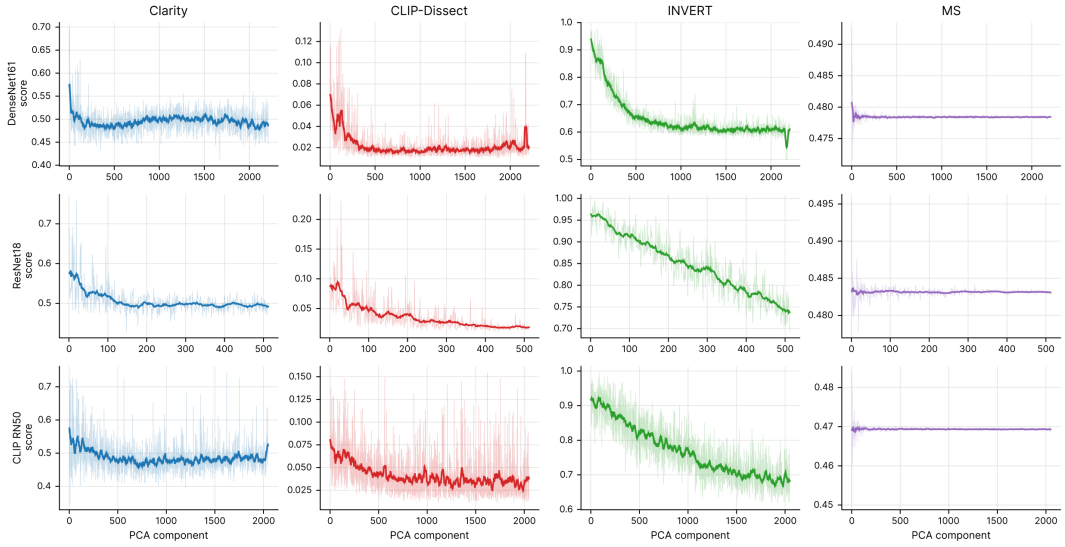

Figure 13: **Interpretability of PCA components.** Concept-quality scores for individual PCA components across DenseNet161, ResNet18, and CLIP RN50. Components are ordered by decreasing explained variance, and darker curves show a moving average over 25 components. Leading principal components are consistently more interpretable than later components under Clarity, CLIP-Dissect, and INVERT.

H Additional Details for Test-Time Sparsification

H.1 Reconstruction–Sparsity Evaluation

We evaluate TTS on the same three model–dataset pairs as the extractor analysis: ResNet18/ImageNet, DenseNet161/Places365, and CLIP RN50/OpenImages. Each setting uses a 5,000-concept

KMeansAE dictionary and $1{,}000$ deterministic held-out validation examples. TTS, matching pursuit (MP), and activation Top-K are evaluated at target reconstruction levels

$$R^2 \in \{0.75, 0.85, 0.90, 0.95\}.$$

For comparison, we additionally evaluate Vanilla SAE checkpoints over multiple L_1 penalties and TopK SAE checkpoints over multiple fixed support sizes.

All methods use the same feature-space reconstruction metric,

$$R^2 = 1 - \frac{\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2}{\|\boldsymbol{y} - \boldsymbol{\mu}_{\text{train}}\|_2^2}, \tag{72}$$

where $\boldsymbol{\mu}_{\text{train}}$ is computed from the corresponding complete training-feature set. Thus, reconstruction values are directly comparable across TTS, MP, activation Top-K, and the SAE baselines.

For dynamic methods, latent coordinates are normalized using their maximum activation on the complete training set. TTS and MP are run once up to the maximum target fidelity, and saved prefix trajectories are used to evaluate intermediate reconstruction targets. Static SAE operating points are evaluated on the same held-out examples.

H.2 Positive and Total Support

The iterative procedures admit several notions of support. We distinguish:

- **Positive support:** refitted coefficients greater than 10^{-8};
- **Nonzero support:** refitted coefficients with absolute magnitude greater than 10^{-8};
- **Total support:** all coordinates selected by the greedy search.

