Learning to See Through Language: An Exploration of Language Modeling's Effect on Visual Representations

Mark Endo **Shiye Su** **Yuhui Zhang** **Serena Yeung-Levy**
Stanford University

Abstract

Does language modeling on image-text data truly improve vision, or does it merely adapt visual features for language alignment? Despite rapid progress in multimodal large language models (MLLMs), this question remains poorly understood. We present the first systematic analysis of visual representations in MLLMs by probing popular model families across a broad suite of dense visual tasks and comparing each MLLM to its exact pre-MLLM vision encoder. We find that language modeling largely improves visual representations for semantic tasks but can degrade performance on fine-grained geometric tasks, such as monocular depth. We further show that scaling the language model consistently improves the quality of visual representations. Given the improved MLLM representations, we next examine how to best extract and aggregate these features. Since no single layer provides a universally strong representation and concatenating features across layers is impractical, we propose a lightweight adapter that efficiently combines features across MLLM layers, turning frozen MLLMs into competitive visual backbones. Finally, we study whether MLLM representations can improve downstream applications, including image generation and inverse dynamics modeling, and show that they can outperform pre-MLLM visual features as general-purpose representations. Together, our results provide a representation-centric perspective for understanding and leveraging MLLMs as visual foundation models.[1]

1 Introduction

Multimodal large language models (MLLMs) have emerged as a powerful paradigm for a wide range of vision-language tasks, including visual question answering, visual reasoning, robotic control, and computer-use agents. In most MLLMs, a pre-trained vision encoder serves as an initial interface between pixels and the language model: it converts visual inputs into representations that are processed by the LLM for response generation. However, visual representations are reshaped during training, both across LLM layers and within the vision encoder, as models are trained end-to-end via next-token prediction on image-text data. This large-scale training drives MLLM capabilities, but its effect on underlying visual representations remains unclear. Most existing evaluations focus on the final language outputs of MLLMs, making it difficult to disentangle the quality of the learned visual features from the language model's ability to interpret and express them.

In this work, we systematically study visual representations inside MLLMs through three central questions:

1. Does the language modeling objective shaping MLLMs produce strong generalist vision learners? §2
2. How can MLLM visual features be most effectively extracted and aggregated in practice? §3
3. Can these features support downstream visual systems beyond traditional CV tasks? §4

[1]Project page: `https://web.stanford.edu/~markendo/projects/probebench`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

We begin by assessing *how large-scale training of MLLMs affects visual representations*. To this end, we propose ProbeBench, a reframing of MLLM evaluation designed to directly measure how visual information is encoded after MLLM training across different stages of the model. Using ProbeBench on four large-scale trained MLLMs, we find that MLLM training can improve visual representation quality, but its effects are strongly task-dependent. Improvements are most pronounced on semantic tasks such as segmentation and semantic correspondence, while geometric tasks show more mixed behavior, with monocular depth often degrading after MLLM training. Across pipeline stages, vision-encoder features are typically strongest and often degrade through the LLM layers, with semantic segmentation as a notable exception: its best representations often emerge in early-to-middle LLM layers. We further find that scaling the language model, while holding the vision encoder fixed, directly improves visual representations within each model family, although differences across MLLM families often exceed within-family scaling gains.

Since no single stage is universally strongest, we next ask *how features across stages should be combined*. We find that multi-stage aggregation can outperform any single layer on most dense tasks, offering a potential path to stronger visual representations from frozen MLLMs. However, this comes at a substantial cost where concatenation inflates feature dimensionality and forces every spatial position through the full language model. To address this, we introduce SLIM (Spatial Latents in MLLMs), a lightweight aggregation module that adaptively selects features across stages while pruning unnecessary LLM computation, turning frozen MLLMs into efficient visual backbones.

Lastly, we go beyond probing-based evaluation to test *whether the visual representations produced by MLLM training transfer to broader visual systems*. We evaluate MLLM features in two settings that move beyond per-image dense prediction: image generation and video-to-action inverse dynamics modeling. In image generation, MLLM features outperform the corresponding pre-MLLM vision encoder, and in inverse dynamics their pipeline-stage trends mirror those observed in ProbeBench, suggesting that the effects of MLLM training on visual representations extend beyond language-conditioned response generation to generation and control.

In summary, this work dissects MLLMs from a representation-centric perspective. Using ProbeBench as a feature-level evaluation suite, we systematically study how strong MLLM visual representations are, how they should be extracted and aggregated, and how they can enable downstream applications. Our findings suggest a new perspective on MLLMs: rather than viewing them solely as models that answer visual questions through language, we can also understand and leverage them as emerging visual foundation models.

2 Assessing Visual Representations of MLLMs

MLLMs are typically trained end-to-end on vision-language data and evaluated via benchmarks that grade the model's generated answers. While these benchmarks indicate end-to-end task performance, they give no direct view of the underlying visual representations. Truly assessing MLLMs as visual learners requires directly studying these representations and how language modeling shapes them. In this section, we first close this gap by proposing ProbeBench (§2.1), a reframing of MLLM benchmarking that evaluates MLLMs as visual foundation models by directly probing the visual representations inside them. Enabled by ProbeBench, we (1) analyze how language modeling changes visual representations relative to the pre-MLLM vision encoder and shapes representations across MLLM pipeline stages (§2.2) and (2) study how language model size affects visual representations (§2.3).

2.1 ProbeBench

ProbeBench evaluates MLLMs as visual foundation models by directly probing their visual representations. Whereas standard MLLM evaluation reflects both feature quality and the model's ability to utilize features through language (an ability that prior work has shown depends heavily on exposure during training to data closely matching the evaluation task [Zhang et al., 2024b, Fu et al., 2025]), ProbeBench targets the features themselves, isolating the learning of visual representations from their realized utility. For each MLLM, we train lightweight task heads on top of frozen features across pipeline stages and on the pre-MLLM vision encoder used for intialization, evaluating on a suite of patch-level visual tasks. This setup provides a clean before-versus-after comparison, isolating the visual learning that emerges from MLLM training.

