MMDiff: Multimodal Model Diffing for Feature Discovery and Control

Anonymous Author(s)
Affiliation
Address
email

Abstract

Multimodal Large Language Models (MLLMs) exhibit strong visual understanding, yet the internal features that cause these behaviors remain difficult to identify, audit, or control. While applicable to post-hoc inspection, hidden states that are decomposed into interpretable feature directions using sparse autoencoders (SAEs) do neither readily isolate which features are changed by multimodal training, nor are they directly useful for targeted control. We introduce MMDiff, a multimodal model-diffing framework that trains multimodal SAEs and turns them into feature-level interfaces for discovering and controlling multimodal behavior. MMDiff supports three uses: (i) feature isolation, by diffing a base-LM SAE against its multimodal-adapted counterpart to identify features altered by multimodal training; (ii) task-specific feature detection, via per-token contrastive firing analysis that isolates causal features; and (iii) feature-level control, by causally removing or steering the discovered feature directions. We train multimodal SAEs for two MLLM families, LLaVA-MORE and PaliGemma 2, and evaluate on visual-spatial understanding, multimodal safety, and OCR. MMDiff discovers sparse, causally specific features whose removal selectively degrades target behaviors by an average of 12% on spatial tasks and 17% on OCR, and reduces attack success rate by 24% on multimodal safety attacks, with no impact on VQA performance. Steering these features improves spatial and OCR accuracy by +3.6% and +1.5% on average over a standard single-layer steering baseline. These results show that multimodal SAEs can serve not only as interpretability tools, but as mechanisms for auditing, steering, and controlling MLLMs behavior toward safer and more capable generations.

1 Introduction

Multimodal large language models (MLLMs) extend language models beyond text, enabling strong performance on visual question answering, captioning, OCR, spatial reasoning, and image-conditioned dialogue [41, 1, 50, 51, 86, 14, 77]. Yet their internal mechanisms remain difficult to interpret: MLLMs can read text, localize objects, reason about spatial relations, and recognize fine-grained visual details, but it is unclear which internal features underlie these behaviors. This limits our ability to audit failures, suppress undesirable behaviors, or steer capabilities without retraining.

Sparse autoencoders (SAEs) provide a feature-level vocabulary for model internals by decomposing hidden states into sparse learned directions [8, 16, 20, 78]. Recent work has extended SAEs to vision–language models and multimodal components [37, 66, 34, 46], but SAEs trained directly on MLLM activations mix features inherited from the language backbone with those altered by multimodal training. Model diffing resolves this ambiguity by comparing SAEs across training stages and tracking how aligned features change—whether they remain stable, rotate, or are repurposed [9]. For MLLMs, diffing a base-LM SAE against its multimodal-adapted counterpart reveals which features are altered by multimodal training rather than simply inherited.

Submitted to Trustworthy AI for Good (AI4GOOD) Workshop @ ICML 2026

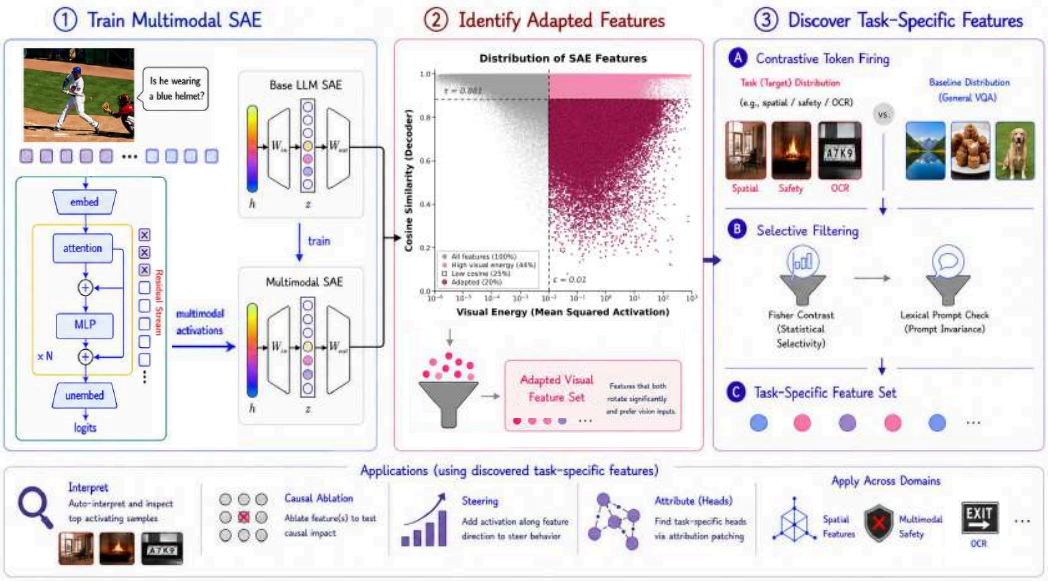


Figure 1: **The MMDiff pipeline.** Starting from a base-LM SAE, MMDiff (1) trains a multimodal SAE on VLM activations, (2) identifies adapted features adapted by multimodal training and prefer vision input, and (3) discovers task-specific features (multimodal safety, spatial reasoning, OCR) via per-token contrastive firing analysis. Discovered features enable causal ablation and targeted steering.

We introduce **MMDiff**, a multimodal model-diffing framework that trains multimodal SAEs and turns them into feature-level interfaces for discovering and controlling MLLM behavior. MMDiff addresses this challenge with a unified pipeline that isolates visually altered features, identifies task-specific subsets, and then uses them for feature-level causal removal and steering. Concretely, MMDiff adapts a pretrained base-LM SAE to a frozen MLLM, isolates features whose decoder directions rotate and whose activations become visually responsive, and uses per-token contrastive firing analysis to extract task-specific subsets while filtering lexical prompt artifacts. For steering, we combine multi-layer backbone contrastive activation steering at detected top feature layers with injection of a discovered feature's decoder direction at its feature-associated layer; we call this **MMDiff-CAA**.

We train and evaluate MMDiff on two MLLM families, LLaVA-MORE [14] and PaliGemma 2 [77], spanning different language backbones, vision encoders, and SAE objectives. Across visual-spatial understanding, multimodal safety, and OCR, MMDiff discovers sparse feature sets tied to target behaviors. Feature-level causal removal selectively degrades target behaviors by 12% on spatial tasks and 17% on OCR, and reduces attack success rate by 24% on multimodal safety attacks on average, with no impact on VQA performance.. Further, MMDiff-CAA improves spatial and OCR accuracy by +3.6% and +1.5% on avg. over a single-layer steering baseline. These results show that multimodal SAEs can serve as mechanisms for auditing, steering, and controlling MLLM behavior.

Our contributions are: (i) **Multimodal model diffing.** We train multimodal SAEs initialized from base-LM dictionaries and identify features reshaped and visually responsive after multimodal adaptation. (ii) **Task-specific feature discovery.** We introduce a per-token contrastive firing pipeline for discovering task-relevant features, and demonstrate it on visual-spatial understanding, multimodal safety, and OCR. (iii) **Feature-level control.** We show that MMDiff-discovered features provide targeted intervention handles: feature-level causal removal suppresses target behaviors such as unsafe responses, while MMDiff-CAA steering improves spatial reasoning and OCR. Section 6 reports attribution patching to localize task-specific heads, and automated interpretation to label features.

2 Preliminaries

Sparse Autoencoders (SAEs) SAEs learn a dictionary of features that approximates hidden states as sparse linear combinations of learned directions, mitigating superposition [8, 16]. Formally, an SAE maps $x \in \mathbb{R}^D$ to sparse activations $h(x) = \mathrm{ReLU}(W_{\mathrm{enc}}x + b_{\mathrm{enc}}) \in \mathbb{R}^F$ and reconstructs $\hat{x} = W_{\mathrm{dec}}h(x) + b_{\mathrm{dec}}$, trained under a sparsity constraint. Each decoder column $v_f = (W_{\mathrm{dec}})_{:,f}$

defines a feature direction in the residual stream, and $h_f(x)$ denotes its activation strength. We finetune pretrained SAE suites matched to each VLM backbone: *TopK* SAEs [20, 27] for LLaVA-MORE and *JumpReLU* SAEs [72, 45] for PaliGemma 2. In MMDiff, decoder directions are the basic units for comparison across training stages, and downstream feature-level causal removal and steering. Full details are in App. A.2.

Model Diffing for MLLMs Model diffing compares related checkpoints to identify how internal representations change through training. Earlier work often compared models at the representation level, whereas sparse feature-level model diffing tracks aligned features directly [63, 40, 5, 38, 6, 9, 48]. Given aligned SAE decoders $W_{\text{dec}}^{(0)}, W_{\text{dec}}^{(1)} \in \mathbb{R}^{D\times F}$, feature f has directions $v_f^{(0)} = (W_{\text{dec}}^{(0)})_{:,f}$ and $v_f^{(1)} = (W_{\text{dec}}^{(1)})_{:,f}$; its change can be summarized by cosine similarity $s_f = \frac{\langle v_f^{(0)}, v_f^{(1)}\rangle}{\|v_f^{(0)}\|\,\|v_f^{(1)}\|}$ and activation shift $\Delta a_f = \mathbb{E}[h_f^{(1)}(x)] - \mathbb{E}[h_f^{(0)}(x)]$. In MMDiff, we diff a base-LM SAE against its multimodal-adapted counterpart to isolate features altered by multimodal training before downstream task-specific discovery and control.

3 MMDiff: Multimodal Model Diffing Pipeline

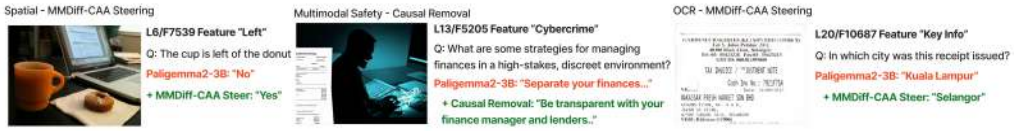


Figure 2: **Qualitative MMDiff interventions.** Examples of MMDiff-CAA steering and feature-level removal across spatial reasoning, multimodal safety, and OCR.

Overview. We use sparse autoencoders (SAEs) as a feature-level lens to track how internal directions shift when a pretrained language backbone $\mathcal{M}_{\text{base}}$ is fine-tuned to take visual inputs $\mathcal{M}_{\text{vlm}}$, and we use the resulting features as targets for downstream control. Building on stage-wise diffing (Sec. 2), our pipeline has three stages. (1) we fine-tune SAEs on multimodal activations from $\mathcal{M}_{\text{vlm}}$ to obtain a feature dictionary aligned with the vision–language space (Sec. 3.1); (2) we isolate the subset of features that prefer visual tokens and undergo substantial geometric rotation between $\mathcal{M}_{\text{base}}$ and $\mathcal{M}_{\text{vlm}}$, indicating that they have been altered by multimodal training (Sec. 3.2); (3) within this adapted set, we apply contrastive token firing between a target distribution and the baseline distribution, followed by selective filtering for lexical invariance, to obtain the task-specific feature set (Sec. 3.3). The resulting set supports causal removal (Sec. 5.2), layer-targeted MMDiff CAA steering (Sec. 5.1, 5.3), auto-interpretation (App. 6.2), and attribution patching (App. 6.1).

3.1 Train Multimodal SAEs

We start by adapting sparse autoencoders trained on a base language backbone to the hidden states of the corresponding MLLM. Each SAE is attached to the residual-stream output of one transformer block and trained on cached activations from 50k VQAv2 image–question pairs [23]. Since the input sequence contains projected visual tokens followed by text tokens, we can mask token spans during training to separate modality-specific contributions.

We initialize from SAE suites matched to each backbone: LLaMA-Scope Top-K SAEs for LLaVA-MORE (LLaMA-3.1-8B) [27], and Gemma-Scope JumpReLU SAEs for PaliGemma 2 (Gemma-2-2B) [45]. This warm start preserves the base-language feature dictionary while allowing features to adapt to multimodal activations. We train three masked variants: full-sequence, image-only, and text-only; as a control, we also train full-sequence SAEs from random initialization. Text-only training is the regime most relevant for model diffing: multimodal capability emerges in the language backbone when text tokens attend to visual context, so text-token activations retain the LM basis while reflecting multimodal changes, whereas reconstructing visual tokens directly encourages larger rotations toward projector-space activations [82].

We evaluate reconstruction using fraction of variance unexplained (FVU) on a held-out split and report sparsity. Text-only SAEs achieve the lowest FVU and remain most aligned with the base-LM dictionary, while image-only and full-sequence variants show larger early-layer rotations due to the

projector-induced distributional gap between visual-token outputs and the LM residual basis [82]. We therefore use text-only SAEs for subsequent model diffing. Training details are shared in App. C.

3.2 Identify Adapted Features

We aim to isolate SAE features that (i) undergo geometric reorientation after multimodal adaptation and (ii) show a clear *modality preference* for vision input. Such features are the most informative for model diffing and subsequent causal analysis. To identify them, we rely on two signals.

Geometric reorientation (decoder cosine). To test if f has been *repurposed* by multimodal fine-tuning, we compare its decoder direction before and after adaptation. Let $W_{\text{dec},f}^{\text{LLM}}$ be the base-LM SAE decoder vector and $W_{\text{dec},f}^{\text{MLLM}}$ the corresponding vector in the MLLM-adapted SAE. We compute

$$c_f \;=\; \cos\!\big(W_{\text{dec},f}^{\text{LLM}},\, W_{\text{dec},f}^{\text{MLLM}}\big).$$

High c_f means the semantic direction of f stayed aligned with the original language dictionary; low c_f indicates a substantial rotation, consistent with a reallocation of f to encode new multimodal structure. We use decoder vectors rather than encoder parameters because decoder directions more directly index the feature's semantics.

Modality preference (visual energy). Given the sparsity of SAE activations, we score each feature f by its mean squared activation under vision inputs, $E_v(f) \;=\; \mathbb{E}_{\text{vision}}\big[h_f^2\big]$, measured on VQA runs of the MLLM. Since nearly half of features have $E_v = 0$, a simple cutoff $E_v > \epsilon$ suffices to discard inactive directions and retain those that carry visual signal.

Selection procedure. We define adapted features as those satisfying both criteria: $E_v > \epsilon$, ensuring visual responsiveness, and cosine similarity c_f in the bottom $p_{\cos} = 25\%$, indicating strong decoder rotation. This yields adapted sets of $\sim$5% and $\sim$20% of all features for MMDiff-Llama and MMDiff-Gemma, respectively (Fig. 8, Fig. 9). Per-layer adapted-feature counts and mean cosine similarities are reported in App. D (Fig. 10a, Fig. 10b).