A coordinate can therefore be selected during the iterative search but later receive a zero or non-positive coefficient after active-set refitting. The main-paper reconstruction curves report positive support, which quantifies the positively weighted concepts associated with the final reconstruction. Total support instead measures the breadth of the iterative search and is relevant both for understanding the procedure and for its computational cost.

Figure 14 clarifies this distinction. The positive support of TTS remains very small even as reconstruction fidelity increases, whereas activation Top-K, the SAE baselines, and MP require substantially more positive active features. At the same time, TTS may select many additional coordinates during its iterative search before refitting. This behavior reflects the additional computation used to construct the compact positive explanation and motivates the runtime analysis below.

H.3 Cost of Test-Time Adaptivity

TTS performs an iterative, input-specific support search rather than a single static encoder pass. We therefore benchmark its inference-time cost separately. The benchmark uses CLIP RN50/OpenImages and measures complete inference and explanation throughput for the standard model, Vanilla SAE, MP, and TTS.

The primary benchmark uses 256 images in matched batches of 16, excludes one warm-up batch per method, and evaluates TTS at target $R^2 = 0.80$. TTS uses an internal batch size of 16 and a maximum of 2,048 selection steps. We additionally evaluate a 64-image grid over target reconstruction levels

$$R^2 \in \{0.25, 0.50, 0.60, 0.70, 0.80, 0.90\}$$

and sparsity-penalty values

$$\alpha \in \{0, 0.25, 0.50, 0.75, 1.0\}.$$

Figure 15 makes explicit the computational trade-off introduced by test-time sparsification. The additional search enables explanation size to adapt to each input and desired reconstruction fidelity, but it is orders of magnitude slower than static SAE encoding or an ordinary model forward pass. Runtime also depends strongly on the requested reconstruction target: higher fidelity requires longer support searches.

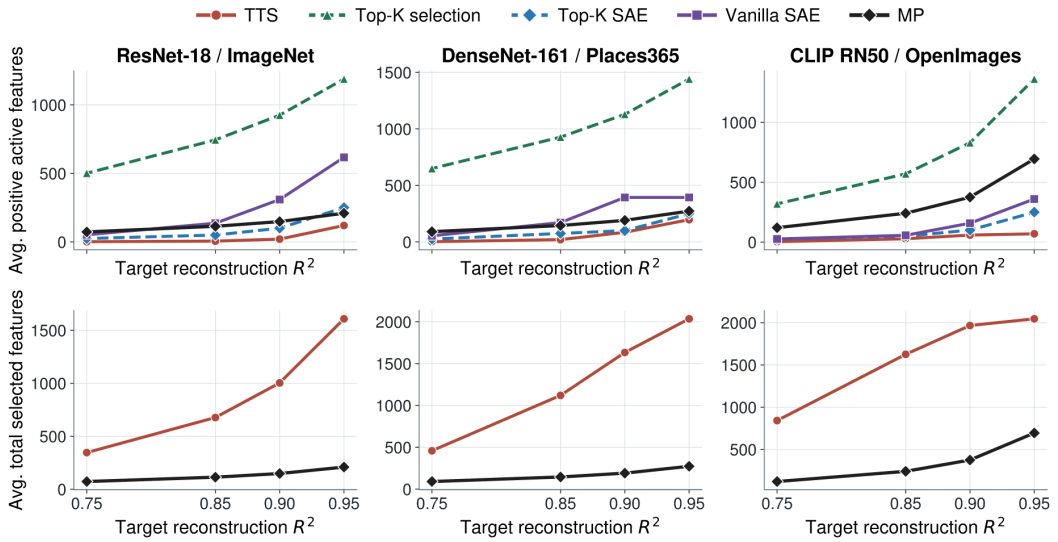

Figure 14: **Positive active support and total selected support. Top:** average number of positively weighted active features required at increasing reconstruction targets. TTS maintains substantially smaller positive support than MP, activation Top-K, TopK SAE, and Vanilla SAE across all three model–dataset pairs. **Bottom:** total number of coordinates selected during the iterative TTS and MP searches. TTS can inspect and select substantially more coordinates than remain positively weighted after refitting, particularly at high reconstruction fidelity. Thus, the strong sparsity in the main paper refers specifically to the final positive active support rather than the total number of greedy selections.