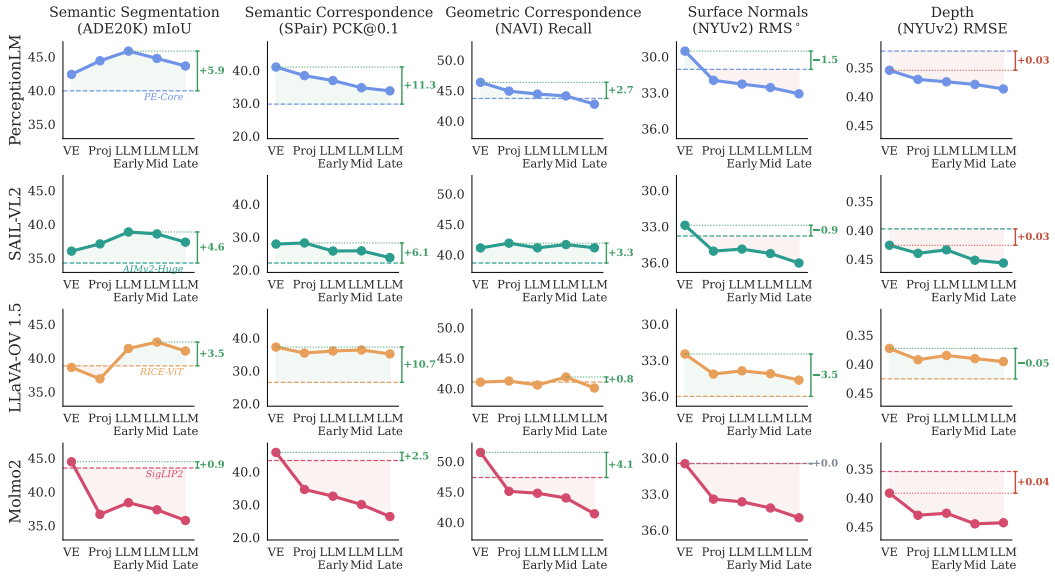

Figure 1: **Assessing MLLM visual representations with ProbeBench.** Each row is one MLLM family, each column is one task. Points trace probe scores across the five pipeline stages (vision encoder → projector → LLM Early/Mid/Late), and dashed lines show the standalone pre-MLLM vision encoder used to initialize the MLLM. MLLM training reshapes visual representations in a task-dependent manner: semantic tasks largely improve while geometric tasks show more mixed effects, with depth often degrading. Pipeline-stage strength also varies, with probed quality often degrading across LLM layers while semantic segmentation peaks at early-to-mid LLM layers.

Rather than relying on a single model or benchmark, we apply ProbeBench to four independently developed MLLM families spanning different pretraining corpora, vision encoders, and language backbones, and evaluate each on five tasks spanning four datasets and two protocols (trained probes and zero-shot). This diversity helps ensure that our findings reflect general trends rather than the particulars of any one model or dataset.

Tasks. ProbeBench is comprised of five dense visual tasks spanning semantic and geometric capabilities: *semantic segmentation* (ADE20K, 150 classes), *depth estimation* (NYUv2), and *surface normal estimation* (NYUv2) use trained probes; *semantic correspondence* (SPair-71k) and *geometric correspondence* (NAVI) are zero-shot evaluations following the protocol of El Banani et al. [2024] and Siméoni et al. [2025]. For trained probes, we use a single linear layer for segmentation (following the DINOv3 protocol) and a 4-layer decoder for depth and surface normals, since metric depth and surface orientation need not be linearly encoded [El Banani et al., 2024]. Full probe architectures, training hyperparameters, and learning-rate sweeps appear in Appendix B.2.

Pipeline stages. For every MLLM, we probe at five stages spanning the full pipeline: the vision encoder output (*VE*), the projector output (*Proj*), and three LLM layers (*Early*, *Mid*, *Late*, located at roughly 25%, 50%, and 75% of total depth), providing broad coverage across the pipeline without requiring substantial compute. For pre-MLLM standalone encoders, we extract from the same vision encoder layer(s) that the MLLM's projector connects to, ensuring an architecture-matched before/after comparison. To control for resolution, we set native tiling mechanisms to a single tile, so that each MLLM and its pre-MLLM vision encoder process the same input resolution. This way, any difference in probing quality reflects representation changes rather than additional resolution. Full details on per-model extraction and resolution control appear in Appendix B.1.

Models. Our model selection is guided by two criteria: the pre-MLLM vision encoder checkpoint must be publicly released to enable clean before-vs-after comparisons (excluding models such as Qwen3-VL [Bai et al., 2025]), and we prioritize MLLMs trained on large-scale image-text data. Following these criteria, we select four MLLMs: PerceptionLM-8B [Cho et al., 2025], SAIL-VL2-

8B [Yin et al., 2025], LLaVA-OneVision-1.5-8B [An et al., 2025], and Molmo2-O-7B [Clark et al., 2026]. These four models span four distinct vision encoder architectures (PE-Core-G14 [Bolya et al., 2025], AIMv2-Huge [Fini et al., 2025], RICE-ViT-Large [Xie et al., 2025], and SigLIP2 SO400M [Tschannen et al., 2025]), three language model families (Llama 3.1 [Grattafiori et al., 2024], Qwen3 [Yang et al., 2025a], OLMo2 [OLMo et al., 2024]), and a wide range of training-data scales ($\sim$9M-298M vision-text samples).

2.2 Impact of MLLM Training

We begin our analysis by applying ProbeBench to investigate how MLLM training reshapes visual representations relative to the pre-MLLM vision encoder and across various model stages. Figure 1 presents probing scores at every pipeline stage of each MLLM, alongside its standalone pre-MLLM vision encoder. We highlight our findings below.

Visual representations are reshaped in task-dependent manner. We find that all four MLLMs improve over their pre-MLLM encoder on segmentation and semantic correspondence for at least one pipeline stage, with the vision encoder often surpassing its initialization. Within geometric tasks the picture is more nuanced: geometric correspondence improves, monocular depth mostly degrades, and surface normals often improve at the vision encoder stage but degrade at subsequent stages. The improvements we observe at the vision encoder stage align with existing literature showing that unfreezing the vision encoder is a more effective training recipe than keeping it frozen [Tong et al., 2024]. Our results indicate that this benefit is not simply because visual features become better adapted for language alignment, but because the visual representations themselves improve, though doing so can also erode certain abilities (e.g., depth). More broadly, our results point to an opportunity for future work to study how language-supervised MLLM training can contribute to building stronger vision foundation models.

Finding 1: MLLM training largely improves visual representation quality for semantic tasks, while geometric tasks are more mixed, with depth often degrading.

Pipeline-stage strength varies. For most tasks, we observe that the vision encoder produces the strongest representations, with probed quality often degrading across LLM layers. However, for segmentation we find that for three of four models, the best representations emerge at early-to-mid LLM layers, hinting that LLM-stage representations carry rich semantic structure. Variation also appears across model families, where Molmo2 shows the largest drop in probed quality from the vision encoder to the projector on most tasks, a gap that persists after controlling for the projector's spatial downsampling (see Appendix C.2). Since visual information must pass through the projector to reach the LLM, this degradation may help explain why, despite using the strongest vision encoder, Molmo2 underperforms on some vision-centric benchmarks (e.g., 50.5 vs. PerceptionLM's 56.0 on BLINK [Fu et al., 2024]). Improving feature quality and preventing representational degradation remains a critical direction for future MLLM research.

Finding 2: Vision encoder features are typically the strongest (often degrading across LLM layers) except for semantic segmentation, which tends to peak at early-to-mid LLM layers.

MLLM features improve toward (but still trail) SOTA visual features. Despite improvements over the pre-MLLM encoder on many tasks, note that MLLM features still trail state-of-the-art self-supervised features. Namely, DINOv3 ViT-7B achieves better results across all five tasks (results in Appendix C.1). While the current standard MLLM recipe uses a language-supervised vision encoder (e.g., CLIP) as the visual backbone, recent work has shown that SSL encoders can scale effectively in multimodal systems [Fan et al., 2025]. One potential direction for future work is to explore how SSL features are reshaped through large-scale MLLM training.