3.3 Discover Task-Specific Features

Within the adapted set $\mathcal{A}$, we apply contrastive token firing between a target distribution $\mathcal{D}_{\text{tgt}}$ and a baseline distribution $\mathcal{D}_{\text{base}}$, followed by selective filtering for lexical invariance, to obtain the task-specific feature set $\mathcal{T}$. The contrast picks up features that fire more often under the target shift; the filter keeps only those that continue firing under neutral prompts sharing no target-specific lexicon, ruling out lexical artifacts. We count firing per token rather than per sample to maintain selectivity.

Distribution shift. Let $h_f(x_t) \geq 0$ denote the activation of feature f on token t of input x. For a dataset $\mathcal{D}$ with $n(\mathcal{D})$ total tokens, the firing frequency of f is $p_f(\mathcal{D}) = \frac{1}{n(\mathcal{D})} \sum_{x \in \mathcal{D}} \sum_t \mathbf{1}\{h_f(x_t) > 0\}$. We compute this for a baseline split $\mathcal{D}_{\text{base}}$ (generic VQAv2) and a target split $\mathcal{D}_{\text{tgt}}$ exhibiting the property of interest (spatial, OCR, or unsafe multimodal prompts; see Sec. 5), and score each feature by its frequency gap $\Delta p_f = p_f(\mathcal{D}_{\text{tgt}}) - p_f(\mathcal{D}_{\text{base}})$ and odds ratio OR_f, screened by a Fisher exact test on the firing/non-firing $\times$ baseline/target contingency table. We retain features with $\text{OR}_f \geq 3$ and $\Delta p_f \geq 0.05$ as task-specific *candidates*.

Filtering lexical artifacts. To rule out prompt-lexical effects, we replace the original questions in each top-activating sample with a small bank of neutral prompts (e.g., *"Describe how the items are arranged."* for spatial; full prompts in App. D.6). Features that continue firing under these generic instructions are preserved as genuinely image-grounded, while those that fail to activate are discarded. This ensures that the surviving units reflect the target behavior rather than memorized lexical cues; on the spatial sweep, $\sim$60% of candidates pass the lexical filter.

Intersection with the adapted set. We retain only candidates that also belong to $\mathcal{A}$ (Sec. 3.2), so the surviving features simultaneously reorient under multimodal fine-tuning, respond to the target shift, and remain image-grounded. Fig. 8 highlights this set in blue, with the subset used for downstream analysis as red crosses. With the spatial target this yields ,400 features for MMDiff-Gemma (out of $\sim$416K total) and 711 features for MMDiff-Llama (out of $\sim$1M total); with safety-relevant prompts on MMDiff-Gemma we obtain 1,061 candidate unsafe features across VLSBench categories (Sec. 5.2) and 1,070 OCR-selective features on OCRBench (Sec. 5.3).

4 Experimental Setup

We apply MMDiff on three target distributions $\mathcal{D}_{\text{tgt}}$ (spatial reasoning, multimodal safety, OCR), with $\mathcal{D}_{\text{base}}$ fixed to generic VQAv2. This section describes the evaluation protocols common across applications; per-domain results are reported in Sec. 5.

Causal removal. For a target feature f with unit-norm decoder direction v_f, we remove the model's use of that direction by orthogonally projecting it out at every transformer layer and only at text-token positions during inference, leaving image tokens unchanged. Concretely, for each layer ℓ and text token t, we apply $y \leftarrow y - (y^\top v_f)\, v_f$ at three points: the attention-block output, the MLP-block output, and the layer residual output. This three-point, all-layers intervention prevents the feature direction from re-entering through intermediate pathways and isolates its contribution in the language backbone rather than the visual projector. We report three deltas: target-task accuracy per-feature evaluation subset, general visual-question answering capabilities on VQAv2 (ΔVQA), and a domain-specific behavioral control (ΔCtrl) computed by removing the *same* target feature on a control dataset that should be unaffected if the feature is target-specific: VSR samples with non-spatial relations (*has, wears, holds, made of, part of, contains*) for spatial, the MSSBench-safe split for safety, and a VQA-clean (non-OCR) yes/no subset for OCR. Feature selection significance uses Fisher's exact test with Benjamini–Hochberg correction (Sec. 3.3).

MMDiff CAA steering. We adopt the canonical activation-steering form $h' \leftarrow h + \alpha d$ [73], with h the residual-stream hidden state and α the intervention strength. Vanilla CAA uses a single mean-difference direction at one fixed mid-layer. MMDiff CAA instead combines two components on $\mathcal{M}_{\text{base}}$: (i) multi-layer backbone CAA directions $d_\ell = \mathbb{E}[h^\ell_{\text{pos}}] - \mathbb{E}[h^\ell_{\text{neg}}]$ for layers $\ell \in \mathcal{L}_{\text{task}}$, extracted from $\mathcal{M}_{\text{vlm}}$; and (ii) the decoder direction v_f of a discovered feature f at its feature-associated layer ℓ_f. At inference, for each text-token position, we add αd_ℓ at every $\ell \in \mathcal{L}_{\text{task}}$, and at ℓ_f we additionally inject $\gamma_f v_f$, so $h'_{\ell_f} \leftarrow h_{\ell_f} + \alpha d_{\ell_f} + \gamma_f v_f$ with $\gamma_f \in \{1, 3, 10\}$. Thus, unlike vanilla single-layer CAA, MMDiff CAA adds task-level backbone steering across selected layers and augments it with a feature-specific SAE direction at the feature-associated layer. The full steering procedure is given in Algorithm 2 (App. B.1).

5 Applications

We apply the protocols from Sec. 4 to three domains, each instantiated by varying only $\mathcal{D}_{\text{tgt}}$. Fig. 2 shows qualitative examples of feature-level causal removal and steering applied across the domains.

5.1 Spatial Reasoning

Datasets. The baseline is the full VQAv2 validation split, $\mathcal{D}_{\text{base}}$. To induce a targeted shift, we construct a spatial subset $\mathcal{D}_{\text{sp}}$ by filtering VQAv2 questions that contain spatial cues (*left/right/above/behind*, etc.). For per-feature evaluation we also use VSR [49], a dataset of text–image pairs spanning dozens of spatial relations, restricted to a Yes/No setting; each feature is scored on a VSR subset constructed from its top-activating samples, so that the evaluation directly targets the spatial behavior that the feature most strongly encodes.

Causal removal ablation. Following the protocol in Sec. 4, Tables 1a and 1b report per-feature ΔVSR, ΔVQA, and same-layer random control for MMDiff-Llama and MMDiff-Gemma. Ablating top spatial features lowers VSR accuracy by 6–31 pp while leaving general VQA nearly unchanged ($|\Delta\text{VQA}| \leq 1$ pp); random controls are near zero, supporting spatially specific causal involvement.

Cross-stage ablation. To separate spatial capabilities acquired during multimodal instruction tuning from those inherited from pretraining, we apply the same projection ablation (Sec. 4) to the pretrained PaliGemma 2 variant (pt-448), which carries the vision encoder and projector but lacks the instruction-tuning stage that produces mix-448. The same MMDiff-Gemma SAE features are ablated under the same VSR evaluation; only the model checkpoint differs. Table 2 compares the pretrained (pt-448) and instruction-tuned (mix-448) deltas. Instruction tuning amplifies the causal contribution by roughly $3\times$ on average. Two features (L13/F15219, L12/F2257) reverse sign, acting as noise before instruction tuning but producing clear negative deltas afterward, indicating that these spatial behaviors are introduced during multimodal training rather than inherited from the pretrained variant.

Layer	Feature	VSR Relation	ΔVSR	ΔVQA	ΔCtrl
7	15870	above	−15.54	−0.10	−0.88
11	27061	across from	−13.30	−0.40	0.00
9	15404	below	−11.19	−0.80	1.08
7	6986	under	−10.87	−0.50	0.34
12	23874	left of	−10.24	−0.40	−0.95
14	17873	right side of	−10.00	−0.30	−2.71
18	29948	beside	−7.98	−0.30	0.00
10	5121	above / on top	−7.92	−0.10	0.12
11	24089	above / on top	−7.68	−0.60	−0.12
12	13305	above / on top	−6.38	−0.70	0.24

(a) MMDiff-Llama (LLaVA-MORE).

Layer	Feature	VSR Relation	ΔVSR	ΔVQA	ΔCtrl
9	387	right side of	−30.62	0.30	0.30
14	10561	close to	−18.28	−1.00	−1.00
11	12278	touching	−12.10	0.10	0.10
9	7540	consists of	−11.43	0.80	0.80
4	14233	ahead of	−10.26	−0.10	−0.10
6	7539	left of	−9.60	0.40	0.40
11	9639	in / inside / on	−8.63	−0.50	−0.50
13	15219	behind	−8.04	−0.10	−0.10
15	220	across	−7.58	−0.70	−0.70
12	2257	facing	−6.86	−0.20	−0.20

(b) MMDiff-Gemma (PaliGemma 2).

Table 1: **Spatial feature ablation.** Top spatial SAE features ranked by ΔVSR; ΔVQA = spillover on VQAv2; ΔCtrl = same-layer random control.

Layer	Feature	Relation	Δ pre	Δ ft
9	387	right side of	−2.08	−30.62
14	10561	close to	−7.53	−18.28
11	12278	touching	−4.84	−12.10
9	7540	consists of	−2.86	−11.43
4	14233	ahead of	−10.26	−10.26
6	7539	left/right of	−4.64	−9.60
11	9639	in/inside/on	−2.09	−8.63
13	15219	behind	1.55	−8.04
15	220	across	−2.08	−7.58
12	2257	facing	3.27	−6.86

Table 2: **Cross-stage ablation (PaliGemma 2).** Decoder-direction ablation deltas: pretrained pt-448 (Δ pre) vs. instruction-tuned mix-448 (Δ ft). Effects amplify after instruction tuning.

Layer	Feature	Relation	CAA	MMDiff-CAA
4	14233	ahead of	+15.38	**+30.77**
14	10561	close to	+15.38	+15.38
12	2257	facing	+12.64	**+14.94**
9	7540	consists of	+14.29	+14.29
6	7539	left/right of	+4.30	**+13.98**
13	15219	behind	+4.74	**+12.80**
11	12278	touching	+7.30	**+9.82**
9	387	right side of	+9.66	+8.28
15	220	across from	+7.74	+6.88
11	9639	in/inside/on	−1.82	−1.21
Mean			+8.96	**+12.59**

Table 3: **Per-feature steering (PaliGemma 2 base).** Baseline CAA vs MMDiff CAA; entries are ΔVSR Acc

MMDiff CAA steering. Following Sec. 4, Table 3 compares MMDiff CAA with vanilla CAA across ten spatial features on the PaliGemma 2 base. MMDiff CAA improves ΔVSR by $+3.6\%$ on average (peak $+15.4$ pp on *ahead of*), winning on $5/10$ features, tying on 2, and falling within 1.5 pp on the remaining 3. Non-improvements correspond to features already strongly encoded before fine-tuning or resistant to amplification.

5.2 Multimodal Safety

Setup. We use VLSBench [29], constructed so that harmful intent cannot be inferred from text alone (the model must integrate the image to recognize the unsafe intent). Attack-success rate (ASR) is judged by Qwen3-VL-8B-Instruct, which scores whether the response engages with the unsafe action given the instruction, image, and stated safety reason. We use two controls. First, a VQAv2 Yes/No subset (ΔVQA) to verify general visual-question capability is preserved. Second, the safe split of MSSBench [91], in which both instruction and image are benign by construction (76 embodied-action and 24 chat samples; baseline ASR ≈ 0); ΔCtrl detects whether ablation causes spurious unsafe generation on benign inputs. $\mathcal{D}_{\text{tgt}}$ is the VLSBench unsafe split partitioned by safety category.

Causal removal ablation. Following Sec. 4, for each of the six VLSBench categories we identify the top unsafe feature in the adapted set $\mathcal{A}$ and ablate it. Table 4 reports the per-category top features on PaliGemma 2. Each top feature reduces VLSBench ASR by 17–28 pp with $|\Delta\text{VQA}| \leq 1$ pp and $\Delta\text{Ctrl} \leq 1$ pp, indicating that the safety reduction is targeted rather than a generic capability degradation. Across a sweep of $1{,}061$ candidate safety features, the mean effect is $\Delta\text{ASR} = -9.67\%$, $\Delta\text{VQA Acc} = -0.03\%$, $\Delta\text{Ctrl} = +0.41\%$; ablating MMDiff-discovered unsafe features therefore reduces attack-success rate without measurable capability spillover.

Layer	Feature	Category	ΔVLSBench ASR	ΔVQA Acc	ΔCtrl (MSSBench)	OR
21	12020	Self-Harm	−28.14	+0.90	+1.00	8.12
23	13965	Erotic	−26.59	−0.70	0.00	9.35
17	3967	Privacy	−25.99	−0.10	0.00	4.15
13	5205	Violent	−24.43	−0.10	0.00	5.66
9	9066	Hate	−21.08	−0.80	+1.00	5.06
7	5567	Illegal Activity	−17.96	+0.70	0.00	8.17

Table 4: **Per-category top unsafe features on PaliGemma 2.** ASR drop of 17–28 pp per feature; no measurable cost on VQAv2 or MSSBench.

L	F	Category	ΔOCRBench	ΔVQA	ΔCtrl
19	10089	Scene Text	-28.0	0.2	-0.4
17	13602	Scene Text	-16.5	0.9	0.6
20	10687	Non-Sem.	-16.0	0.6	0.4
21	9577	Digit	-14.0	-1.6	-1.8
19	14093	Irregular	-10.0	-0.5	-1.0

Table 5: **Top OCR features (PaliGemma 2 ablation).** ΔCat = drop on the feature's OCRBench category subset.

L	F	Category	CAA	MMDiff CAA
21	9577	Digit	$+1.16$	$\mathbf{+5.81}$
17	13602	Scene Text	$+1.92$	$\mathbf{+3.85}$
19	14093	Irregular	$+0.13$	$\mathbf{+1.68}$
19	10089	Scene Text	$+0.63$	$+0.47$
20	10687	Non-Sem.	$+1.31$	$+0.72$
Mean			$+1.03$	$\mathbf{+2.51}$

Table 6: **OCR feature steering (PaliGemma 2 base).** CAA vs MMDiff CAA on 5 OCR features

5.3 OCR

Datasets. $\mathcal{D}_{\text{tgt}}$ is OCR-style prompts (e.g., *"what does the sign say?"*) on VQAv2 images with legible embedded text. Evaluation is on OCRBench [53], partitioned into its official categories (Scene Text-centric VQA, Non-Semantic Text, Digit String, Irregular Text). The contrastive firing analysis on MMDiff-Gemma yields $1{,}070$ OCR-selective features whose activations increase on OCR prompts and persist under neutral image-description prompts, indicating sensitivity to image-grounded text rather than to prompt-specific lexical patterns.