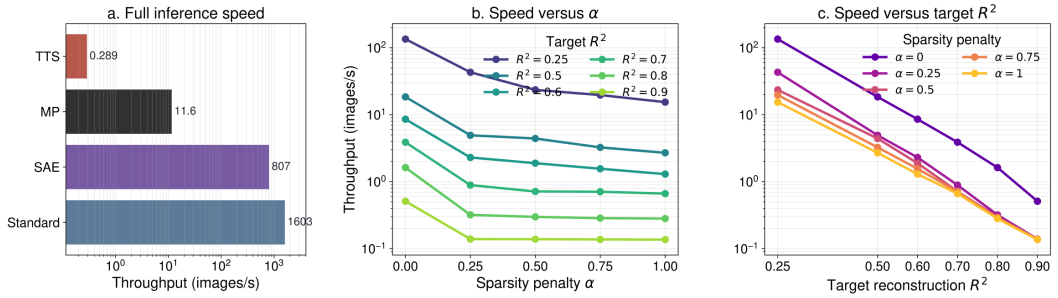

Figure 15: **The sparsity gained by TTS comes with additional test-time computation. (a)** Full inference throughput on CLIP RN50/OpenImages. TTS processes 0.289 images/s, compared with 11.6 for MP, 807 for Vanilla SAE, and 1603 for the standard model. **(b)** TTS throughput as a function of sparsity penalty and target reconstruction fidelity. **(c)** The same measurements organized by target R^2. Higher requested fidelity substantially increases the iterative test-time cost. TTS is therefore intended for on-demand explanation rather than latency-critical prediction.

This speed experiment is a separate historical batched benchmark and was not rerun with the final $\lambda_{\text{align}} = 10$ KMeansAE checkpoint used for the reconstruction–sparsity curves in Figure 4. We therefore use it to characterize the computational scaling and practical cost of the TTS procedure, rather than combining its numerical results directly with the final reconstruction experiment.

I Additional Details for Functional Steering

This section provides additional details for the functional evaluation in Section 4.3. We test whether semantic TTS-AE coordinates can serve as control variables and whether jointly modifying statistically related concepts yields more effective and distribution-compatible interventions than modifying a single latent coordinate.

I.1 CORE-MODEL STEERING PROTOCOL

Models and targets. We evaluate the final TTS-AE extractors for ResNet18/ImageNet, DenseNet161/Places365, and CLIP RN50/OpenImages. Each setting uses five semantic targets and 128 held-out images per target. The targets are acoustic guitar, African elephant, bookcase, cello, and fire engine for ResNet18; amphitheater, aquarium, cafeteria, corridor, and playground for DenseNet161; and bicycle, book, car, chair, and dog for CLIP RN50.

Target-concept matching. Each semantic target is matched to one latent coordinate using training data only. Positive and negative examples are divided into disjoint selection and validation folds. We compute a one-vs-rest AUROC for every latent on the selection fold, retain the highest-scoring coordinate, and evaluate its AUROC on the validation fold. No held-out intervention image or downstream steering outcome is used for concept matching. The intervention target $a^\star$ is the 90th percentile of the selected latent activation on the training statistics sample.