2.3 Impact of Language Model Size

End-to-end MLLM performance scales with language model size, with prior work identifying perception as a central bottleneck in small MLLMs [Endo and Yeung-Levy, 2025]. However, the degree to which this scaling behavior reflects improvement in the underlying visual representations versus the LLM's utilization of the pre-trained visual features remains unexplored. ProbeBench

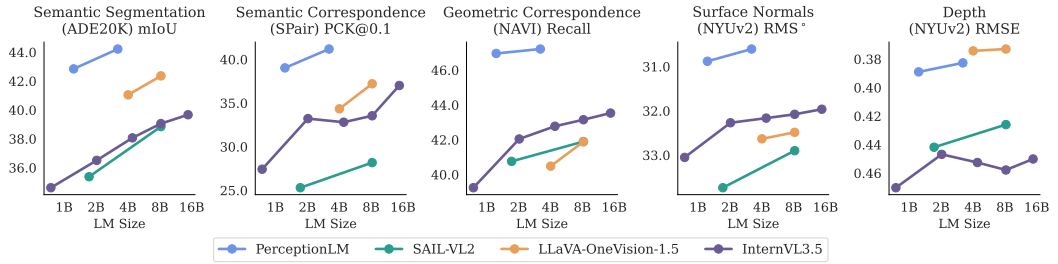

Figure 2: **Probed visual representation quality vs. language model size.** Each curve is a fixed-encoder MLLM family, with LM size on the x-axis and best probing score on the y-axis. With the vision encoder held fixed across sizes within each family, scaling the LM consistently improves probed quality across evaluated tasks.

enables us to address this by solely studying the quality of learned visual features, removing the confounding effect of LLM utilization. For a controlled experiment, we examine model families that hold the vision encoder fixed across multiple language model sizes. Thus, from the families in the previous study, we utilize PerceptionLM (1B/3B), SAIL-VL2 (2B/8B), and LLaVA-OneVision-1.5 (4B/8B). Additionally, we include InternVL 3.5 [Wang et al., 2025b], which provides five configurations (1B/2B/4B/8B/14B) on a single vision encoder for a more comprehensive comparison. For each model, we report the best probing score across pipeline stages. Figure 2 presents the resulting curves and we report our findings below.

Language model scaling consistently improves visual representation quality. We find that scaling the language model improves probed visual quality across all four families and all five tasks, indicating that larger language models genuinely produce more robust visual representations. Prior work has studied multimodal scaling primarily through final training loss [Shukor et al., 2025] or downstream VQA performance [Han et al., 2026], and the prevailing view is that performance is largely determined by how well the LLM extracts and interprets visual features [Fu et al., 2025, Liu et al., 2025]. By measuring the representations directly, we instead find that LM scaling improves the visual representations themselves, with gains surprisingly extending upstream to the vision encoder (see Appendix C.4).

Language model size is one of several factors shaping visual representations. While scaling the LM helps within each family, gaps between families at the same parameter count often exceed within-family scaling gains. For instance, PerceptionLM-1B not only outperforms other small models, but even surpasses every 8B-scale model from the other three families on all tasks except depth. These cross-family gaps indicate that choices that differ across MLLMs (e.g., vision encoder) have a larger impact on the underlying visual representation quality. This suggests that using a stronger vision encoder may be essential for equipping MLLMs with robust visual capabilities, likely more so than post-training alone, highlighting the importance of carefully designing and selecting vision encoders in MLLM development.

Finding 3: Scaling the language model directly improves visual representations within each family, though differences across MLLMs often exceed within-family scaling gains.

3 Towards MLLMs as Effective Visual Backbones

Our analyses show that MLLM visual representations often meaningfully improve over the pre-MLLM vision encoder. Given the rapid pace of MLLM development, effectively extracting these features as visual backbones is an increasingly important problem. In this section, we explore how to do so both effectively and efficiently. We first study how features from different MLLM pipeline stages interact (§3.1), finding that the effect of adding stages is largely inconsistent. Building on this, we propose SLIM (§3.2), a parameter- and compute-efficient adapter that adaptively aggregates features across pipeline stages without inflating feature dimensionality.

3.1 Assessing Synergy of Multi-Stage Features

Per-stage probing in §2.2 hinted that different stages of an MLLM encode different kinds of visual information. We now ask whether these representations are synergistic: can combining features from multiple stages outperform any single layer? Starting from the vision encoder layer that feeds into the LLM, we test three settings: (i) adding four earlier ViT layers, (ii) adding the projector output together with three LLM layers (early, mid, late), or (iii) combining all of the above (nine stages total). We focus on PerceptionLM-8B and report the change in probing score relative to the single-layer vision encoder baseline across the five ProbeBench tasks.

As shown in Figure 3, multi-stage aggregation can improve performance, but its effects are both stage- and task-dependent. Adding ViT-layer features improves segmentation and surface-normal prediction, but degrades zero-shot correspondence and depth estimation. Adding projector and LLM features, as expected, improves segmentation; it also improves geometric correspondence. Combining these feature groups typically yields gains comparable to the stronger of the two groups used in isolation.

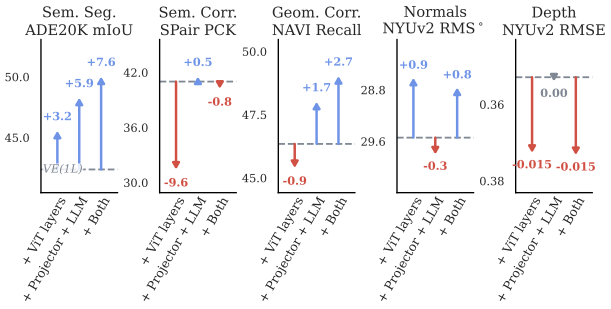

Figure 3: **Multi-stage feature aggregation.** Score change relative to single-layer vision encoder features on PerceptionLM-8B.

However, such naive feature combination has clear limitations. Depth estimation is a notable exception: the degradation introduced by ViT features persists after combination, leading to a drop comparable to using ViT features alone. Since no fixed feature composition performs best across all tasks, this motivates the design of a better general-purpose aggregation strategy.

Finding 4: Aggregating features across MLLM pipeline stages can outperform vision encoder features alone, but the effect of adding stages is inconsistent.

3.2 SLIM: Spatial Latents In MLLMs

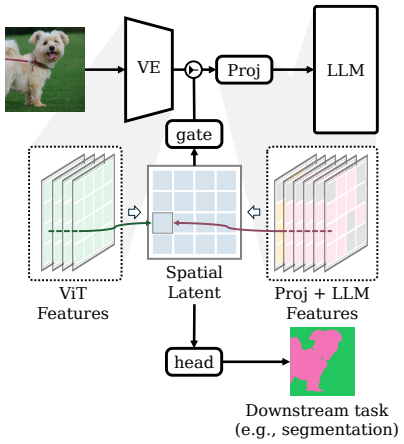

Figure 4: **SLIM: spatial latents in MLLMs.** We propose SLIM, a spatial-grid probe that cross-attends over multi-stage MLLM features with shared projections and learns a per-position gate to prune redundant LLM tokens.

While aggregating features across MLLM stages shows promise, feature concatenation has three drawbacks that prevent it from serving as a practical visual backbone. First, concatenation requires committing in advance to a specific set of stages, but as shown above, the right combination varies across tasks, requiring per-task manual experimentation to identify a useful set. Second, the approach is not scalable: the per-location feature dimension grows linearly with the number of stages, reaching ∼24k dimensions at PerceptionLM-8B with 9 stages. Third, running every spatial position through the full language model is costly in practice, since LLM layers carry substantially more parameters and compute per token than ViT layers.