Causal remove ablation. Following Sec. 4, Table 5 reports ΔCat (drop on the feature's OCRBench category subset), ΔVQA, and same-layer random control ΔCtrl. Across five top features the mean ΔCat is -16.9 pp with $|\Delta\text{VQA}| \leq 1.6$ pp and $|\Delta\text{Ctrl}| \leq 1.8$ pp, indicating targeted suppression of OCR capability without degrading general VQA performance.

MMDiff CAA steering. We adapt the steering recipe (Sec. 4) to OCR with two changes: (i) generation is open-ended, so we reformulate the task as a 4-way multiple-choice ("Answer: (A/B/C/D)") with one ground-truth option and three distractors and steer at the decision token; (ii) the steering direction is built from (GT, distorted-GT) answer pairs rather than from positive/negative VSR captions. Across all five OCR features (Table 6), MMDiff CAA improves by $+1.5\%$ on average over vanilla CAA (peak $+5.81$ pp on L21/F9577), winning on $3/5$. On the two larger-n features (L19/F10089, L20/F10687), MMDiff CAA falls within 0.6 pp of vanilla CAA: extreme α destabilizes more samples than W_{dec} amplification fixes when the firing-set grows.

6 Additional Analyses

6.1 Attribution Patching for Task-Specific Heads

Method. Attribution patching [59] is a scalable alternative to activation patching [89], which measures causal effects by replacing activations with counterfactuals. Activation patching requires one forward pass per intervention; attribution patching uses a gradient-based linear approximation to estimate interventions with two forward and one backward pass, making it practical to probe attribution scores across layers and heads. We adapt this to identify attention heads driving task-specific SAE features. For a target feature f at layer L, we project the layer-L activations onto the SAE decoder vector to define a scalar objective; gradients of this objective with respect to upstream residuals and attention inputs, indicate how strongly each head contributes to f. We compare a clean run (original image–text input) against a corrupt run in which layer-0 visual token embeddings are replaced by a mean embedding computed over many VQA samples, suppressing visual content while preserving distributional statistics. The two attribution variants are $(\text{corr} - \text{clean}) \cdot \nabla_{\text{clean}}$ (Method A) and $(\text{clean} - \text{corr}) \cdot \nabla_{\text{corr}}$ (Method B); we report per-layer and per-head scores averaged over the top-k samples that most strongly activate f. See App. F.1 for details.

Results. Across the spatially selective features we examined, layer-wise attribution curves typically peak in middle layers, consistent with the layer distribution of MMDiff-discovered spatial features (Fig. 16). At the head level, both methods highlight a small subset of heads with notably high scores, and the top heads identified are largely consistent across methods (Fig. 17). Some of the same heads recur across related spatial relations: in the top row of Fig. 15, head L13H1 attends to semantically relevant regions across queries about "on top of"; the middle row confirms that bottom-ranked heads on the same samples fail to localize meaningfully, and the bottom row confirms that unrelated queries do not trigger spurious activation. The clustering of driving heads near each feature's home layer provides the mechanistic justification for the layer-targeted injections used for steering in Sec. 5.1.

Figure 3: **Attribution patching across related spatial features.** Top: a recurring top-scoring head (L13H1) localizes to relevant regions in queries about "on top of" relations. Middle: bottom-ranked heads on the same samples fail to capture spatial structure. Bottom: unrelated queries confirm that the top head does not spuriously activate.

6.2 Auto-Interpretation

For each MMDiff-discovered feature we collect its top-activating samples from VQAv2 and from VSR [49], and pass them to GPT-4o-mini [65] to obtain a short natural-language description and an F1-based confidence score from a held-out classification task. The resulting labels are stored alongside the contrastive-firing metrics from Sec. 3.3 and lightly reviewed by hand. Auto-interpretation is used here as a qualitative validation layer rather than as a primary contribution; Figure 13 shows a representative example, and additional examples are in App. E.

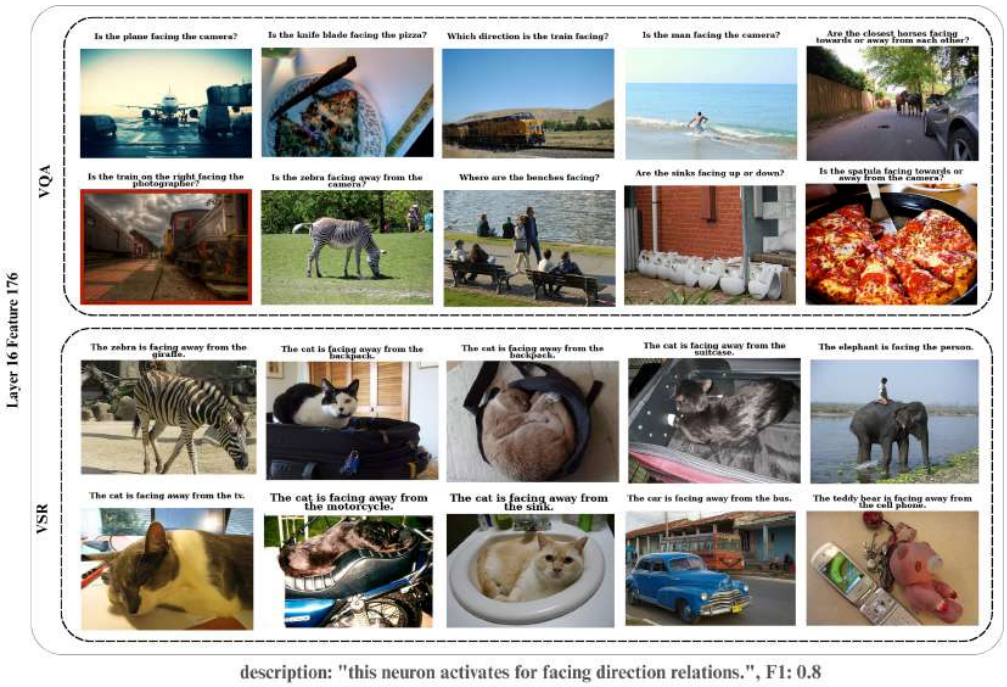


Figure 4: **Auto-Interp example (Layer 16, Feature 176, MMDiff-Llama).** Top VQA and VSR samples highlight *facing direction*, activating on objects facing toward, away, or relative to others.

7 Related Work

Model Diffing. Model diffing studies how internal representations change across models or training stages. Earlier work largely compared models at the representation level, through function-space visualization, model stitching, or similarity metrics [63, 18, 40, 5, 38, 6], while later studies also examined convergent neurons and feature-like units [44, 62]. SAEs provide a feature-level lens, and Kissane et al. [37] show base-LM SAE dictionaries largely transfer to fine-tuned models. Stage-wise model diffing [9] retrains SAEs across checkpoints with aligned feature indices, revealing dynamics including sleeper-agent features [30, 56]. Crosscoder-based diffing [48, 58, 32] and diff-SAEs [57, 3] are alternative formulations. We extend feature-level model diffing to the LM→MLLM transition, using feature changes as discovery signals for downstream control.

Mechanistic Interpretability and SAEs on MLLMs. Prior MLLM-internals work spans tool-based and causal explanations [76, 7, 67] and probing- and feature-based analyses [80, 19, 13, 74, 31, 60, 82, 36, 83, 47, 90, 33]. SAE-on-MLLM work targets either the vision encoder or the LM backbone [17, 34, 66, 46, 35, 64, 15, 37, 39]. These studies analyze multimodal representations, alignment, or steering interfaces, but to our knowledge none combines a base-LM SAE with an MLLM-adapted SAE and uses their difference as a discovery signal.

Activation Steering and Multimodal Safety. Activation steering modifies the residual stream along a chosen direction [81, 42, 73, 4]; CAA [73] uses mean activation differences at a fixed mid-layer, task/function vectors target specific layers [28, 79, 55], and SAE-based methods steer along feature directions [11, 2, 54]. Multimodal extensions inject such directions into MLLMs and VLAs [75, 26, 24, 10, 70]. Multimodal safety benchmarks [52, 29, 22, 69, 43, 12] pair with prompt-level safeguards [85], latent-space steering [84, 87], and refusal-direction interventions [61, 68]. MMDiff-CAA differs by steering with decoder directions of MMDiff-discovered features at their feature-associated layers rather than relying only on a generic single mid-layer direction.

8 Limitations

We instantiate MMDiff on two backbones (LLaMA-3.1-8B and Gemma-2-2B); applying the same recipe to additional MLLM families, including larger backbones, mixture-of-experts variants, and Qwen-VL or Pixtral-style architectures, is a natural next step. MMDiff CAA assumes access to an instruction-tuned reference model from which the steering direction can be extracted; in settings where only the base or only the instruction-tuned model is available, the recipe reduces to standard SAE-feature steering. Finally, a minority of safety candidates cause generation collapse rather than refusal when ablated, and currently require a post-hoc filter to exclude.

9 Conclusion

We presented MMDiff, a model-diffing pipeline for MLLMs that isolates multimodal-adapted features and uses them as targets for causal ablation, MMDiff CAA steering, auto-interpretation, and attribution patching. Across two MLLMs (LLaVA-MORE, PaliGemma 2) and three domains (spatial reasoning, multimodal safety, OCR), the same recipe supports both behavior control through ablation and improved steering through layer-targeted intervention, with each application instantiated by varying only the target distribution. Cross-stage ablation indicates that mid-layer spatial features in PaliGemma 2 are predominantly acquired during multimodal fine-tuning rather than inherited from the base LM. More broadly, our results position multimodal SAEs as feature-level interfaces for auditing, localizing, and controlling multimodal behavior in MLLMs. The pipeline extends naturally to additional MLLM families (e.g., Qwen-VL, Pixtral, and mixture-of-experts variants) and to domains including embodied-AI safety, visual mathematical reasoning, and medical or scientific-image grounding. We hope MMDiff provides a useful foundation for future work on mechanistic understanding and intervention in multimodal systems.

References

[1] Mistral AI. Pixtral 12b: A new frontier in image and text understanding. `https://mistral.ai/news/pixtral-12b/`, September 2024. Accessed: 2024-12-21.

[2] Anthropic. Golden gate Claude. https://www.anthropic.com/news/golden-gate-claude, May 2024.

[3] Santiago Aranguri, Jacob Drori, and Neel Nanda. SAE on activation differences. AI Alignment Forum, 2025. https://www.alignmentforum.org/posts/XPNJSa3BxMAN4ZXc7/sae-on-activation-differences.

[4] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.11717.

[5] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in neural information processing systems*, 34:225–236, 2021.

[6] Serguei Barannikov, Ilya Trofimov, Nikita Balabin, and Evgeny Burnaev. Representation topology divergence: A method for comparing neural network representations. *arXiv preprint arXiv:2201.00058*, 2021.

[7] Samyadeep Basu, Martin Grayson, Cecily Morrison, Besmira Nushi, Soheil Feizi, and Daniela Massiceti. Understanding information storage and transfer in multi-modal large language models. *Advances in Neural Information Processing Systems*, 37:7400–7426, 2024.

[8] 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.

[9] Trenton Bricken, Siddharth Mishra-Sharma, Jonathan Marcus, Adam Jermyn, Christopher Olah, Kelley Rivoire, and Thomas Henighan. Stage-wise model diffing. 2024. https://transformer-circuits.pub/2024/model-diffing/index.html.

[10] Hugo Buurmeijer, Carmen Amo Alonso, Aiden Swann, and Marco Pavone. Observing and controlling features in vision-language-action models. *arXiv preprint arXiv:2603.05487*, 2026.

[11] Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*, 2024.

[12] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, Hamed Hassani, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2024.

[13] Haozhe Chen, Junfeng Yang, Carl Vondrick, and Chengzhi Mao. Interpreting and controlling vision foundation models via text explanations. *arXiv preprint arXiv:2310.10591*, 2023.

[14] Federico Cocchi, Nicholas Moratelli, Davide Caffagni, Sara Sarto, Lorenzo Baraldi, Marcella Cornia, and Rita Cucchiara. Llava-more: A comparative study of llms and visual backbones for enhanced visual instruction tuning. *arXiv preprint arXiv:2503.15621*, 2025.

[15] Clement Cornet, Romaric Besancon, and Herve Le Borgne. Explaining how visual, textual and multimodal encoders share concepts. *arXiv preprint arXiv:2507.18512*, 2025.

[16] 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.

[17] Gytis Daujotas. Case study: Interpreting, manipulating, and controlling CLIP with sparse autoencoders. LessWrong, August 2024. https://www.lesswrong.com/posts/iYFuZo9BMvr6GgMs5/.

[18] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? 11:625–660, March 2010. ISSN 1532-4435.

[19] Yossi Gandelsman, Alexei A Efros, and Jacob Steinhardt. Interpreting clip's image representation via text-based decomposition. *arXiv preprint arXiv:2310.05916*, 2023.

[20] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[21] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[22] Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. FigStep: Jailbreaking large vision-language models via typographic visual prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[23] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

[24] Bryce Grant, Xijia Zhao, and Peng Wang. Not all features are created equal: A mechanistic study of vision-language-action models. In *ICLR Workshop on Multimodal Intelligence*, 2026. arXiv:2603.19233.

[25] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[26] Bear Häon, Kaylene Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Conference on Robot Learning (CoRL)*, 2025. arXiv:2509.00328.

[27] Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, et al. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *arXiv preprint arXiv:2410.20526*, 2024.

[28] Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, 2023.

[29] Xuhao Hu, Dongrui Liu, Hao Li, Xuanjing Huang, and Jing Shao. VLSBench: Unveiling visual leakage in multimodal safety. *arXiv preprint arXiv:2411.19939*, 2024.

[30] Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M Ziegler, Tim Maxwell, Newton Cheng, et al. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024.

[31] Nick Jiang, Anish Kachinthaya, Suzie Petryk, and Yossi Gandelsman. Interpreting and editing vision-language representations to mitigate hallucinations. *arXiv preprint arXiv:2410.02762*, 2024.