Steering. Let z denote the non-negative, train-max-normalized latent representation, j the target coordinate, and

$$\delta_j = \gamma(a^\star - z_j) \tag{73}$$

the requested target-coordinate change at strength γ. Standard steering uses

$$\Delta z^{\mathrm{std}} = \delta_j e_j, \qquad z^{\mathrm{std}} = \Pi_+(z + \Delta z^{\mathrm{std}}). \tag{74}$$

Covariance-aware steering instead follows

$$d_j^{\mathrm{cov}} = \frac{\Sigma_{:,j}}{\Sigma_{jj} + \rho}, \qquad \Delta z^{\mathrm{cov}} = \beta \delta_j d_j^{\mathrm{cov}}, \tag{75}$$

with $z^{\mathrm{cov}} = \Pi_+(z + \Delta z^{\mathrm{cov}})$ and $\rho = 10^{-6}$. The scalar β is determined separately for each example so that

$$\|D(z^{\mathrm{cov}}) - D(z)\|_2 \approx \left\|D(z^{\mathrm{std}}) - D(z)\right\|_2. \tag{76}$$

Thus, the two methods inject approximately equal-magnitude perturbations into the original model feature space. The random control uses a deterministic random latent direction with zero target-coordinate component and is matched using the same procedure.

For all methods, the intervention applied to the frozen model is the decoded residual

$$\Delta y = D(z^{\mathrm{steer}}) - D(z), \tag{77}$$

which is added to the original representation y.

Strengths and target efficacy. We evaluate $\gamma \in \{1, 5, 10, 25, 50\}$. The primary target-specific quantity is the fraction of the available rank gap closed,

$$G_{\mathrm{rank}} = \frac{r_{\mathrm{before}} - r_{\mathrm{after}}}{r_{\mathrm{before}} - 1}. \tag{78}$$

A value of zero indicates no improvement, one indicates that the intended target reaches rank one, and negative values indicate movement away from the target. Confidence intervals are obtained by bootstrap resampling held-out images, with equal total weight assigned to each semantic target.

I.2 DISTANCE TO EMPIRICAL REPRESENTATION SUPPORT

To assess whether steering remains close to observed representations without reusing the covariance matrix that defines covariance-aware steering, we use a nearest-neighbor diagnostic. Let

$$\mathcal{Z}_{\mathrm{train}} = \{z_1, \ldots, z_N\} \tag{79}$$

denote real latent representations from the training distribution. For a representation z, let $\mathcal{N}_k(z)$ denote its k nearest training latents in Euclidean distance. We define

$$d_{\mathrm{kNN}}(z) = \frac{1}{k} \sum_{z_i \in \mathcal{N}_k(z)} \|z - z_i\|_2, \tag{80}$$