To resolve these issues, we propose SLIM (Spatial Latents In MLLMs): a spatial grid of learnable latents that cross-attends to multi-stage MLLM features and adaptively prunes LLM tokens. The architecture is summarized in Figure 4; we describe its three core ideas below and include detailed architecture and hyperparameters in Appendix E.

Spatial latent grid with per-location cross-attention. SLIM maintains a fixed grid of learnable spatial latent queries $Q \in \mathbb{R}^{H \times W \times D}$, where $H \times W$ matches the ViT patch grid and $D = 512$ is a small bounded dimension. Features from all stages s (ViT blocks,

Table 1: **Performance of SLIM** on ProbeBench tasks across three MLLMs. Spatial latents improve over the best MLLM stage on every task and model, and over the standalone vision encoder in all but one setting, becoming competitive with strong general-purpose vision encoders. Our learned gate prunes most LLM tokens with minimal accuracy drop. Parentheses give the change relative to the MLLM best stage for ungated SLIM and the fraction of LLM tokens pruned for pruned SLIM.

	Sem. Seg. (ADE20K) ↑	Sem. Corr. (SPair) ↑	Geom. Corr. (NAVI) ↑	Normals (NYUv2) ↓	Depth (NYUv2) ↓
General-purpose vision encoders					
SigLIP2 g/16	41.0	58.7	45.7	31.4	0.378
Web-SSL DINO 7B	40.0	62.4	51.7	29.6	0.314
DINOv3 7B	53.2	74.7	58.0	28.4	0.295
PerceptionLM-8B					
Pre-MLLM VE	39.9	46.0	43.1	31.1	0.321
MLLM best stage	45.8	57.7	46.0	29.5	0.355
SLIM (ungated)	50.9 (+5.1)	60.5 (+2.8)	48.2 (+2.2)	28.9 (−0.6)	0.336 (−0.02)
SLIM (pruned)	50.6 (60%)	61.2 (75%)	45.9 (84%)	28.8 (65%)	0.353 (100%)
SAIL-VL2-8B					
Pre-MLLM VE	34.3	37.4	38.1	33.8	0.398
MLLM best stage	38.8	48.4	42.4	32.9	0.426
SLIM (ungated)	48.5 (+9.7)	50.5 (+2.0)	43.6 (+1.2)	31.1 (−1.8)	0.395 (−0.03)
SLIM (pruned)	48.2 (58%)	50.2 (73%)	42.7 (82%)	31.5 (97%)	0.377 (99.8%)
LLaVA-OneVision-1.5-8B					
Pre-MLLM VE	38.8	41.7	39.3	36.0	0.426
MLLM best stage	42.4	49.8	42.9	32.5	0.373
SLIM (ungated)	49.5 (+7.2)	55.4 (+5.6)	45.7 (+2.8)	30.1 (−2.4)	0.337 (−0.04)
SLIM (pruned)	49.3 (58%)	56.4 (64%)	45.9 (75%)	30.4 (100%)	0.344 (100%)

projector, LLM blocks) $X_{i,j}^{(s)}$ are projected to dimension D and stacked as key-value pairs; at each spatial position (i, j) the latent query $Q_{i,j}$ cross-attends only to features at that same position:

$$Q_{i,j} \leftarrow Q_{i,j} + \mathrm{CrossAttn}\big(Q_{i,j},\, \{F_{i,j}^{(s)}\}_{s\in\mathcal{S}_{i,j}}\big), \quad F_{i,j}^{(s)} = \gamma_s \odot W X_{i,j}^{(s)} + \beta_s.$$

The extractor's output is always $H \times W \times D$ regardless of how many stages contribute, eliminating the dimension blow-up of feature concatenation. Cross-attention weights are trained end-to-end on the downstream task, so the model learns task-appropriate weighting over stages rather than relying on manual layer selection. To keep the parameter count bounded as the number of stages grows, stages with the same input dimension share a single linear projection W, with per-stage identity restored via FiLM-style [Perez et al., 2018] scale γ_s and shift β_s ($2D$ parameters per stage). This keeps the parameter count nearly constant as more stages are added.

Adaptive depth via gating. Even with bounded dimension and learned stage weighting, every position still traverses the full LLM. Since LLM stages primarily aid semantic features (§2.2), most positions likely need only low-level ViT features. SLIM exploits this with a learned gate, a linear projection of the latent followed by a sigmoid, that scores each spatial position after the ViT-stage cross-attention. For positions deemed complete, the gate discards the LLM-stage cross-attention output and keeps the pre-LLM latent. Because Q is trained end-to-end, the gate is task-informed: it learns to forward only positions where deeper processing helps. During training, the gate uses a straight-through estimator under a sparsity regularizer $\lambda \cdot \mathrm{mean}(g)$; at inference, pruned tokens are physically removed from the LLM forward pass, giving compute savings proportional to the prune rate.

Setup. We evaluate SLIM on all five ProbeBench tasks. Like the probes in ProbeBench, SLIM is trained for each task on top of frozen MLLM features: its spatial latents replace the probe's input features, while the task heads match those of §2.1. Because the MLLM itself is never updated, this adds only a lightweight adapter, and we use the same training recipe across all tasks and models (Appendix E). Since SLIM has learned parameters, for correspondence we adopt a lightweight

supervised protocol [Luo et al., 2023], training the spatial latents on the SPair train set, evaluating on SPair test, and applying the trained head zero-shot to NAVI for geometric correspondence.

Baselines. We compare SLIM against the standalone vision encoder probed in isolation (*Pre-MLLM VE*) and the oracle best individual MLLM stage (*MLLM best stage*), which together capture what is achievable without multi-stage aggregation. We also report the SLIM head without our learned gate (*SLIM (ungated)*) to isolate the value of multi-stage aggregation from gating. To assess SLIM's practical utility as a visual backbone, we further benchmark directly against strong general-purpose vision encoders spanning language-supervised and self-supervised pretraining: SigLIP2 g/16 [Tschannen et al., 2025], Web-SSL DINO 7B [Fan et al., 2025], and DINOv3 7B [Siméoni et al., 2025]. All baselines receive identical treatment, using single-stage probes trained under the same protocol, including the supervised protocol for correspondence rather than zero-shot evaluation.

Results. As shown in Table 1, without pruning, SLIM substantially outperforms the oracle best individual MLLM stage in every setting across all three models and five tasks, confirming that aggregating across stages captures information no single stage carries on its own and resolving the inconsistency of fixed-stage concatenation (§3.1). In addition, it outperforms the standalone vision encoder in all but one model–task pair (depth on PerceptionLM). With our learned per-position gate enabled, SLIM still beats the oracle best stage in 14 of 15 settings with little loss in quality, even when most LLM tokens are pruned. In addition, the learned gate prunes nearly all LLM tokens on depth and the fewest on segmentation, in line with our finding that LLM-stage features are stronger for semantic than geometric tasks (§2.2). We additionally compare SLIM's learned gate against VisionZip's attention-based token pruning [Yang et al., 2025b] at matched prune rates in Appendix E.1.