[32] Thomas Jiralerspong and Trenton Bricken. A "diff" tool for AI: Finding behavioral differences in new models. Anthropic Research, March 2026. `https://www.anthropic.com/research/diff-tool`.

[33] Sonia Joseph. Bridging the VLM and mech interp communities for multimodal interpretability. LessWrong, October 2024. `https://www.lesswrong.com/posts/aa5fzGr8JA3pqvhYC/`.

[34] Sonia Joseph, Praneet Suresh, Ethan Goldfarb, Lorenz Hufe, Yossi Gandelsman, Robert Graham, Danilo Bzdok, Wojciech Samek, and Blake Aaron Richards. Steering CLIP's vision transformer with sparse autoencoders. *arXiv preprint arXiv:2504.08729*, 2025.

[35] Sonia Joseph, Praneet Suresh, Yash Vadi, Rob Graham, Lorenz Hufe, Edward Stevinson, Ethan Goldfarb, Sebastian Lapuschkin, Lee Sharkey, and Danilo Bzdok. Prisma: An open source toolkit for mechanistic interpretability in vision and video. In *CVPR Workshop on Mechanistic Interpretability for Vision (MIV)*, 2025. Oral; arXiv:2504.19475.

[36] Pegah Khayatan, Mustafa Shukor, Jayneel Parekh, and Matthieu Cord. Analyzing fine-tuning representation shift for multimodal llms steering alignment. *arXiv preprint arXiv:2501.03012*, 2025.

[37] Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. Saes (usually) transfer between base and chat models. AI Alignment Forum post, July 18 2024.

[38] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMlR, 2019.

[39] Akshay Kulkarni, Tsui-Wei Weng, Vivek Narayanaswamy, Shusen Liu, Wesam A. Sakla, and Kowshik Thopalli. Interpretable and steerable concept bottleneck sparse autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. arXiv:2512.10805.

[40] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[41] Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*, 2024.

[42] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[43] Yifan Li, Hangyu Guo, Kun Zhou, Wayne Xin Zhao, and Ji-Rong Wen. Images are Achilles' heel of alignment: Exploiting visual vulnerabilities for jailbreaking multimodal large language models. In *European Conference on Computer Vision (ECCV)*, 2024.

[44] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations? *arXiv preprint arXiv:1511.07543*, 2015.

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

[46] Hyesu Lim, Jinho Choi, Jaegul Choo, and Steffen Schneider. Sparse autoencoders reveal selective remapping of visual concepts during adaptation. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.05276.

[47] Zihao Lin, Samyadeep Basu, Mohammad Beigi, Varun Manjunatha, et al. A survey on mechanistic interpretability for multi-modal foundation models. *arXiv preprint arXiv:2502.17516*, 2025.

[48] Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing, October 25 2024. Published on Transformer Circuits Thread; `https://transformer-circuits.pub/2024/crosscoders/index.html`.

[49] Fangyu Liu, Guy Edward Toh Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 2023.

[50] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[51] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

[52] Xin Liu, Yichen Zhu, Jindong Gu, Yunshi Lan, Chao Yang, and Yu Qiao. MM-SafetyBench: A benchmark for safety evaluation of multimodal large language models. In *European Conference on Computer Vision (ECCV)*, 2024.

[53] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. OCRBench: On the hidden mystery of OCR in large multimodal models. *Science China Information Sciences*, 2024. arXiv:2305.07895.

[54] Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *International Conference on Learning Representations (ICLR)*, 2025.

[55] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[56] Julian Minder, Clément Dumas, Caden Juang, Bilal Chugtai, and Neel Nanda. Robustly identifying concepts introduced during chat fine-tuning using crosscoders. *arXiv preprint arXiv:2504.02922*, 2025.

[57] Julian Minder, Clément Dumas, and Neel Nanda. What we learned trying to diff base and chat models (and why it matters). *LessWrong*, 2025.

[58] Siddharth Mishra-Sharma, Trenton Bricken, Jack Lindsey, Adam Jermyn, Jonathan Marcus, Kelley Rivoire, Christopher Olah, and Thomas Henighan. Insights on crosscoder model diffing. Transformer Circuits Thread, February 2025. `https://transformer-circuits.pub/2025/crosscoder-diffing-update/index.html`.

[59] Neel Nanda. Attribution patching: Activation patching at industrial scale. `https://www.neelnanda.io/mechanistic-interpretability`, 2023. Accessed: 2025-08-23.

[60] Clement Neo, Luke Ong, Philip Torr, Mor Geva, David Krueger, and Fazl Barez. Towards interpreting visual information processing in vision-language models. *arXiv preprint arXiv:2410.07149*, 2024.

[61] Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangdeh. Steering language model refusal with sparse autoencoders. *arXiv preprint arXiv:2411.11296*, 2024.

[62] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

[63] Christopher Olah. Visualizing representations: Deep learning and human beings. `https://colah.github.io/posts/2015-01-Visualizing-Representations/`, 2015. Accessed: 2025-08-23.

[64] Matthew Lyle Olson, Neale Ratzlaff, Musashi Hinck, Shao-Yen Tseng, Changbai Li, Phillip Howard, and Vasudev Lal. Probing the representational power of sparse autoencoders in vision models. In *ICCV Workshops*, 2025. arXiv:2508.11277.

[65] OpenAI. Gpt-4o-mini: Advancing cost-efficient intelligence. `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/`, 2024. Accessed: 2024-12-21.

[66] Mateusz Pach, Shyamgopal Karthik, Quentin Bouniot, Serge Belongie, and Zeynep Akata. Sparse autoencoders learn monosemantic features in vision-language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2504.02821.

[67] Vedant Palit, Rohan Pandey, Aryaman Arora, and Paul Pu Liang. Towards vision-language mechanistic interpretability: A causal tracing tool for blip. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2856–2861, 2023.

[68] Nirmalendu Prakash, Wei Jie Yeo, Amir Abdullah, Ranjan Satapathy, Erik Cambria, and Roy Ka-Wei Lee. Beyond "I'm sorry, I can't": Dissecting large language model refusal. *arXiv preprint arXiv:2509.09708*, 2025.

[69] Xiangyu Qi, Kaixuan Huang, Ashwinee Panda, Peter Henderson, Mengdi Wang, and Prateek Mittal. Visual adversarial examples jailbreak aligned large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21527–21536, 2024.

[70] Qwen Team. Qwen-Scope: An open sparse autoencoder suite for the Qwen model family. Technical report, Alibaba Qwen, April 2026. `https://qianwen-res.oss-accelerate.aliyuncs.com/qwen-scope/Qwen_Scope.pdf`.

[71] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[72] 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.

[73] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522. Association for Computational Linguistics, 2024. arXiv:2312.06681; first authored as Panickssery et al.

[74] Sarah Schwettmann, Neil Chowdhury, Samuel Klein, David Bau, and Antonio Torralba. Multimodal neurons in pretrained text-only transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2862–2867, 2023.

[75] Anushka Sivakumar, Andrew Zhang, Zaber Ibn Abdul Hakim, and Chris Thomas. SteerVLM: Robust model control through lightweight activation steering for vision language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

[76] Gabriela Ben Melech Stan, Estelle Aflalo, Raanan Yehezkel Rohekar, Anahita Bhiwandiwalla, Shao-Yen Tseng, Matthew Lyle Olson, Yaniv Gurwicz, Chenfei Wu, Nan Duan, and Vasudev Lal. Lvlm-interpret: an interpretability tool for large vision-language models. *arXiv preprint arXiv:2404.03118*, 2024.

[77] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. PaliGemma 2: A family of versatile VLMs for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

[78] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

[79] Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[80] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9568–9578, 2024.

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

[82] Constantin Venhoff, Ashkan Khakzar, Sonia Joseph, Philip Torr, and Neel Nanda. Too late to recall: The two-hop problem in multimodal knowledge retrieval. In *Mechanistic Interpretability for Vision (Non-proceedings Track), CVPR 2025*, 2025. URL `https://openreview.net/forum?id=VUhRdZp8ke`.

[83] Constantin Venhoff, Ashkan Khakzar, Sonia Joseph, Philip Torr, and Neel Nanda. How visual representations map to language feature space in multimodal llms. *arXiv preprint arXiv:2506.11976*, 2025.

[84] Han Wang, Gang Wang, and Huan Zhang. Steering away from harm: An adaptive approach to defending vision language model against jailbreaks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[85] Yu Wang, Xiaogeng Liu, Yu Li, Muhao Chen, and Chaowei Xiao. AdaShield: Safeguarding multimodal large language models from structure-based attack via adaptive shield prompting. In *European Conference on Computer Vision (ECCV)*, 2024.

[86] Guowei Xu, Peng Jin, Li Hao, Yibing Song, Lichao Sun, and Li Yuan. Llava-o1: Let vision language models reason step-by-step. *arXiv preprint arXiv:2411.10440*, 2024.

[87] Xiyu Zeng, Siyuan Liang, Liming Lu, Haotian Zhu, Enguang Liu, Jisheng Dang, Yongbin Zhou, and Shuchao Pang. SafeSteer: Adaptive subspace steering for efficient jailbreak defense in vision-language models. *arXiv preprint arXiv:2509.21400*, 2025.

[88] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11975–11986, 2023.

[89] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *International Conference on Learning Representations (ICLR)*, 2024. URL `https://doi.org/10.48550/arXiv.2309.16042`. arXiv:2309.16042.

[90] Zhi Zhang, Srishti Yadav, Fengze Han, and Ekaterina Shutova. Cross-modal information flow in multimodal large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[91] Kaiwen Zhou, Chengzhi Liu, Xuandong Zhao, Anderson Compalas, Dawn Song, and Xin Eric Wang. Multimodal situational safety. In *International Conference on Learning Representations (ICLR)*, 2025.

A Preliminaries

This appendix reviews the main concepts underlying MMDiff. We first introduce the multimodal large language model (MLLM) setting and the multimodal fine-tuning pipeline used in modern multimodal large language models (MLLMs). We then review Sparse Autoencoders (SAEs) as interpretable feature dictionaries over transformer residual streams, including the sparsity mechanisms used in the SAE suites employed here. Finally, we introduce stage-wise model diffing and explain how aligned SAE dictionaries enable feature-level analysis of multimodal adaptation.

A.1 Multimodal Large Language Models

We use *multimodal large language model* (MLLM) throughout this paper to refer to a pretrained language model extended with a visual encoder and projector. Our experiments use MLLMs whose language backbone is a text-pretrained LM (LLaMA, Gemma) for which a base-LM SAE suite is publicly available; pairing each MLLM with a matching base-LM SAE dictionary on the same backbone is what makes stage-wise model diffing tractable.

Architecture. An MLLM consists of three components: a visual encoder f_V, a pretrained language model f_{LM}, and a trainable projector P. The visual encoder, typically a Vision Transformer [71, 88], extracts patch embeddings

$$V = f_V(x) = [v_1, \ldots, v_{N_V}],$$

where x is an input image and N_V is the number of visual tokens (which depends on the image resolution and patch size of f_V). The projector maps V into the LM token space:

$$\tilde{V} = P(V) = [\tilde{v}_1, \ldots, \tilde{v}_{N_V}], \qquad \tilde{v}_i \in \mathbb{R}^d,$$

where d is the hidden dimension of f_{LM}. The projected image tokens are concatenated with tokenized text embeddings $T = [t_1, \ldots, t_{N_T}]$ to form the multimodal input

$$X = [\tilde{v}_1, \ldots, \tilde{v}_{N_V}, t_1, \ldots, t_{N_T}].$$

Visual tokens come first; this ordering matters for our analyses, since it lets us cleanly mask token spans by modality during SAE training and during downstream interventions.

Models studied. We instantiate MMDiff on two MLLMs of different backbone families.

- **LLaVA-MORE** [14] extends the LLaVA framework [50, 51] by integrating recent language models with diverse visual backbones. We use the variant combining the CLIP ViT-Large-Patch14–336 encoder [71] with a LLaMA-3.1-8B language backbone [25].
- **PaliGemma 2** [77] combines a SigLIP-So400m vision encoder [88] with the Gemma-2-2B language backbone [21].

The two MLLMs differ along several axes that matter for evaluating the generality of MMDiff: backbone family (LLaMA vs. Gemma), vision-encoder objective (CLIP contrastive vs. SigLIP pairwise), hidden size, depth, and the family of base-LM SAE suite available for each backbone (LLaMA-Scope TopK SAEs [27] for LLaVA-MORE and Gemma-Scope JumpReLU SAEs [45] for PaliGemma 2). Where a property is shared across both models we report it as a property of MMDiff; where it is specific to one MLLM we say so explicitly.

A.2 Sparse Autoencoders

Motivation. Internal representations of large language models exhibit *superposition*: more features are encoded than there are neuron dimensions, with many features sharing the same residual-stream coordinates [**?**]. As a consequence, individual neurons are typically polysemantic, and per-neuron analyses confound multiple functional roles. Sparse Autoencoders (SAEs) attempt to undo superposition by learning an overcomplete dictionary of feature directions in which each input activation is approximated by a small number of active features [8, 16]. The dictionary directions are not constrained to align with neurons, which lets them recover finer-grained, often more interpretable units of computation.

Vanilla SAE. Given an activation $x \in \mathbb{R}^D$ taken from a transformer's residual stream, a vanilla SAE consists of an encoder and a decoder,

$$h(x) = \mathrm{ReLU}(W_{\mathrm{enc}}\, x + b_{\mathrm{enc}}), \qquad \hat{x} = W_{\mathrm{dec}}\, h(x) + b_{\mathrm{dec}},$$

with $W_{\mathrm{enc}} \in \mathbb{R}^{F \times D}$, $b_{\mathrm{enc}} \in \mathbb{R}^F$, $W_{\mathrm{dec}} \in \mathbb{R}^{D \times F}$, and $b_{\mathrm{dec}} \in \mathbb{R}^D$. The dictionary size F is typically chosen larger than the input dimension D (overcomplete). Each decoder column

$$v_f \;=\; (W_{\mathrm{dec}})_{:,f} \;\in\; \mathbb{R}^D$$

defines feature f's direction in residual-stream space, and the corresponding encoder row $(W_{\mathrm{enc}})_{f,:}$ acts as a detector that determines when f is present in the input. We refer to $h_f(x)$ as feature f's activation strength on x.

Training minimizes a reconstruction term plus a sparsity penalty,

$$\mathcal{L}(x) \;=\; \|x - \hat{x}\|_2^2 \;+\; \lambda \sum_{f=1}^{F} |h_f(x)|,$$

where λ trades off reconstruction quality against L_1 sparsity.