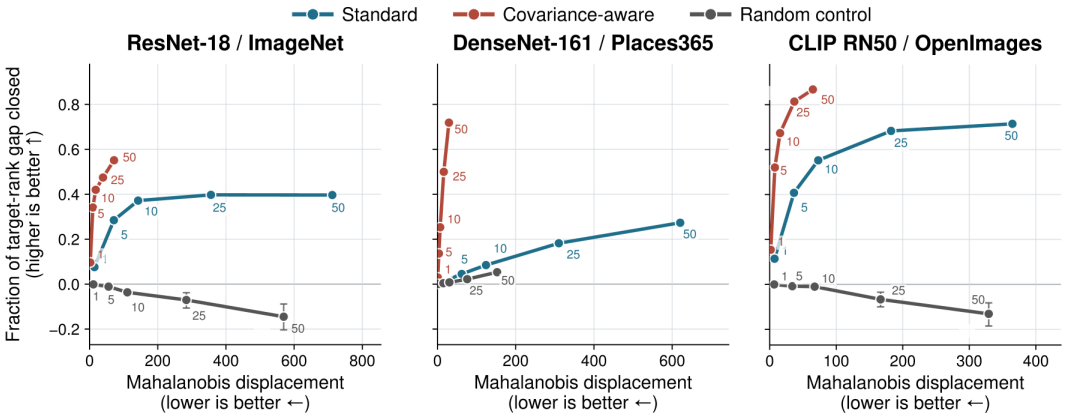

Figure 16: **Steering efficacy versus covariance-normalized latent displacement.** For ResNet18/ImageNet, DenseNet161/Places365, and CLIP RN50/OpenImages, we plot the fraction of the target-rank gap closed against the Mahalanobis displacement of the latent intervention. All methods are matched per example in decoded feature-space perturbation norm, and point labels denote steering strength. Covariance-aware steering achieves stronger target-directed changes at substantially smaller Mahalanobis displacement than standard steering. Because both the intervention direction and this diagnostic depend on the same empirical covariance matrix, we treat this result as evidence of consistency with the learned second-order geometry rather than as an independent estimate of manifold proximity.

and measure the change induced by steering as

$$\Delta d_{\mathrm{kNN}} = d_{\mathrm{kNN}}(z^{\mathrm{steer}}) - d_{\mathrm{kNN}}(z). \tag{81}$$

We use $k = 5$ in the main experiment. A positive value indicates that steering moves the representation farther from nearby real training representations, while a value near or below zero indicates that it remains within, or moves toward, locally observed representation support. Figure 5 jointly compares this quantity with target-rank gap closure.

Because this diagnostic depends only on distances to real training representations and does not use Σ, it provides an independent test of whether covariance-aware steering better preserves empirical representation support.

I.2.1 Mahalanobis Geometry Diagnostic

As a complementary diagnostic, we also measure displacement under the covariance geometry itself. For a latent update Δz, we compute

$$d_M(\Delta z) = \sqrt{\Delta z^\top (\Sigma + \epsilon I)^{-1} \Delta z}. \tag{82}$$

The Mahalanobis analysis therefore complements the covariance-independent kNN result. It verifies that covariance-aware steering follows the second-order geometry it was designed to exploit, while the kNN analysis tests proximity to observed training representations directly.

I.3 Qwen Visual-Token Steering Example

We additionally use Qwen2-VL-2B-Instruct to illustrate that the same intervention mechanism can modify open-ended multimodal behavior. Qwen is treated separately from the three core models because the available Qwen-specific KMeansAE is a frozen earlier extractor rather than the final extractor used in the main quantitative evaluation.

The visual representation is obtained by mean-pooling the projected visual tokens produced by the Qwen visual PatchMerger. We encode this pooled representation with the frozen KMeansAE, apply the latent intervention, and decode the resulting feature residual. The residual is then added to each

projected visual token before generation, preserving the token-specific variation while shifting their common visual representation.

For the example in Figure 6, we steer the concept *car*. The corresponding latent is identified from OpenImages training features using the same AUROC-based concept-matching procedure as for the core models. Covariance-aware steering follows the corresponding covariance direction and is projected onto the non-negative latent domain. Its scale is chosen using the same decoded feature-norm matching procedure described above.

The unmodified and steered models are prompted with

```
Describe the image briefly.  Mention the main objects.
```

In the displayed example, covariance-aware steering toward *car* introduces the target concept into the generated description while retaining the restaurant setting, elephant, people, and other surrounding scene content. The example is intended as a qualitative demonstration that interventions in the learned concept space can propagate to free-form VLM outputs.

J Additional Details for Global Explanations

This appendix provides additional details for the global explanation analysis in Section 5. The local TTS and global procedures operate on the same dense TTS-AE concept representation but answer different questions. Local TTS asks which concepts are sufficient to reconstruct one image representation, whereas the global analysis asks which concepts systematically predict a fixed model target across a reference distribution.

J.1 Sparse Global Readout

Let

$$\boldsymbol{z}(x) = E(f(x)) \in \mathbb{R}^K \tag{83}$$

denote the dense TTS-AE representation. For a scalar frozen-model target $g_c(x)$, concept activations are standardized using statistics from the fitting split,

$$\widetilde{z}_j(x) = \frac{z_j(x) - \mu_j}{\sigma_j + \epsilon}. \tag{84}$$

A sparse global explanation is then

$$\widehat{g}_c(x) = b_c + \sum_{j \in S_c} w_{cj} \widetilde{z}_j(x), \tag{85}$$

where S_c is a small target-specific support.