Comparison with general-purpose vision encoders. Beyond SLIM's consistent improvement over the oracle best-stage baseline, we also find that it is competitive with general-purpose vision encoders. For example, the strongest SLIM setup using PerceptionLM features surpasses SigLIP2 on all five tasks, most notably on segmentation ($+9.9$) and depth ($+0.042$). Relative to the two DINO families, SLIM improves upon Web-SSL on segmentation and surface normals, and loses out on the correspondence tasks and depth. While it is still outperformed by DINOv3, SLIM narrows the gap relative to the best individual MLLM stage on every task, coming within 2.3 mIoU on segmentation and $0.5°$ on surface normals, supporting the potential of MLLM features as visual backbones as MLLMs continue to advance.

Finding 5: SLIM demonstrates a strong ability to aggregate features across MLLM stages while doing so efficiently, charting a path forward for using MLLMs as visual backbones.

4 MLLM Use in Downstream Systems

Having analyzed MLLM visual representations through ProbeBench, we now consider how these features can serve as backbones for more complex downstream applications. We focus on two case studies: image generation (§4.1), since strong visual representations support high-quality, fast-converging generative modeling [Zheng et al., 2025, Yu et al., 2024]; and inverse dynamics modeling (IDM) (§4.2), which leverages visual latents for world modeling and sequential decision making [Baker et al., 2022, Brandfonbrener et al., 2023].

4.1 Image Generation

We turn to image generation as a downstream application of MLLM visual representations. Visual generation is itself a rich area of research, and recent work has demonstrated that the standard VAE latent of latent diffusion models [Rombach et al., 2022], which is trained only for reconstruction and is semantically unstructured, can be effectively replaced by semantically rich visual features such as SigLIP2, leading to stronger generation results [Zheng et al., 2025]. Given our findings on the improved semantics of MLLM visual features (§2.2), we therefore investigate their potential for this form of image generation. We adopt the Representation Autoencoder (RAE) framework of Zheng et al. [2025], which replaces the VAE encoder with a frozen pretrained representation encoder paired with a trained ViT decoder. Because the encoder is frozen, generation quality depends crucially on the encoder's latent space: reconstruction FID measures how much pixel-level information the latents

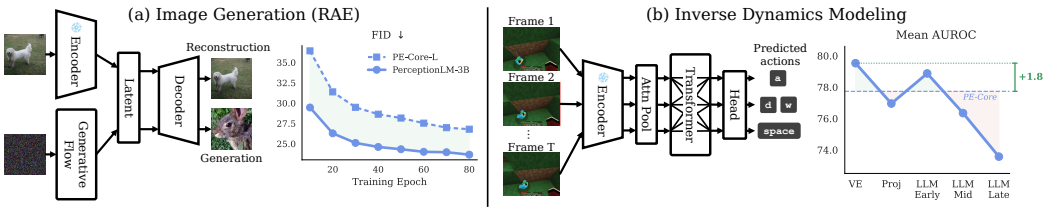

Figure 5: **MLLM use in image generation and IDM.** MLLM visual representations from PerceptionLM improve over the initialized vision encoder features (PE-Core) in image generation, and show a modest improvement in IDM.

retain, while generation FID measures how well the latent distribution can be learned by the diffusion model. RAE thus turns image generation into a controlled probe of representation quality.

Setup. We compare two encoders as the frozen RAE encoder: PE-Core-L [Bolya et al., 2025] and the vision encoder of PerceptionLM-3B (PLM-VE), which is initialized from PE-Core-L and further trained end-to-end as part of the MLLM. This pairing isolates the effect of MLLM visual-instruction fine-tuning while holding architecture and pretrained initialization fixed, providing a controlled before-vs-after comparison analogous to our probing setup in §2.2. Following the two-stage RAE recipe of Zheng et al. [2025], we first train a ViT decoder on top of the frozen encoder's latents and then a class-conditional diffusion transformer on top, both on ImageNet [Deng et al., 2009]. Due to compute constraints, our recipe is downscaled relative to Zheng et al. [2025], so our results are not directly comparable to their headline numbers; our goal is the *controlled, relative* comparison between PE-Core-L and PLM-VE. Full architectural, training, and evaluation details appear in Appendix F.

Results. At stage 1, the PLM-VE RAE reaches a stronger reconstruction FID (rFID 4.03 vs. 5.23 for PE-Core), indicating that MLLM training does not destroy the low-level information needed for pixel-faithful decoding. At stage 2, this advantage carries into generation: Figure 5(a) shows PLM-VE consistently outperforming PE-Core in gFID throughout training, converging to 23.7 vs. 26.8 at epoch 80. Classifier-free guidance preserves the gap (PLM-VE 22.3 vs. PE-Core 23.7). These results support that visual instruction tuning can enhance the visual latents for diffusion-based distribution learning, supporting more faithful reconstruction and more effective class-conditional generation.

4.2 Inverse Dynamics Modeling

In addition to image generation, we test whether the improved semantics of MLLM visual features surfaced by ProbeBench extend beyond per-image prediction to dynamics-aware visual understanding: the ability to encode how a scene evolves over time. Strong representations of this kind are needed to solve many open challenges in vision, including video understanding, world modeling, and embodied agents, making it important to assess whether MLLM training yields stronger visual features for these settings. As an entry point we consider inverse dynamics modeling: predicting the action sequence that produced an observed video. IDM requires representations that capture both per-frame scene semantics and inter-frame changes, directly assessing the dynamics-aware quality of MLLM features.

Setup. We use the Minecraft VPT setting [Baker et al., 2022]: each example is a $T{=}16$ frame clip with a 22-dimensional action target (20 binary keys + 2 continuous mouse deltas). We train on $\sim$18 hours of contractor gameplay at 20 fps (1.3M frames), and evaluate on a separately-held-out test set of 30 recordings ($\sim$2.5 hours). Following the frozen-backbone protocol of §2, we keep the visual model fixed and train only a lightweight per-frame attentive pool $\rightarrow$ bidirectional temporal transformer $\rightarrow$ MLP head to predict actions, with binary cross-entropy on keys and a small auxiliary mean squared error on mouse deltas (weight 0.1). We compare PerceptionLM-8B at every pipeline stage against its pre-MLLM PE-Core encoder, and report mean AUROC across the 20 binary keys on the test set. Full architectural and training details are in Appendix G.

Results. As shown in Figure 5(b), the pipeline-stage profile of PerceptionLM-8B for IDM closely mirrors what we observed for many ProbeBench tasks (§2.2): the vision encoder scores above its pre-MLLM initialization, while features degrade inside the LLM in mid and late layers. On IDM, the vision encoder's gain is modest ($+1.8$ mean AUROC; 95% CI $[-0.9, +3.0]$, paired bootstrap; Appendix G), whereas the degradation across LLM stages is more pronounced (78.9 at the early LLM

stage vs. 73.6 at the late stage). This suggests that the pipeline-stage trends surfaced by ProbeBench extend beyond per-image dense prediction, motivating future work that applies multi-stage extraction such as SLIM to more complex pipelines like IDM.

Finding 6: MLLM features show promise as backbones for image generation, and their pipeline-stage trends on inverse dynamics modeling mirror those observed in ProbeBench.