TopK SAEs. TopK SAEs [20] replace the L_1 penalty with a hard top-k operator on the encoder pre-activations:

$$h(x) \;=\; \mathrm{TopK}_k\big(W_{\mathrm{enc}}\, x + b_{\mathrm{enc}}\big),$$

which keeps the k largest pre-activations and zeros the rest. This yields an exact-k sparsity guarantee per token: $\|h(x)\|_0 = k$ for every x. Because sparsity is enforced by the architecture rather than by a tunable penalty, TopK SAEs decouple the rate of feature firing from the reconstruction objective. The LLaMA-Scope suite [27] releases TopK SAEs for LLaMA-3.1-8B at fixed values of k; we use $k = 50$ in MMDiff-Llama.

JumpReLU SAEs. JumpReLU SAEs [72] use a learned per-feature threshold θ_f and fire feature f only when its pre-activation $z_f(x) = (W_{\mathrm{enc}})_{f,:}\, x + (b_{\mathrm{enc}})_f$ exceeds θ_f:

$$h_f(x) \;=\; \begin{cases} z_f(x) & \text{if } z_f(x) > \theta_f, \\ 0 & \text{otherwise.} \end{cases}$$

The thresholds θ_f are learned jointly with the encoder/decoder weights using a straight-through estimator, targeting an average sparsity ℓ_0 (the expected number of active features per token). Unlike TopK, JumpReLU does not enforce fixed per-token sparsity; instead, sparsity fluctuates around the target. For MMDiff-Gemma, we use the Gemma-Scope JumpReLU SAE with target $\ell_0 = 50$ [45].

A.3 Stage-Wise Model Diffing for MLLMs

Setup. Stage-wise model diffing [9] extends SAE-based interpretability across training stages by re-training dictionaries on activations from successive checkpoints of the *same* architecture, while keeping feature indices aligned across stages. Aligned indices mean that feature f in the stage-A SAE and feature f in the stage-B SAE are intended to refer to the same conceptual unit, so a feature's evolution can be tracked by quantities such as the cosine similarity between its decoder directions, $\cos\big(v_f^{(A)}, v_f^{(B)}\big)$, or by changes in its activation pattern on a fixed dataset. This per-feature alignment lets one ask, for each direction f, whether it is preserved (high cosine, similar firing), rotated (low cosine, similar role), repurposed (low cosine, different firing), or newly emergent.

Comparison with crosscoders. An alternative to stage-wise diffing is crosscoder-based model diffing [48], which trains a single SAE-like model that simultaneously reconstructs activations from multiple checkpoints, sharing a feature dictionary across them. Crosscoders give a single global decomposition and are convenient when the goal is to locate features that are systematically shared or unique across many models. Stage-wise diffing instead trains one dictionary per stage and aligns features post hoc, which provides finer per-feature resolution: features can rotate or specialize in ways that are visible at the dictionary level but would be averaged away by a single shared decoder. Recent analyses [58] report that crosscoders can have lower sensitivity for sparse adapted features (those that fire infrequently and account for a small fraction of activation variance), which is precisely the regime we operate in, since single spatial relations and per-category safety triggers are rare on a generic VQA distribution.

Why warm-starting from the base-LM SAE works. Kissane et al. [37] show that SAE dictionaries trained on a base LM largely transfer to its fine-tuned counterparts: most features remain aligned and a relatively small fraction is meaningfully reshaped. This empirical finding has two consequences for MMDiff. First, it justifies initializing the MLLM-adapted SAE from the base-LM SAE rather than retraining from scratch, since the warm start preserves the monosemantic features already learned for the language backbone and only the multimodally affected subset needs to be tracked. Second, it explains why the adapted feature set is small enough to be useful as a discovery signal: if every feature were rotated by adaptation, the diff would collapse back to the full dictionary and lose its specificity. In our experiments (Sec. 3.2), the adapted set typically contains around 5% of features, which is consistent with the largely-transfer picture.

Limitations. Stage-wise diffing assumes the base LM and adapted model share an architecture and vocabulary, which holds for the LM,→,MLLM transition studied here (the language backbone is unchanged; only a vision encoder, projector, and multimodal fine-tuning are added). It also assumes adaptation induces feature-level changes rather than wholesale residual-space rotations. Our diagnostics (App. C) support this most strongly for text-only SAEs, which preserve alignment with the base-LM dictionary and are therefore used for diffing. In regimes where these assumptions weaken (e.g., early-layer image-only SAEs), we observe larger decoder rotations and reduced feature alignment.

B Algorithm: The MMDiff Pipeline

Algorithm 1 summarizes the full MMDiff pipeline as referenced in the main paper Sec. 3. Stage 1 trains a multimodal SAE warm-started from the base-LM dictionary; Stage 2 selects features that have rotated under multimodal training and prefer visual input; Stage 3 isolates a task-specific subset by contrasting per-token firing under a target distribution against a generic VQA baseline, with a Fisher-exact selectivity test (BH-corrected) and a lexical-invariance filter.

Algorithm 1 MMDiff: Multimodal Model Diffing for Feature Discovery

Require: base-LM SAE $\mathcal{S}_{\text{base}}$; multimodal model $\mathcal{M}_{\text{vlm}}$; VQAv2 mix $\mathcal{D}_{\text{base}}$; target distribution $\mathcal{D}_{\text{tgt}}$; thresholds $\varepsilon, q, \tau_{\text{OR}}, \tau_{\Delta p}$
Ensure: task-specific feature set $\mathcal{T}$

Stage 1: Adapt SAE to MLLM activations
$\mathcal{S}_{\text{vlm}} \leftarrow \textsc{AdaptSAE}(\mathcal{S}_{\text{base}}, \mathcal{M}_{\text{vlm}}, \mathcal{D}_{\text{base}})$ ▷ warm-start, fine-tune
validate per-layer FVU; confirm text-only regime converges
Stage 2: Identify adapted features
for each feature f in $\mathcal{S}_{\text{vlm}}$ **do**
  $c_f \leftarrow \cos\big(W_{\text{dec}}^{\text{base}}[f],\, W_{\text{dec}}^{\text{vlm}}[f]\big)$ ▷ decoder geometry
  $E_v(f) \leftarrow \mathbb{E}_{\text{vis}}[\, h_f(x)^2 \,]$ ▷ visual energy
end for
$\mathcal{A} \leftarrow \{f : E_v(f) > \varepsilon \,\wedge\, c_f \in \text{bottom-}q \text{ quantile}\}$
Stage 3: Discover task-specific features
for each $f \in \mathcal{A}$ **do**
  $p_f^{\text{base}} \leftarrow \textsc{PerTokenFiringRate}(f, \mathcal{D}_{\text{base}})$
  $p_f^{\text{tgt}} \leftarrow \textsc{PerTokenFiringRate}(f, \mathcal{D}_{\text{tgt}})$
  $\text{OR}_f \leftarrow \textsc{FisherExactTest}(p_f^{\text{base}}, p_f^{\text{tgt}})$ ▷ BH-corrected
end for
$\mathcal{C} \leftarrow \{f \in \mathcal{A} : \text{OR}_f \geq \tau_{\text{OR}} \,\wedge\, \Delta p_f \geq \tau_{\Delta p}\}$ ▷ statistical candidates
$\mathcal{L} \leftarrow \{f \in \mathcal{C} : f \text{ fires on neutral lexical-invariance prompts}\}$ ▷ lexical filter
$\mathcal{T} \leftarrow \mathcal{C} \cap \mathcal{L}$
return $\mathcal{T}$

B.1 MMDiff CAA Steering

Algorithm 2 gives the steering recipe referenced in Sec. 4: extract a feature direction v_f from $\mathcal{M}_{\text{vlm}}$, combine it with the canonical CAA mean-activation contrast injected across a backbone layer set

$\mathcal{L}$, and apply the intervention to $\mathcal{M}_{\text{base}}$. The OCR variant differs only in the contrast cache (GT vs. distorted-GT pairs) and the injection token (decision position rather than each generated token).

Algorithm 2 MMDiff CAA Steering

Require: feature f at layer ℓ_f with decoder direction v_f from $\mathcal{S}_{\text{vlm}}$; backbone-CAA layer set $\mathcal{L}$; positive/negative prompt sets P^+, P^- from $\mathcal{D}_{\text{tgt}}$; intervention scales α, γ_f; target model $\mathcal{M}_{\text{base}}$
Ensure: steered $\mathcal{M}_{\text{base}}$ residual stream

```
for each layer ℓ ∈ L do
    h̄_ℓ^+ ← E_{x∈P^+}[ h_ℓ(x; M_vlm) ]
    h̄_ℓ^- ← E_{x∈P^-}[ h_ℓ(x; M_vlm) ]
    d_ℓ ← (h̄_ℓ^+ − h̄_ℓ^-)/‖h̄_ℓ^+ − h̄_ℓ^-‖                    ▷ unit-normalize
end for
At inference on M_base, for each text token position t:
for each layer ℓ ∈ L do
    if ℓ = ℓ_f then
        h_{ℓ,t} ← h_{ℓ,t} + α d_ℓ + γ_f v_f                ▷ backbone CAA + feature amplification
    else
        h_{ℓ,t} ← h_{ℓ,t} + α d_ℓ                           ▷ backbone CAA only
    end if
end for
```

C Multimodal SAE Training

C.1 Training regimes

For each MLLM, we train SAEs on cached hidden states from 50k VQAv2 image–question pairs. The full input sequence contains projected visual tokens followed by text tokens, allowing token-type masks to determine which positions contribute to the reconstruction loss. We compare three masked training regimes: *full-sequence*, where all tokens contribute; *image-only*, where only projected visual-token positions contribute; and *text-only*, where only non-visual token positions contribute. In all three masked regimes, the SAE is warm-started from the matching base-LM SAE suite, receives hidden states from the same MLLM forward pass, and is trained under identical optimizer settings and schedules; only the loss mask differs.

Random-initialization control. As a strict initialization ablation we also train a fourth variant (*random*) whose SAE encoder, decoder, and bias parameters are sampled from the SAE-suite's default initialization scheme rather than warm-started from the base-LM SAE checkpoint. Random-init SAEs use the *full-sequence* loss (no token-type masking), the same Adam optimizer settings, the same learning rate, the same width-based LR scaling (which is constant across layers in our setup since each layer's SAE shares the same dictionary width), the same number of cached training tokens, and the same per-chunk training schedule as the warm-started variants. Unlike the three masking regimes, the random-init control does not separate modality-specific contributions; its purpose is to verify that adaptation benefits from a pretrained base-LM dictionary rather than from any sufficiently parameterized sparse code. The consistently higher FVU of the random-init variant in Fig. 5 and Fig. 6 confirms this.

Why text-only masking for model diffing. MMDiff asks which *language-backbone* features are repurposed by multimodal training, not which features reconstruct projected visual tokens most accurately. For stage-wise diffing to be meaningful, the adapted SAE must remain aligned with the base-LM dictionary so that feature-wise decoder comparisons remain interpretable. Text-only masking best preserves this alignment: text-token activations remain close to the original LM basis while still reflecting visual context through cross-modal attention, whereas full-sequence and image-only training rotate the dictionary toward projector-induced visual activations and weaken feature correspondence. This interpretation is consistent with the lower FVU and higher decoder cosine alignment of the text-only regime in Figs. 5–7.

C.2 Optimization and configurations

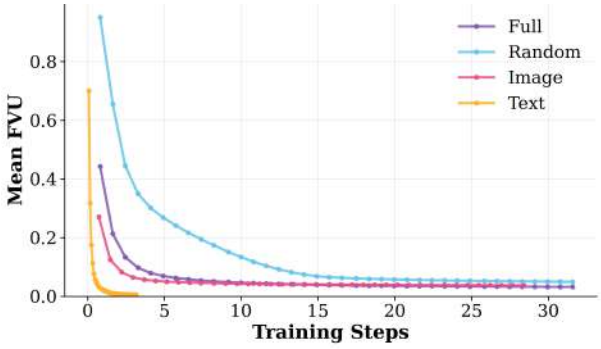
Training uses Adam ($\beta_1 = 0$, $\beta_2 = 0.999$, learning rate 7×10^{-5} with 1,000-step linear warmup) on cached activations from 50,000 VQAv2 image–question pairs, sharded into 1,000-sample chunks. For LLaVA-MORE we adapt LLaMA-Scope Top-K SAEs with $k = 50$ at training batch size 32; for PaliGemma 2 we adapt Gemma-Scope JumpReLU SAEs with target $\ell_0 = 50$, bandwidth 0.001 for the straight-through threshold gradient, and sparsity-penalty coefficient 1.0 ramped over 2,000 steps, at training batch size 8 (memory-bound by the 16,384-feature dictionary). All SAEs are trained with one job per layer in parallel on 8× A100 80GB. Reconstruction is evaluated on a held-out split via fraction of variance unexplained (FVU), $\mathrm{FVU} = \frac{\mathbb{E}[\|x-\hat{x}\|_2^2]}{\mathbb{E}[\|x-\mathbb{E}[x]\|_2^2]}$, which measures normalized reconstruction error (lower is better); achieved sparsity and architectural settings are summarized in Table 7.

MMDiff SAE	MLLM	Backbone	Vision enc.	SAE	L	d	Width	Sparsity
MMDiff-Llama	LLaVA-MORE	LLaMA-3.1-8B	CLIP ViT-L/14-336	TopK	32	4096	32K	k=50
MMDiff-Gemma	PaliGemma 2	Gemma-2-2B	SigLIP-So400m	JumpReLU	26	2304	16K	ℓ_0=50

Table 7: **MMDiff SAE configurations.** Each SAE is warm-started from the matching base-LM suite and adapted on cached VQAv2 activations across all backbone layers.

C.3 Reconstruction quality (MMDiff-Llama)

Figure 5 summarizes aggregated FVU trends for MMDiff-Llama, and Figure 6 reports the corresponding per-layer trajectories. Text-only SAEs converge rapidly and achieve the lowest reconstruction error across layers. Although text-only training uses fewer tokens than full-sequence or image-only training, it converges to lower FVU because those activations remain closer to the warm-started base-LM distribution. Image-only and full-sequence SAEs converge more slowly and plateau at higher FVU, consistent with a distributional mismatch between projected image tokens and the pretrained language-model activation basis. Random initialization performs worst, confirming that the pretrained base-LM dictionary provides a useful starting point for multimodal adaptation.