The target depends on the frozen evaluator. For ImageNet ResNet18, g_c is the pre-softmax class logit. For CLIP RN50, it is the cosine similarity between the normalized image representation and the frozen text embedding of `a photo of a {class}`. For Qwen2-VL-2B-Instruct, it is the teacher-forced $\log p(\mathrm{Yes}) - \log p(\mathrm{No})$ for a fixed target-presence question.

Unlike local TTS, the global support S_c is shared across images. A positive coefficient w_{cj} means that, conditional on the other selected terms in the sparse readout, larger activation of concept j predicts a larger target output. A negative coefficient has the opposite interpretation. These coefficients describe predictive associations and are not interpreted as causal effects by themselves.

J.2 Global Faithfulness

We measure how faithfully the sparse readout predicts the frozen target on held-out data using

$$R^2_{\mathrm{global}}(c) = 1 - \frac{\sum_n \left(g_c(x_n) - \widehat{g}_c(x_n)\right)^2}{\sum_n \left(g_c(x_n) - \bar{g}_c\right)^2 + \epsilon}, \tag{86}$$

where $\bar{g}_c$ is the mean target value on the evaluation split. This differs from the feature-space reconstruction R^2 used by TTS: here the quantity being approximated is the frozen model output rather than its internal representation.

We additionally report marginal concept–target associations to aid semantic inspection. These measure the univariate relationship between one concept and the target and are complementary to the multivariate coefficients in Eq. 85. Correlated concepts can have similar marginal associations while only one is retained by a sparse multivariate support.

J.3 Illustrative ResNet18 Global Explanation

The qualitative report in the top panel of Figure 7 uses ImageNet ResNet18 and its frozen TTS-AE representation. The global readout is fitted over all $5{,}000$ concept coordinates using a positive $L = 4$ OMP-style head. The frozen backbone and concept encoder use the complete cached ImageNet training set of $1{,}281{,}167$ examples, while the disjoint $50{,}000$-image ImageNet validation set is used as held-out test data.

For the *tennis ball* class, the resulting four-concept readout reaches train $R^2 = 0.5334$ and test $R^2 = 0.5401$. The strongest selected coordinate is directly labeled *tennis ball*; the remaining coordinates correspond to Rhodesian ridgeback, Yorkshire terrier, and Welsh springer spaniel concepts. The concept cards in Figure 7 show the multivariate readout coefficient, the marginal concept–target association, and top-activating images.

This qualitative $L = 4$ report is intended to illustrate how a sparse global explanation can be inspected. The quantitative generated-evidence experiment below uses a separately fitted signed sparse protocol and should not be interpreted as re-evaluating this exact four-concept estimator.

J.4 Generated-Evidence Validation

The generated-evidence experiment tests whether positively weighted concepts identified by sparse global readouts prospectively increase the corresponding frozen target score. The protocol uses 20 predeclared concrete ImageNet targets and three source-model representations:

- ImageNet ResNet18,
- CLIP RN50,
- Qwen2-VL-2B-Instruct.

The model checkpoints, TTS-AE activations, VLMGrid descriptions, target definitions, class manifest, and sparse operating points were frozen before any image generation.

Concept selection. For each source-model and target pair, the signed sparse global head provides up to three selected coordinates with positive standardized coefficients. If fewer than three selected coefficients are positive, we do not backfill the support with unselected concepts. Each coordinate is represented by its previously frozen VLMGrid concept description.

To avoid trivial lexical leakage, the primary analysis discards positive descriptions containing the target class name, a curated synonym, or a direct or useful hypernym. Unfiltered cases are retained separately for audit. For each positive concept, we construct a deterministic random control from the same checkpoint-compatible concept-label pool, excluding selected coordinates and lexical leaks. Controls are matched first by description length and then by activation prevalence. We additionally include the most-negative selected concepts and the direct target name as separate controls.