5 Related Work

Probing visual representations of MLLMs. A growing line of work analyzes MLLM visual features through the lens of LLM utilization: whether the visual information already inside the model is fully exploited by the language head. Liu et al. [2025] show that image value states in the LLM's KV cache encode rich perception information, but this information often fails to surface through the model's text answer. Fu et al. [2025] similarly find a performance drop from probing-based to generative evaluation on perception tasks; both conclude that visual representations are adequate but underutilized. Our work instead asks the more fundamental question of whether and how the visual representations themselves are reshaped by language modeling. A handful of recent works begin to examine this representational question, but at a much narrower scope: Liu et al. [2026] analyze Qwen2-VL and Qwen2.5-VL vision encoders on ADE20K and NYUv2, and Wu et al. [2026] report that LLM layers recover segmentation performance in LLaVA-style training (665K examples).

Multi-layer feature aggregation. Existing MLLM aggregation methods share a fundamentally different goal than ours: improving language-side generation rather than building strong visual backbones. DenseConnector [Yao et al., 2024] concatenates intermediate ViT features with a fixed schedule; DeepStack [Meng et al., 2024] (and Qwen3-VL's fusion module) injects mid-layer visual tokens directly into LLM decoder layers; MMFuser [Cao et al., 2024] cross-attends from a final-layer query over earlier ViT layers; and Cambrian-1's Spatial Vision Aggregator [Tong et al., 2024] aggregates across multiple vision encoders via a learnable spatial grid. All are integrated as an input to LLM training rather than evaluated as standalone visual backbones. SLIM in contrast aggregates per-spatial-location across the stages of a single frozen MLLM, yields a visual backbone for downstream tasks, and adds adaptive depth gating for efficiency. SLIM also relates to attentive probing of ViTs, where recent works improve probe parameter efficiency [Psomas et al., 2026] or fuse multiple ViT layers for classification [Ciernik et al., 2026]; beyond aggregating across a full MLLM, SLIM's gate also reduces compute by pruning spatial positions from the LLM forward pass.

Visual token pruning in MLLMs. A growing line of work prunes visual tokens to reduce LLM computation while preserving the MLLM's ability to answer text queries. For example, FastV [Chen et al., 2024], SparseVLM [Zhang et al., 2024a], and ATP-LLaVA [Ye et al., 2025] use text-guided importance scores to decide which tokens to drop, whereas VisionZip [Yang et al., 2025b] scores tokens directly from ViT attention without needing text. In contrast to these works, which target inference efficiency for text generation, SLIM repurposes token pruning as a representation-quality lens: the gate is trained jointly on the downstream task and prunes stages that do not carry useful visual information, rather than tokens that the LLM does not need to answer.

6 Conclusion

In this work, we take a representation-centric view of MLLMs and show that language modeling on image-text data does more than align visual features with language: it substantially reshapes the visual representations themselves. Through ProbeBench, we find that MLLM training, particularly with larger language models, generally strengthens semantic visual capabilities. We further introduce a lightweight aggregation adapter that efficiently combines frozen MLLM features across layers and show that MLLM features hold promise for downstream tasks such as image generation and inverse dynamics modeling. Together, these findings suggest that MLLMs should be understood not only as language interfaces for visual reasoning, but also as emerging visual foundation models whose representations can be studied, extracted, and leveraged in their own right.

Acknowledgments. This material is based upon work supported by the U.S. National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2146755 (for M.E.). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, Chunsheng Wu, Huajie Tan, Chunyuan Li, Jing Yang, Jie Yu, Xiyao Wang, Bin Qin, Yumeng Wang, Zizhen Yan, Ziyong Feng, Ziwei Liu, Bo Li, and Jiankang Deng. Llava-onevision-1.5: Fully open framework for democratized multimodal training, 2025. URL https://arxiv.org/abs/2509.23661.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025.

David Brandfonbrener, Ofir Nachum, and Joan Bruna. Inverse dynamics pretraining learns good representations for multitask imitation. *Advances in Neural Information Processing Systems*, 36: 66953–66978, 2023.

Yue Cao, Yangzhou Liu, Zhe Chen, Guangchen Shi, Wenhai Wang, Danhuai Zhao, and Tong Lu. Mmfuser: Multimodal multi-layer feature fuser for fine-grained vision-language understanding. *arXiv preprint arXiv:2410.11829*, 2024.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

Jang Hyun Cho, Andrea Madotto, Effrosyni Mavroudi, Triantafyllos Afouras, Tushar Nagarajan, Muhammad Maaz, Yale Song, Tengyu Ma, Shuming Hu, Suyog Jain, Miguel Martin, Huiyu Wang, Hanoona Rasheed, Peize Sun, Po-Yao Huang, Daniel Bolya, Nikhila Ravi, Shashank Jain, Tammy Stark, Shane Moon, Babak Damavandi, Vivian Lee, Andrew Westbury, Salman Khan, Philipp Krähenbühl, Piotr Dollár, Lorenzo Torresani, Kristen Grauman, and Christoph Feichtenhofer. Perceptionlm: Open-access data and models for detailed visual understanding, 2025. URL https://arxiv.org/abs/2504.13180.

Laure Ciernik, Marco Morik, Lukas Thede, Luca Eyring, Shinichi Nakajima, Zeynep Akata, and Lukas Muttenthaler. Beyond the final layer: Attentive multi-layer fusion for vision transformers. *arXiv preprint arXiv:2601.09322*, 2026.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, Sangho Lee, Zhongzheng Ren, Chris Dongjoo Kim, Yinuo Yang, Vincent Shao, Yue Yang, Weikai Huang, Ziqi Gao, Taira Anderson, Jianrui Zhang, Jitesh Jain, George Stoica, Winson Han, Ali Farhadi, and Ranjay Krishna. Molmo2: Open weights and data for vision-language models with video understanding and grounding, 2026. URL https://arxiv.org/abs/2601.10611.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Mohamed El Banani, Amit Raj, Kevis-Kokitsi Maninis, Abhishek Kar, Yuanzhen Li, Michael Rubinstein, Deqing Sun, Leonidas Guibas, Justin Johnson, and Varun Jampani. Probing the 3d awareness of visual foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21795–21806, 2024.

Mark Endo and Serena Yeung-Levy. Downscaling intelligence: Exploring perception and reasoning bottlenecks in small multimodal models. *arXiv preprint arXiv:2511.17487*, 2025.

David Fan, Shengbang Tong, Jiachen Zhu, Koustuv Sinha, Zhuang Liu, Xinlei Chen, Michael Rabbat, Nicolas Ballas, Yann LeCun, Amir Bar, et al. Scaling language-free visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 370–382, 2025.

Enrico Fini, Mustafa Shukor, Xiujun Li, Philipp Dufter, Michal Klein, David Haldimann, Sai Aitharaju, Victor G Turrisi da Costa, Louis Béthune, Zhe Gan, et al. Multimodal autoregressive pre-training of large vision encoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9654, 2025.

Stephanie Fu, Tyler Bonnen, Devin Guillory, and Trevor Darrell. Hidden in plain sight: Vlms overlook their visual representations. *arXiv preprint arXiv:2506.08008*, 2025.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pages 148–166. Springer, 2024.