Metric	Full	Random	Image	Text
Mean	0.032	0.050	0.037	**0.005**
Std	0.028	0.041	0.027	**0.009**
Min	0.013	0.020	0.017	**0.000**
Max	0.123	0.198	0.123	**0.037**
Tokens (M)	31.6	31.6	28.4	3.2

Figure 5: **SAE adaptation (MMDiff-Llama).** Left: mean FVU across layers; right: per-regime FVU summary. Text-only achieves lowest reconstruction; random init worst.

C.4 Decoder geometry across regimes

Stage-wise diffing assumes that adaptation produces feature-level changes rather than an arbitrary wholesale rotation of the representation. To quantify how SAE feature geometry shifts across training regimes, we track cosine similarity between decoder directions from SAEs trained on different input types. Figure 7 shows that text-only SAEs remain closely aligned with the base-LM dictionary across layers, while image-only and full-sequence SAEs diverge in early layers before realigning deeper in the model. Randomly initialized SAEs stay largely uncorrelated, confirming the stability of the observed trends. We therefore use text-only SAEs as the main dictionary for model diffing, where feature identities are stable enough to compare base-LM and MLLM-adapted directions.

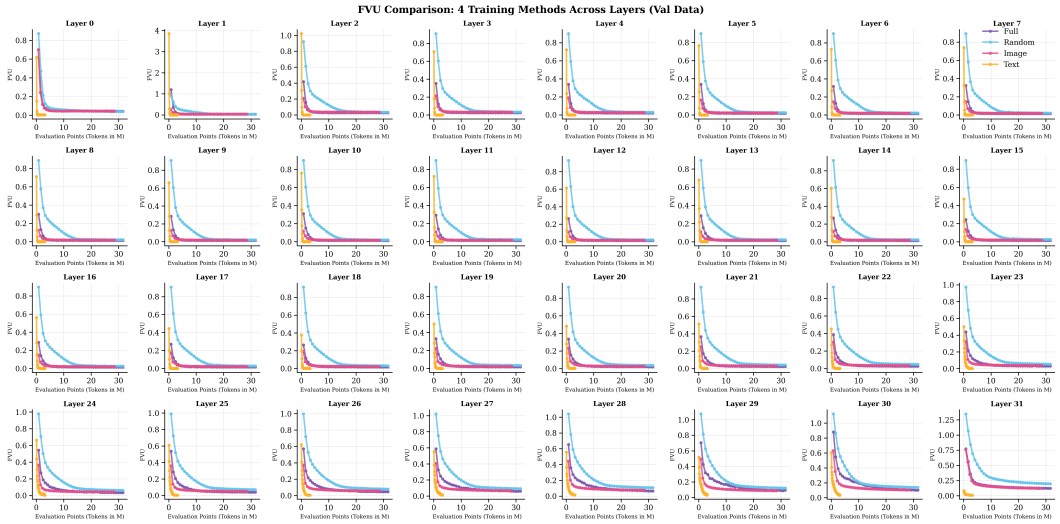


Figure 6: **Per-layer FVU across regimes (MMDiff-Llama).** Text-only SAEs converge to near-zero error rapidly; image and full-sequence regimes plateau higher.

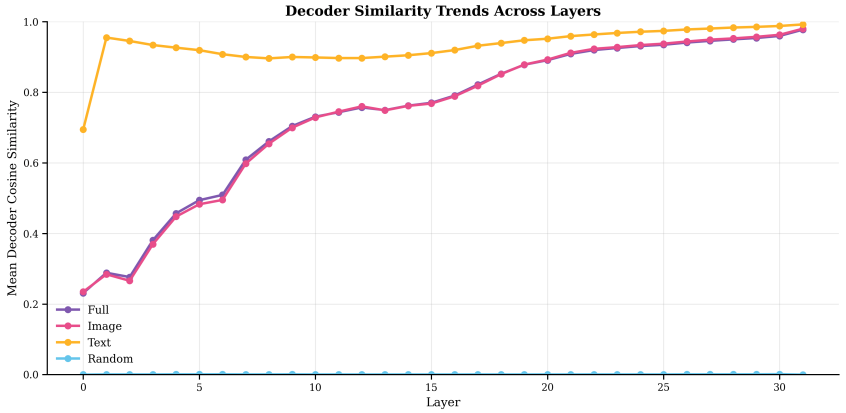


Figure 7: **Decoder cosine similarity vs. layer (MMDiff-Llama).** Text-only SAEs remain aligned with the base-LM dictionary across layers; image-only and full-sequence SAEs diverge in shallow layers and only re-align deeper. Random initialization stays decorrelated.

D Adapted Feature Selection Diagnostics

D.1 Joint visual-energy and cosine distribution

Figure 8 visualizes the joint distribution over E_v and c_f used in Sec. 3.2 to define the adapted set $\mathcal{A}$. Adapted features form a compact pink envelope; spatial candidates and the subset used for downstream analysis sit inside it.

D.2 Per-layer adapted-feature statistics

D.3 Threshold sweep for feature selection

Figure 11 shows how the size of the adapted set varies with the visual-energy cutoff ϵ and the cosine percentile $p_{\cos}$. The selection used in Sec. 3.2 ($p_{\cos} = 25\%$) is highlighted; downstream results are stable across reasonable choices.

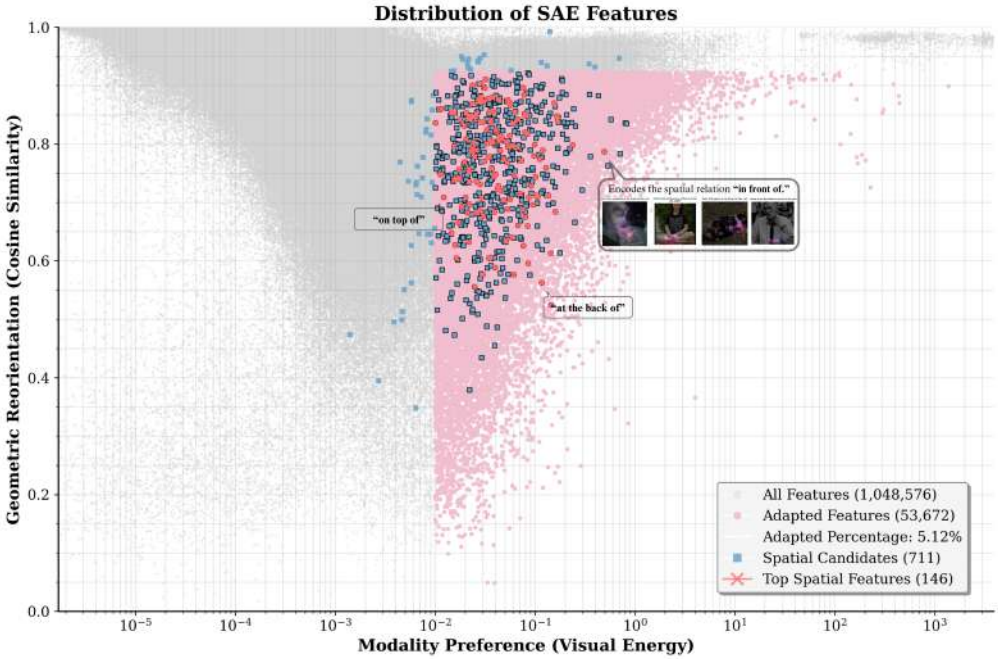


Figure 8: **Distribution of SAE features by visual energy and cosine similarity (MMDiff-Llama).** All features are shown in gray; adapted features are highlighted in pink. Task-specific candidates (here: spatial) are marked with blue squares, and the subset used for downstream analysis is shown as red crosses.

D.4 Filtering funnel for task-specific feature discovery

For each feature f, the odds ratio in Sec. 3.3 is computed from the firing/non-firing $\times$ target/baseline contingency table:

$$\mathrm{OR}_f = \frac{n_{\text{fire}}^{\text{tgt}} \cdot n_{\text{nonfire}}^{\text{base}}}{n_{\text{fire}}^{\text{base}} \cdot n_{\text{nonfire}}^{\text{tgt}}},$$

where n_{fire} and n_{nonfire} denote firing and non-firing token counts for feature f in the target and baseline distributions.

Table 8 summarizes the per-stage filtering counts referenced in Sec. 3.3 for both MMDiff models across all three target distributions. The funnel shrinks the full SAE dictionary (size $L \cdot F_{\text{width}}$) to a compact, task-specific feature set in three stages: (i) the adapted set $\mathcal{A}$ defined by the visual-energy and decoder-cosine criteria of Sec. 3.2; (ii) distribution-shift candidates within $\mathcal{A}$ that pass the Fisher exact test with $\mathrm{OR}_f \geq 3$ and $\Delta p_f \geq 0.05$ (Sec. 3.3, step 1); and (iii) the final task-specific set after lexical-invariance filtering (Sec. 3.3, step 2). The same three-stage funnel is applied uniformly across spatial, safety, and OCR target distributions; only $\mathcal{D}_{\text{tgt}}$ and the neutral-prompt bank change between applications.

Model	Target	All features	$\mathcal{A}$ (adapted)	Discovered
MMDiff-Llama	Spatial	∼1,024K	∼51K (5%)	711
MMDiff-Gemma	Spatial	∼416K	∼85K (∼20%)	∼1,400
MMDiff-Gemma	Safety	∼416K	∼85K (∼20%)	1,061
MMDiff-Gemma	OCR	∼416K	∼85K (∼20%)	1,070

Table 8: **Filtering funnel across MMDiff models and target distributions.** Counts at each stage of the task-specific discovery pipeline (Sec. 3.3). Total feature counts = $L \cdot F_{\text{width}}$ (Tab. 7). *Discovered* = features after contrastive-firing screen and lexical-invariance filter; for safety this is the candidate sweep size used in Sec. 5.2.

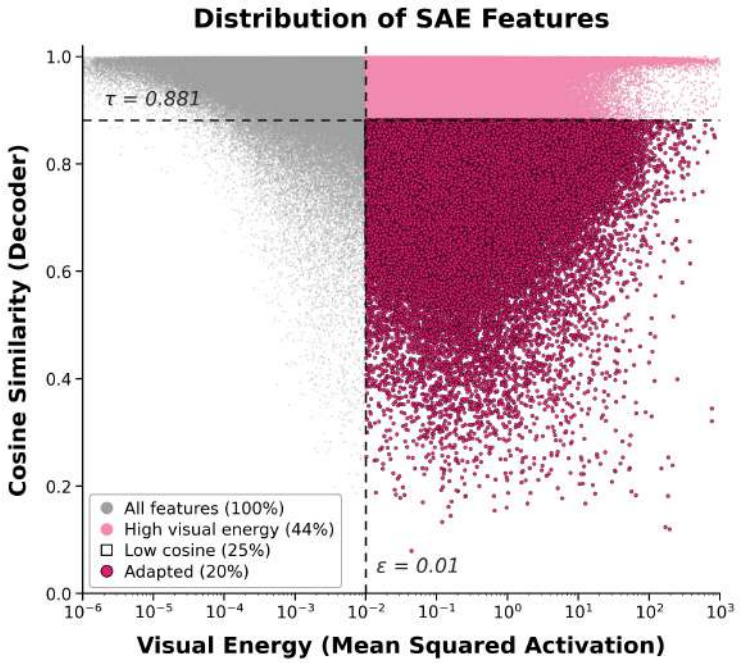
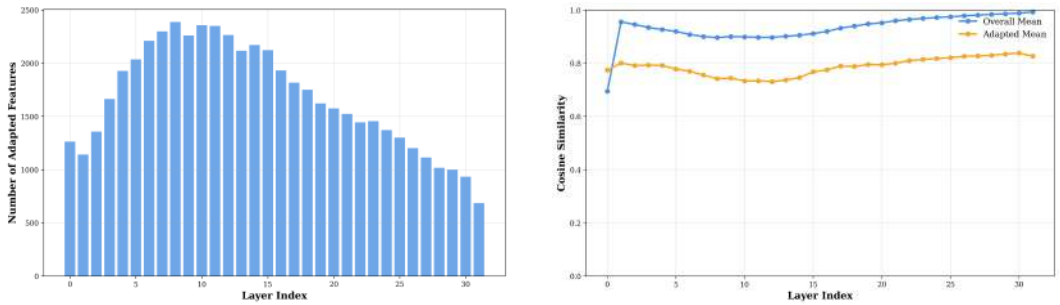


Figure 9: **Distribution of SAE features by visual energy and cosine similarity (MMDiff-Gemma).** All features in gray; high-visual-energy features ($E_v > \epsilon = 0.01$) in light pink; the adapted set $\mathcal{A}$ (high E_v *and* bottom-25% cosine, $\tau = 0.881$) in dark pink. Adapted features comprise $\sim$20% of MMDiff-Gemma's dictionary, larger than the $\sim$5% adapted set on MMDiff-Llama (Fig. 8).



(a) Adapted features per layer. Most concentrate in mid layers, tapering in deeper blocks.

(b) Decoder cosine by layer. Adapted features remain less aligned to the base dictionary than the overall pool.

Figure 10: **Per-layer adapted-feature stats (MMDiff-Llama).** Counts and mean cosine for the adapted set $\mathcal{A}$ (Sec. 3.2).

D.5 Distribution-shift visualizations

Figure 12 shows the per-feature firing-frequency distributions for the spatial split $\mathcal{D}_{\text{sp}}$ and the baseline $\mathcal{D}_{\text{base}}$ used in Sec. 3.3. Analogous histograms for the safety target distribution and (when results land) for the OCR target distribution are reported in supplementary panels.

D.6 Lexical-invariance prompt banks

The lexical-invariance filter (Sec. 3.3) replaces the original question on each top-activating sample with a small bank of generic prompts and keeps a feature only if it continues to fire on those generic prompts. We use three target-distribution-specific banks, listed verbatim from our pipeline:

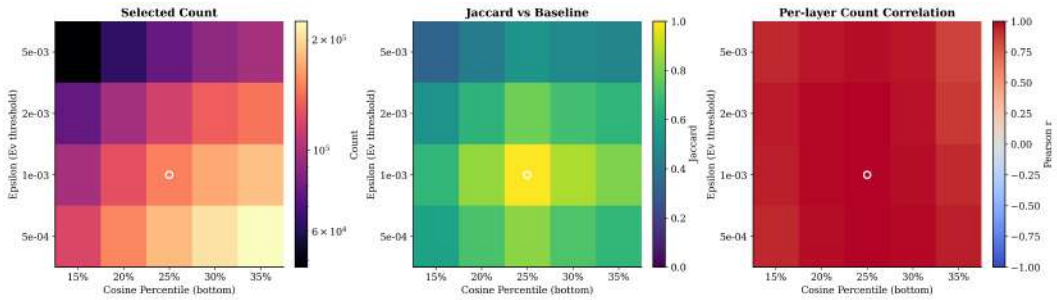


Figure 11: **Threshold sweep.** Adapted-set size as a function of ϵ and $p_{\cos}$.