Generation conditions. Images are generated with FLUX.1-schnell at 512×512 resolution, four denoising steps, guidance scale 0, BF16, and three fixed seeds. All conditions use the same prompt template:

```
A clear, realistic photograph containing {concepts}.
```

Apart from the direct target-name control, the prompts contain no target class, model name, readout coefficient, rank, or desired model outcome.

For each source-target-seed tuple, the planned conditions are:

- each of up to three positive concepts individually;

Table 1: **Generated-evidence validation.** Mean standardized target score for diagonal source=evaluator comparisons. Positively weighted concepts produce larger target responses than matched random and negative controls, and combining selected concepts strengthens the effect.

Condition	Mean target score
Negative three-concept bundle	-0.47
Matched random concept	-0.41
Selected positive concept	$+0.26$
Top-2 positive concepts	$+1.76$
Top-3 positive concepts	$+1.95$
Target-name control	$+3.87$

- the top two positive concepts jointly;
- the top three positive concepts jointly;
- matched random concepts individually;
- a matched random three-concept bundle;
- the three most-negative selected concepts jointly;
- the direct target-name positive control.

The maximum design contains $1{,}980$ generations. The completed experiment contains $1{,}977$ result units because a small number of sparse supports contain fewer eligible concepts. Images are not regenerated after target scores are inspected.

Independent generation checks. GroundingDINO is used independently of the three target evaluators to check whether the requested concept is visible and whether the target object is accidentally introduced. Failed checks are retained and flagged rather than silently removed. A deterministic 10% subset is additionally exported for blinded human auditing.

J.5 Scoring and Aggregation

Every generated image is evaluated by all three frozen evaluators, producing a 3×3 source-explanation/evaluator transfer design. To make target scores comparable across models and targets, each frozen output is standardized using its clean held-out target distribution:

$$g_c^{\mathrm{std}}(x) = \frac{g_c(x) - \mu_{c,\mathrm{clean}}}{\sigma_{c,\mathrm{clean}}}. \tag{87}$$

The primary comparisons are selected-positive versus matched-random concepts, selected-positive versus negatively weighted concepts, and the trend from one to two to three jointly generated positive concepts. Seeds are first aggregated within a concept–target pair, so individual generated images are not treated as independent target-level replicates.

Table 1 shows a clear ordering. Random and negatively weighted concepts produce below-baseline target responses, whereas concepts selected with positive global coefficients raise the target score. Combining the top two or three positive concepts produces a substantially larger response, approaching but remaining below the direct target-name positive control.

J.6 Interpretation and Limitations

The FLUX experiment provides prospective behavioral evidence for the global explanations: concept descriptions are selected from frozen sparse heads before generation, translated into new visual evidence, and then evaluated by frozen models. It therefore tests a stronger property than correlation within the original fitting dataset.

It is nevertheless not a single-factor causal intervention. Text-to-image generation may alter scene composition, lighting, style, object pose, and correlated objects together with the requested concept. The effect of a generated concept can also differ across targets and evaluators. Results should

therefore be interpreted together with the held-out fidelity of the corresponding global readout and with the generation-compliance checks.

J.7 Relationship to Local TTS

Local TTS and the global readout impose sparsity at different levels:

- **Local TTS** selects an input-dependent support $S_\alpha(x)$ whose observed concepts are sufficient to reconstruct one internal representation to a chosen fidelity.
- **Global explanation** selects a target-dependent support S_c whose concepts predict one frozen model output across a dataset.

Neither procedure requires retraining the underlying TTS-AE representation. The dense concept space therefore acts as a reusable substrate from which different sparse explanations can be constructed according to the explanatory question: local reconstruction, targeted intervention, or global output behavior.