Valentin Gabeur, Shangbang Long, Songyou Peng, Paul Voigtlaender, Shuyang Sun, Yanan Bao, Karen Truong, Zhicheng Wang, Wenlei Zhou, Jonathan T Barron, et al. Image generators are generalist vision learners. *arXiv preprint arXiv:2604.20329*, 2026.

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.

Junlin Han, Shengbang Tong, David Fan, Yufan Ren, Koustuv Sinha, Philip Torr, and Filippos Kokkinos. Learning to see before seeing: Demystifying llm visual priors from language pre-training. In *International Conference on Learning Representations*, volume 2026, pages 61025–61055, 2026.

Benlin Liu, Amita Kamath, Madeleine Grunde-McLaughlin, Winson Han, and Ranjay Krishna. Visual representations inside the language model. *arXiv preprint arXiv:2510.04819*, 2025.

Yikun Liu, Yuan Liu, Shangzhe Di, Haicheng Wang, Zhongyin Zhao, Le Tian, Xiao Zhou, Jie Zhou, Jiangchao Yao, Yanfeng Wang, et al. Versavit: Enhancing mllm vision backbones via task-guided optimization. *arXiv preprint arXiv:2602.09934*, 2026.

Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *Advances in Neural Information Processing Systems*, 36:47500–47510, 2023.

Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang. Deepstack: Deeply stacking visual tokens is surprisingly simple and effective for lmms. *Advances in Neural Information Processing Systems*, 37:23464–23487, 2024.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Bill Psomas, Dionysios Christopoulos, Eirini Baltzi, Ioannis Kakogeorgiou, Tilemachos Aravanis, Nikos Komodakis, Konstantinos Karantzalos, Yannis Avrithis, and Giorgos Tolias. Attention, please! revisiting attentive probing through the lens of efficiency. In *International Conference on Learning Representations*, volume 2026, pages 77155–77186, 2026.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Mustafa Shukor, Enrico Fini, Victor Guilherme Turrisi da Costa, Matthieu Cord, Joshua Susskind, and Alaaeldin El-Nouby. Scaling laws for native multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–23, 2025.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai C Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *Advances in Neural Information Processing Systems*, 37: 87310–87356, 2024.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. *arXiv preprint arXiv:2504.05741*, 2025a.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3. 5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025b.

Boyong Wu, Sanghwan Kim, and Zeynep Akata. From drop-off to recovery: A mechanistic analysis of segmentation in mllms. *arXiv preprint arXiv:2603.17228*, 2026.

Yin Xie, Kaicheng Yang, Xiang An, Kun Wu, Yongle Zhao, Weimo Deng, Zimin Ran, Yumeng Wang, Ziyong Feng, Roy Miles, et al. Region-based cluster discrimination for visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1793–1803, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19792–19802, 2025b.

Huanjin Yao, Wenhao Wu, Taojiannan Yang, YuXin Song, Mengxi Zhang, Haocheng Feng, Yifan Sun, Zhiheng Li, Wanli Ouyang, and Jingdong Wang. Dense connector for mllms. *Advances in Neural Information Processing Systems*, 37:33108–33140, 2024.

Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. Atp-llava: Adaptive token pruning for large vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24972–24982, 2025.

Weijie Yin, Yongjie Ye, Fangxun Shu, Yue Liao, Zijian Kang, Hongyuan Dong, Haiyang Yu, Dingkang Yang, Jiacong Wang, Han Wang, Wenzhuo Liu, Xiao Liang, Shuicheng Yan, and Chao Feng. Sail-vl2 technical report, 2025. URL https://arxiv.org/abs/2509.14033.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024a.

Yuhui Zhang, Alyssa Unell, Xiaohan Wang, Dhruba Ghosh, Yuchang Su, Ludwig Schmidt, and Serena Yeung-Levy. Why are visually-grounded language models bad at image classification? *Conference on Neural Information Processing Systems (NeurIPS)*, 2024b.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

Supplementary for *Learning to See Through Language: An Exploration of Language Modeling's Effect on Visual Representations*

This supplementary material provides additional discussion and details on the methods and experiments in the main paper. Appendix A discusses open questions and future directions raised by our analyses. Appendix B details the ProbeBench protocol used in §2, including matched pre-MLLM vision encoder baselines, pipeline-stage extraction, resolution control, and probe architectures and training. Appendix C reports additional ProbeBench results, including comparisons with state-of-the-art self-supervised features, a control for projector spatial downsampling, a multi-tile input analysis, and per-stage probing across language model sizes. Appendix D examines the relation between ProbeBench and standard end-to-end MLLM benchmarking, with a probe-based reanalysis of CV-Bench 3D Depth. Appendix E covers the SLIM architecture and training details from §3, along with a comparison against VisionZip token pruning and an analysis of the learned gate. Appendix F describes the image-generation (RAE) setup used in §4.1. Appendix G describes the inverse-dynamics-modeling setup used in §4.2.

A Discussion and Future Directions

This work is a first systematic step toward rethinking MLLMs as emerging foundation models whose representations can be studied, extracted, and leveraged in their own right. In ProbeBench, we focus on dense spatial semantic and geometric tasks, reflecting the patch-based way MLLMs process images, though future work may consider other regimes such as image or video classification. On the modeling side, we focus on the predominant late-fusion design, in which a pre-trained vision encoder is bridged to a language model; future work can analyze natively trained multimodal models and how their visual representations are shaped solely by language modeling. Towards utilization, our downstream case studies provide a first demonstration that MLLM features can serve as visual backbones, and a promising direction is more large-scale training of systems built on top of these representations. Overall, our analyses have surfaced many open questions about how language modeling shapes vision and how the resulting shifts can be best steered. We sketch a few directions we find most promising.

Training dynamics and recipe. Unfreezing the vision encoder reshapes its representations in beneficial ways, but with risk: certain abilities, notably monocular depth, can degrade. Tracing how these shifts emerge over the course of training, and how they depend on design choices like instruction tuning data and training recipes would clarify whether these changes are intrinsic to language-aligned learning or an effect of more specific design choices. Relatedly, while the MLLMs we study are independently developed, they are all trained on broadly similar web-scale image-text data, so our analysis does not fully rule out dependence on the pretraining data. Fully isolating this effect would require training MLLMs on controlled data mixtures, which we leave to future work.

Mechanisms behind stage-wise shifts. A natural follow-up is to characterize the mechanisms that drive how different visual abilities improve or degrade across MLLM pipeline stages. For example, why does segmentation tend to improve in the LLM layers while monocular depth degrades, and why do some MLLM-trained vision-encoder strengths fail to carry through the projector? A deeper understanding of these stage-wise dynamics could point toward concrete interventions to preserve vision-encoder strengths downstream and to broaden the LLM's ability to support dense prediction.

Unified text and image-generation models. Concurrent work suggests that image generators learn surprisingly general visual features [Gabeur et al., 2026]. Models that train both text and image generation in a single backbone are therefore a natural test bed: image-generation training may compensate for visual properties that text generation steers representations away from, such as fine spatial detail, potentially yielding stronger visual representations than either objective achieves alone.