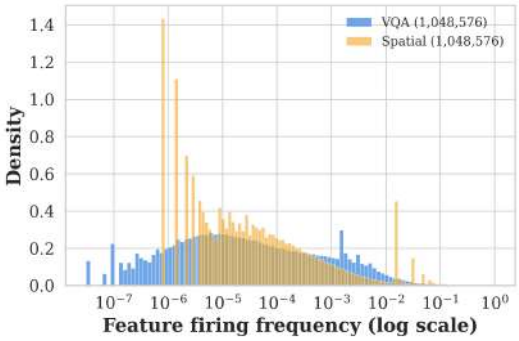


Figure 12: **Firing-frequency distributions for $\mathcal{D}_{\text{base}}$ and $\mathcal{D}_{\text{sp}}$.**

Spatial.

1. *Describe how the items are arranged.*
2. *Comment on the overall layout and organization of the scene.*
3. *Summarize the structure in terms of grouping or separation.*
4. *Explain the relative positioning of objects without naming directions.*
5. *Describe patterns of arrangement, such as order or symmetry.*

The spatial bank avoids any explicit spatial vocabulary (*left/right/above/below*, etc.) so that surviving features cannot be firing on the spatial *lexicon* of the prompt; they must be firing on the spatial *content* of the image.

OCR and Multimodal Safety.

1. *Describe this image.*
2. *What do you see in this picture?*
3. *Summarize the contents of the image.*
4. *Describe the objects and scene in this image.*
5. *What is happening in this image?*

For OCR and safety, the bank is generic image-description prompts (no OCR-specific cue like "what does the sign say", no safety-specific cue). Surviving features must fire on visual content alone.

A feature passes the lexical filter if it remains active above the activation cutoff $\eta = 0.01$ on at least one of the bank's prompts (per top-k activating sample, $k = 5$).

E Auto-Interpretation

E.1 Auto-Interpretation: Examples

For each MMDiff-discovered feature we collect its top-activating samples from VQAv2 and VSR [49] and pass them to GPT-4o-mini [65], which proposes a one-sentence description and an F1-based confidence score from a held-out classification task. Outputs are stored alongside the contrastive-firing metrics from Sec. 3.3 and lightly reviewed by hand; auto-interpretation is used as a qualitative validation layer rather than as a primary contribution. Figure 13 shows a representative example (Layer 16, Feature 176, MMDiff-Llama: features sensitive to *facing direction*). Two further examples follow; in both cases, the top-activating samples agree across VQA and VSR, and the interpretations are consistent and monosemantic.

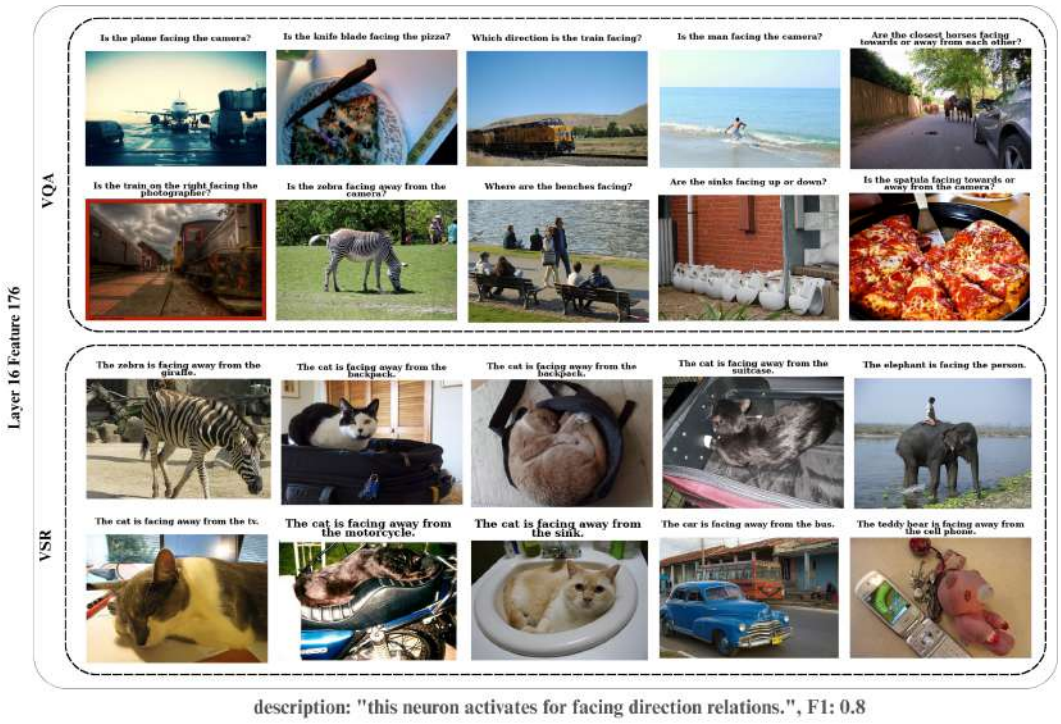


Figure 13: **Auto-Interp example (L16/F176, MMDiff-Llama).** Top VQA + VSR samples highlight *facing direction*.

E.2 Auto-Interpretation and Scoring Pipeline

We evaluate interpretability using an automated feature-description pipeline with two variants: *RAW* (image+text) and *OVERLAY* (image+text+top-head heatmaps). For each feature f:

1. Select up to k=5 top-activating samples (deduped across VQA / VQA-spatial / VSR).
2. Call the API once to generate a single concise description.
3. Validate using held-out positive samples and random VQA negatives (two short rounds).
4. Compute F1 as a lightweight proxy for description confidence.

Outputs are stored per feature as JSON (`description`, examples, classification results). Adding overlays improves interpretability, with early results showing a typical gain of about $+0.2$ F1.

Prompt A: Description (RAW / OVERLAY). **System.** You are analyzing individual neurons using their top-activating samples (image+text; OVERLAY also includes attention heatmaps). **Task.** Produce one short, lower-case sentence completing: "this neuron activates for ...". **Guidelines.**

Figure 14: **Additional Auto-Interp examples.** Two adapted features; top VQA + VSR samples show consistent spatial relations.

Figure 15: **Attribution patching on related spatial features.** Top: top head L13H1 localizes to relevant regions for "on top of". Middle: bottom-ranked heads fail to localize. Bottom: unrelated queries do not trigger spurious activation.

Base it on consistent patterns supported by image (+overlays) and text; be specific; no hedging. **Return.** `{"description": "one concise sentence"}`.

Prompt B: Validation (F1). **System.** You are validating a neuron description against short examples (image+text; OVERLAY adds heatmaps). **Task.** For each sample, output 1 if it reasonably matches the description; else 0. **Return.** `{"classifications": [0/1, ...]}`.

F Attribution Patching: Aggregated and Per-Feature Panels

Attribution patching [59] is an efficient alternative to activation patching [89], replacing repeated forward passes by a gradient-based linear approximation that estimates interventions with two forward and one backward pass. We adapt it to identify which attention heads drive a task-specific SAE feature f at layer L: we read out the SAE decoder direction v_f at layer L to define a scalar objective, and use gradients with respect to upstream residuals and attention inputs to score each head's contribution. The clean run uses original image–text input; the corrupt run replaces layer-0 visual token embeddings with a mean embedding over many VQA samples. The two variants are $(\text{corr} - \text{clean}) \cdot \nabla_{\text{clean}}$ (Method A) and $(\text{clean} - \text{corr}) \cdot \nabla_{\text{corr}}$ (Method B); we report per-layer and per-head scores averaged over the top-k samples activating f.

Across the spatial features we examined, layer-wise attribution curves peak in middle layers, consistent with the layer distribution of MMDiff-discovered spatial features (Fig. 16). At the head level, both methods highlight a small subset of heads, and the top heads are largely consistent across variants (Fig. 17). Some heads recur across related spatial relations: in the top row of Fig. 15, head L13H1 attends to semantically relevant regions across queries about "on top of"; the middle row confirms that bottom-ranked heads on the same samples fail to localize, and the bottom row confirms that unrelated queries do not trigger spurious activation. This clustering near ℓ_f is consistent with the layer-targeted injection site used by MMDiff CAA in Sec. 5.1.

F.1 Formalism

This subsection makes the attribution-patching procedure explicit: the scalar objective, the corruption operator, and the per-layer and per-head score formulas, including the absolute-value choice raised in the reviews.

Notation. For an input x, let $h_t^{(\ell)}(x) \in \mathbb{R}^d$ denote the residual-stream output of transformer layer ℓ at token position t, and let $z_t^{(\ell)}(x) \in \mathbb{R}^d$ denote the input to the layer-ℓ output projection $W_O^{(\ell)}$, i.e. the concatenation of all head outputs at position t. We write $z_t^{(\ell,h)}(x) \in \mathbb{R}^{d_h}$ for the slice of $z_t^{(\ell)}(x)$ corresponding to head h. Token positions split as $\mathcal{T}(x) = \mathcal{T}_{\text{img}}(x) \sqcup \mathcal{T}_{\text{post}}(x)$, where $\mathcal{T}_{\text{img}}(x)$ is the contiguous span of projected visual tokens and $\mathcal{T}_{\text{post}}(x)$ is the post-image text span at which the model produces its answer.

Scalar objective. Given a target SAE feature f with decoder direction $v_f \in \mathbb{R}^d$ from the layer-ℓ adapted SAE, we define a scalar objective by reading out v_f at a chosen layer L (in our experiments $L = \ell$ unless stated otherwise):

$$\mathcal{L}(f \mid x) \;=\; \sum_{t \,\in\, \mathcal{T}_{\text{post}}(x)} \big\langle h_t^{(L)}(x),\; v_f \big\rangle. \tag{1}$$

Note that $\mathcal{L}$ uses the inner product with the decoder direction, *not* the SAE reconstruction $\hat{h}_t^{(L)} = \sum_{f'} h_{f'}(x)\, v_{f'}$ nor the gated activation $h_f(x)$. Using the inner product makes $\mathcal{L}$ a linear functional of the residual stream, so gradients flow regardless of whether f is selected by the TopK or JumpReLU sparsity gate, and the linearization in Eqs. (4)–(7) below is well-defined.

Corruption operator. Let $\bar{e}_{\text{img}} \in \mathbb{R}^d$ be the mean layer-0 input over visual-token positions, computed across N reference VQAv2 samples drawn outside the target distribution (we use $N = 256$):

$$\bar{e}_{\text{img}} \;=\; \frac{1}{\sum_i |\mathcal{T}_{\text{img}}(x_i)|} \sum_{i=1}^{N} \sum_{t \,\in\, \mathcal{T}_{\text{img}}(x_i)} h_t^{(0)}(x_i). \tag{2}$$

The clean run uses the original x; the corrupt run replaces layer-0 inputs at every visual-token position with $\bar{e}_{\text{img}}$, leaving all other positions and all later layers' computations unchanged:

$$h_t^{(0)}(x^{\text{corr}}) \;=\; \begin{cases} \bar{e}_{\text{img}} & \text{if } t \in \mathcal{T}_{\text{img}}(x), \\ h_t^{(0)}(x) & \text{otherwise.} \end{cases} \tag{3}$$

This preserves the layer-0 distributional statistics of visual-token positions while suppressing the image-specific signal that drives the target feature, so $h_t^{(\ell)}(x^{\text{clean}}) - h_t^{(\ell)}(x^{\text{corr}})$ at any later layer ℓ is the residual-stream change attributable to the visual content of x.

Per-layer attribution. For each layer $\ell < L$ we evaluate the first-order approximation of the change in $\mathcal{L}(f \mid x)$ when $h_t^{(\ell)}$ is patched, using the gradient of $\mathcal{L}$ at one of the two endpoints. Method A linearizes around the clean run and Method B linearizes around the corrupt run:

$$s_\ell^{(A)}(f \mid x) \;=\; \frac{1}{|\mathcal{T}_{\text{post}}(x)|} \sum_{t \in \mathcal{T}_{\text{post}}(x)} \Big| \big\langle h_t^{(\ell)}(x^{\text{corr}}) - h_t^{(\ell)}(x^{\text{clean}}),\; \nabla_{h_t^{(\ell)}} \mathcal{L}(f \mid x^{\text{clean}}) \big\rangle \Big|, \tag{4}$$

$$s_\ell^{(B)}(f \mid x) \;=\; \frac{1}{|\mathcal{T}_{\text{post}}(x)|} \sum_{t \in \mathcal{T}_{\text{post}}(x)} \Big| \big\langle h_t^{(\ell)}(x^{\text{clean}}) - h_t^{(\ell)}(x^{\text{corr}}),\; \nabla_{h_t^{(\ell)}} \mathcal{L}(f \mid x^{\text{corr}}) \big\rangle \Big|. \tag{5}$$

The per-token absolute value is taken *before* averaging, so $s_\ell^{(A)}$ and $s_\ell^{(B)}$ measure attribution magnitude rather than signed effect: a high score means the layer's residual contribution is causally aligned with f in either direction.

Per-head attribution. The per-head version replaces the residual-stream activation $h_t^{(\ell)}$ in Eqs. (4)–(5) with the layer-ℓ output-projection input restricted to head h, $z_t^{(\ell,h)}$:

$$s_{\ell,h}^{(A)}(f \mid x) \;=\; \frac{1}{|\mathcal{T}_{\text{post}}(x)|} \sum_{t \in \mathcal{T}_{\text{post}}(x)} \Big| \big\langle z_t^{(\ell,h)}(x^{\text{corr}}) - z_t^{(\ell,h)}(x^{\text{clean}}),\; \nabla_{z_t^{(\ell,h)}} \mathcal{L}(f \mid x^{\text{clean}}) \big\rangle \Big|, \tag{6}$$

$$s_{\ell,h}^{(B)}(f \mid x) \;=\; \frac{1}{|\mathcal{T}_{\text{post}}(x)|} \sum_{t \in \mathcal{T}_{\text{post}}(x)} \Big| \big\langle z_t^{(\ell,h)}(x^{\text{clean}}) - z_t^{(\ell,h)}(x^{\text{corr}}),\; \nabla_{z_t^{(\ell,h)}} \mathcal{L}(f \mid x^{\text{corr}}) \big\rangle \Big|. \tag{7}$$

We attribute to $z^{(\ell,h)}$ rather than to the post-output-projection contribution because this isolates the per-head signal before mixing through $W_O^{(\ell)}$.

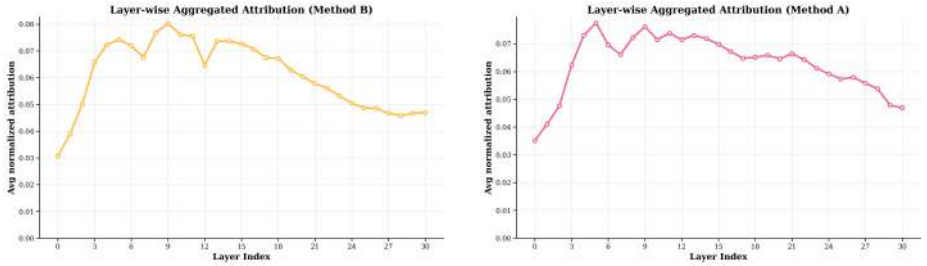


Figure 16: Layer-wise aggregated attribution curves for Method B (left) and Method A (right). Both peak around middle layers, consistent with the emergence of spatial features.

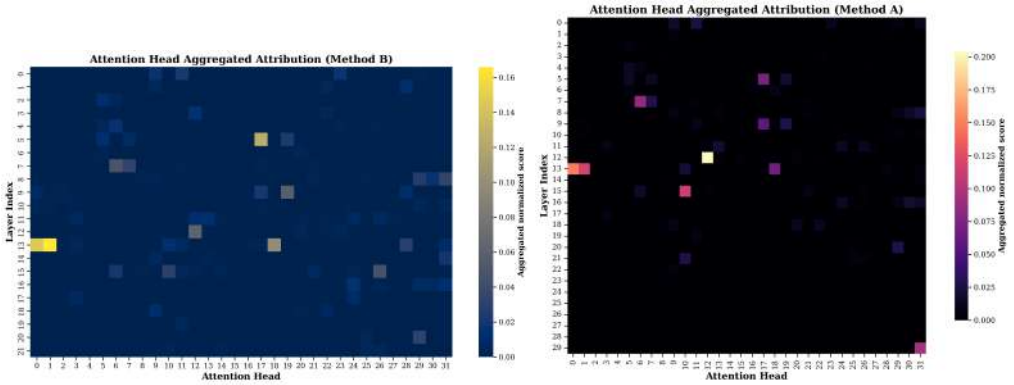


Figure 17: Attention head aggregated attribution maps for Method B (left) and Method A (right). Both highlight a similar set of specialized heads with high attribution scores.

Aggregation across samples. Both per-layer and per-head scores are averaged over the top-k samples that most strongly activate f on the target distribution (we use $k = 100$ unless stated otherwise). With $\mathcal{X}_f^{\text{top}}$ the top-k activating set and $\bullet \in \{A, B\}$,

$$S_\ell^{(\bullet)}(f) = \frac{1}{|\mathcal{X}_f^{\text{top}}|} \sum_{x \in \mathcal{X}_f^{\text{top}}} s_\ell^{(\bullet)}(f \mid x), \qquad S_{\ell,h}^{(\bullet)}(f) = \frac{1}{|\mathcal{X}_f^{\text{top}}|} \sum_{x \in \mathcal{X}_f^{\text{top}}} s_{\ell,h}^{(\bullet)}(f \mid x). \tag{8}$$

Top-driving heads are obtained by ranking $S_{\ell,h}^{(\bullet)}(f)$ in decreasing order; because the per-token absolute value is already taken inside the per-sample score (Eqs. (6)–(7)), this ranking is by attribution magnitude regardless of the sign of the underlying contrast.

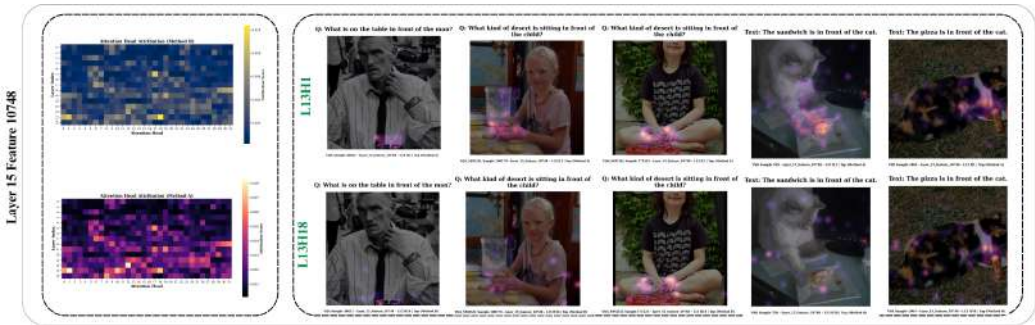
Figure 16 aggregates per-layer attribution scores across the spatial features in Sec. 5.1, showing the mid-layer concentration referenced in the main text. Figure 17 aggregates per-head scores. Figure 18 provides per-feature panels showing the top-scoring heads and their attention maps for individual features. Figure 19 examines how the recurring top heads behave under custom semantic prompts.

F.2 Bottom-Ranked Heads as a Control

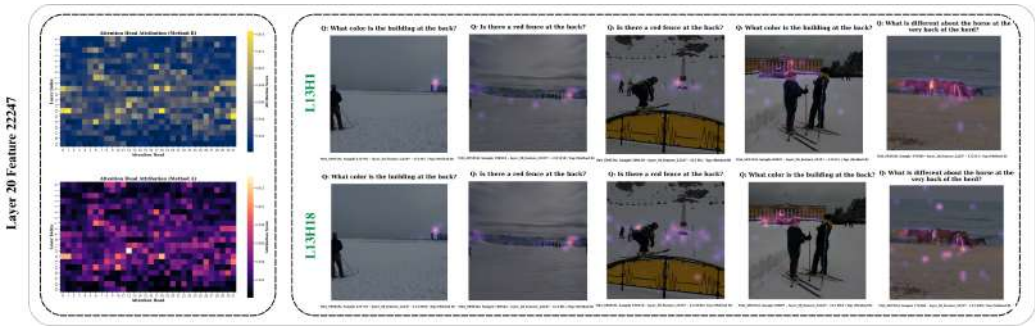
As a control, we visualize overlays from the *bottom-ranked* heads (per method, per feature). Across VSR and VQA top-activating samples, these heads generally fail to localize semantically relevant regions, in contrast to the top-ranked heads in Fig. 18.

G OCR Feature Examples

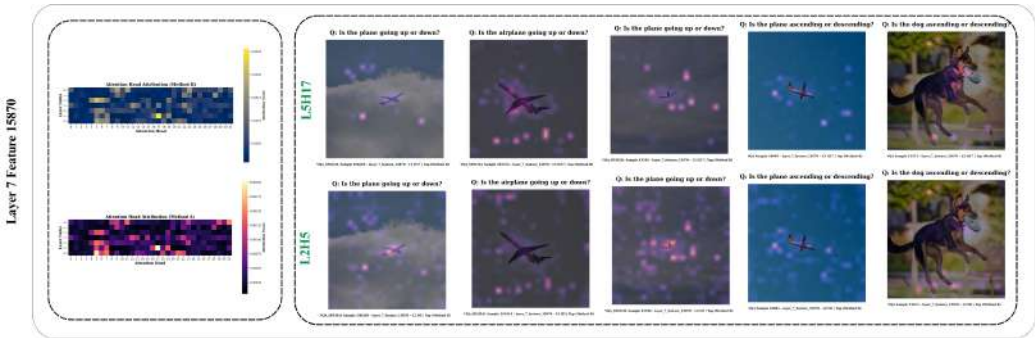
We also apply our distribution-shift procedure to OCR-style prompts (e.g., "What does the sign say?"). Fig. 21 shows that OCR-selective features cluster within the same adapted region as the spatial subset, indicating that multimodal fine-tuning concentrates visually grounded capabilities into a compact envelope of feature space.



(a) **Layer 15, Feature 10748.** VSR Relation: "in front of." Top heads (Method A): `L13H1`, `L12H12`, `L13H18`. Top heads (Method B): `L13H18`, `L5H17`, `L13H1`. *Overlap*: `L13H1`, `L13H18`. Attention overlays are shown on the top-activating samples across VSR and VQA.



(b) **Layer 20, Feature 22247.** VSR Relation: "at the back of." Top heads (Method A): `L12H12`, `L13H18`, `L13H1`. Top heads (Method B): `L13H1`, `L13H18`, `L14H31`. *Overlap*: `L13H1`, `L13H18`. Attention overlays are shown on the top-activating samples across VSR and VQA.



(c) **Layer 7, Feature 15870.** VSR Relation: "above." Top heads (Method A): `L5H17`, `L6H5`, `L0H31`. Top heads (Method B): `L5H17`, `L2H5`, `L2H6`. *Overlap*: `L5H17`. Attention overlays are shown on the top-activating samples across VSR and VQA.

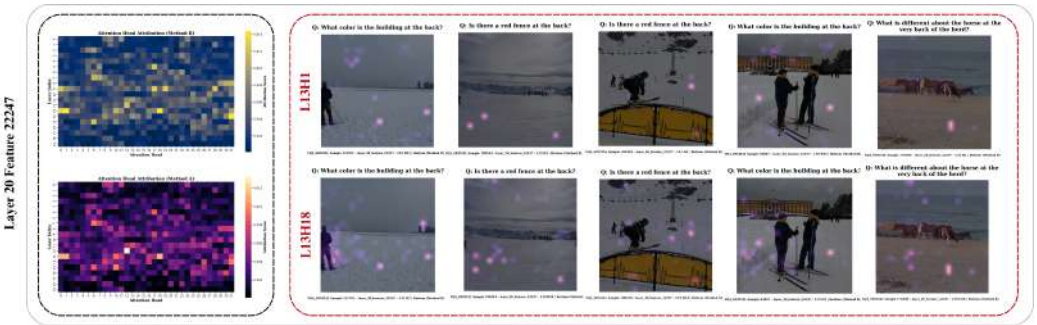
Figure 18: **Attribution patching on individual spatial features.** Each subfigure displays aggregated head/layer attribution maps (left) and attention overlays (right) using the strongest heads on the feature's top-activating samples across both VSR and VQA.

Qualitative examples confirm that these features reliably activate on embedded text and that associated heads localise to glyph regions (Fig. 22), consistent with image-grounded text processing.

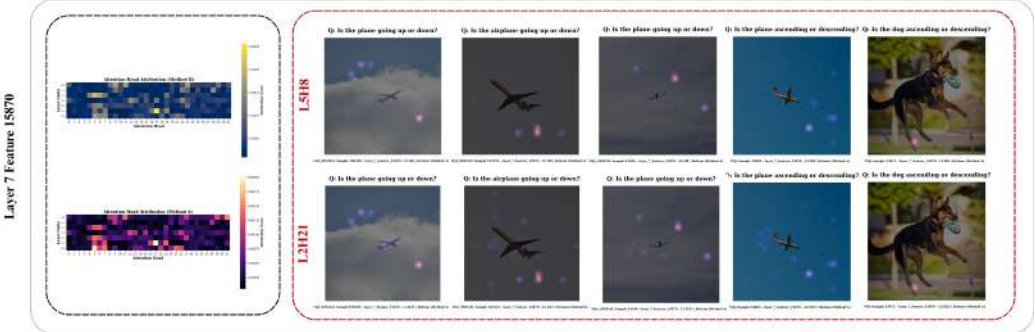
Qualitative panels show top-activating samples and overlay maps for OCR-selective features identified in Sec. 5.3. The main-paper ablation and steering tables for OCR are Tables 5 and 6; this appendix is reserved for per-feature qualitative panels.

Figure 19: **Attention head visualizations across queries.** Each row shows one image with attention overlays from a single high-attribution head across multiple spatial and non-spatial custom queries. The same heads consistently focus on semantically relevant regions.

(a) **Layer 15, Feature 10748.** VSR Relation: “in front of.”



(b) **Layer 20, Feature 22247.** VSR Relation: “at the back of.”



(c) **Layer 7, Feature 15870.** VSR Relation: “above.”

Figure 20: **Bottom-ranked heads yield weak localization.** For each feature, we show overlays from the lowest-scoring heads under Methods A and B on the feature’s top-activating samples across VSR and VQA. In contrast to Fig. 18, these heads produce diffuse or irrelevant attention.

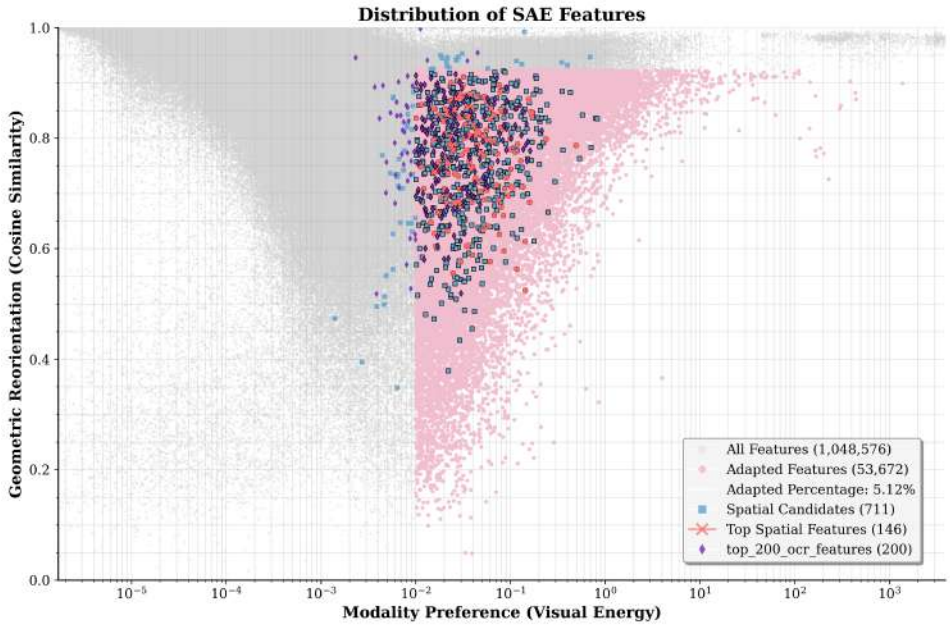


Figure 21: **Distribution of OCR features.** Top OCR candidates (purple) cluster among adapted units (pink), paralleling the spatial subset (blue).

Figure 22: **Layer 25, Feature 25568.** Activates on storefront and clothing text; top heads align to characters.