Stronger starting encoders. Even after MLLM training, MLLM visual representations trail self-supervised baselines such as DINOv3 on every ProbeBench task, with substantial differences across MLLM families that share a language model but use different starting encoders. This points to room for improvement at initialization: instead of defaulting to a language-aligned encoder such as

CLIP, starting from a more visually generalist encoder could meaningfully shift the post-training representation landscape.

LM capacity, scaling, and distillation. Our scaling analysis finds that increasing language-model capacity improves probed visual quality at every pipeline stage, including the vision encoder itself. This offers a new perspective on the existing practice of MLLM distillation: beyond transferring language abilities, distilling from a larger MLLM may help preserve the stronger visual features that develop through large-LM training. Measuring how much of this visual benefit actually transfers to a smaller student is an interesting question for future work.

Using ProbeBench going forward. ProbeBench is useful both as a diagnostic tool and as a benchmark, but the right way to interpret results depends on the purpose. For diagnosing a particular MLLM, e.g., to evaluate training recipes, the vision encoder is itself a meaningful target of analysis. For benchmarking MLLMs against each other on a leaderboard, the encoder's representations are less informative because the LLM never accesses them directly; the projector and downstream stages determine what reaches the language model, and the benchmark score should be computed there.

B ProbeBench Setup

This section provides the full extraction protocol and probe training details used in §2.

B.1 Detailed Probing Setup

Matched pre-MLLM ViT baselines. Each MLLM is paired with the exact pre-MLLM checkpoint it was initialized from: SigLIP2 SO400M for Molmo 2, RICE-ViT-Large for LLaVA-OV 1.5, AIMv2-Huge for SAIL-VL2, and PE-Core-G14 for PerceptionLM. For each pair, the matched ViT baseline is extracted from the same vision encoder layer(s) that the MLLM's projector connects to, ensuring an architecture-matched before/after comparison. This matters in two cases. (i) Molmo 2's vision backbone concatenates SigLIP2 layers $[-3, -9]$ (producing 2304d features), so we evaluate a matched SigLIP2 variant with the same multi-layer concatenation. (ii) PLM's vision encoder is a 47-layer truncation of PE-Core-G14 (50 layers), so we evaluate PE-Core at layer 46 (the last layer of PLM's ViT) rather than the final layer 49. For SAIL-VL2 and LLaVA-OV 1.5, the MLLM and its pre-MLLM ViT both use the final encoder layer, so no adjustment is needed. Following the DINOv3 zero-shot evaluation protocol, we evaluate each matched ViT both with and without the final layer normalization and report the maximum, which can shift correspondence scores meaningfully (e.g. PE-Core-G14 NAVI: 28.1% with final norm, 39.5% without).

MLLM pipeline stages. For each MLLM, we capture features at five pipeline stages:

- *Vision encoder (VE)*: output of the model's ViT, after any internal multi-layer concatenation it performs.
- *Projector (Proj)*: output of the projector module that maps ViT features into the LLM input embedding space.
- *LLM Early/Mid/Late*: residual stream at the corresponding LLM transformer block, captured via a forward hook. We extract at roughly 25%, 50%, and 75% of the LLM's depth (the exact layer indices per model appear in Table 2).

For LLM-internal extraction, we deliberately do *not* apply the model's final RMSNorm, as the norm's parameters are fit to the last layer's distribution and are out-of-distribution at intermediate depths. Image-token positions are identified using the model's image-token ID and reshaped into a 2D spatial grid using each model's tile composition logic.

Table 2: Per-MLLM LLM layer indices (0-indexed transformer blocks) used for early/mid/late extraction.

Model	Total LLM layers	Early / Mid / Late	LLM backbone
Molmo 2 O-7B	32	7 / 15 / 23	OLMo2-7B
PerceptionLM 8B	32	7 / 15 / 23	Llama 3.1-8B
SAIL-VL2 8B	36	8 / 17 / 26	Qwen3-8B
LLaVA-OV 1.5 8B	36	8 / 17 / 26	Qwen3-8B
PerceptionLM 1B	16	3 / 7 / 11	Llama 3.2-1B
PerceptionLM 3B	28	6 / 13 / 20	Llama 3.2-3B
SAIL-VL2 2B	28	6 / 13 / 20	Qwen3-1.7B
LLaVA-OV 1.5 4B	36	8 / 17 / 26	Qwen3-4B
InternVL 3.5 1B	28	6 / 13 / 20	Qwen3-0.6B
InternVL 3.5 2B	28	6 / 13 / 20	Qwen3-1.7B
InternVL 3.5 4B	36	8 / 17 / 26	Qwen3-4B
InternVL 3.5 8B	36	8 / 17 / 26	Qwen3-8B
InternVL 3.5 14B	40	9 / 19 / 29	Qwen3-14B

Resolution control. SAIL-VL2 and PerceptionLM natively use dynamic image tiling (up to 12 and 36 tiles plus a thumbnail respectively), giving each MLLM substantially more pixels per image than its standalone baseline. For all main results, we set `max_tiles=1, skip_thumbnail=True`, so each MLLM processes a single 448×448 image at native feature dimensionality (the same input

resolution as the pre-MLLM encoder). Molmo 2 is resolution-controlled by design: we extract only its global view, matching SigLIP2's single-crop processing. LLaVA-OV 1.5 is already single-resolution by design.

Spatial correspondence in dense probes. For dense prediction tasks (depth, segmentation, surface normals), the feature at position (i, j) must represent the same image region as the ground-truth label at (i, j). This requires that input images are *resized* to the target resolution rather than *center-cropped*. We verified the preprocessing of every model in our study; SigLIP2, PE-Core, RICE-ViT, Molmo 2, SAIL-VL2, and PLM all resize to a square (preserving full spatial coverage), but AIMv2's default HuggingFace processor center-crops by default. For AIMv2, we override the processor with `do_center_crop=False` to disable center-cropping.

B.2 Probe Architectures and Training

Trained probes. Semantic segmentation (ADE20K, 150 classes) uses a single 1×1 convolution preceded by SyncBatchNorm, following the DINOv3 protocol. Depth estimation (NYUv2) and surface normal estimation (NYUv2) use a 4-layer ConvNet decoder ($1 \times 1 \to 3 \times 3 \to 3 \times 3 \to 1 \times 1$) with SyncBatchNorm + GELU activations and hidden dimension 256. Surface normals are predicted as a 4-channel output (n_x, n_y, n_z + uncertainty), trained with an uncertainty-weighted angular loss following Probe3D [El Banani et al., 2024].

Training hyperparameters. All trained probes use AdamW with 4 NVIDIA GPUs (either L40S or A100) and effective batch size 16 (`torchrun -nproc_per_node=4` with batch size 4 per GPU). Depth and segmentation use a learning-rate sweep $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$ and weight-decay sweep $\{10^{-4}, 10^{-3}\}$, training 6 heads simultaneously and selecting the best on validation. Surface normals use a wider sweep ($\{5 \times 10^{-3}, 5 \times 10^{-2}, 0.5\}$ for LR), as we found the optimal LR for the nonlinear normals probe falls outside the segmentation/depth range. We train segmentation for 32 epochs, and depth/normals for 25 epochs, with a warmup-then-cosine schedule, gradient clipping at 35, and SyncBatchNorm.

Zero-shot tasks. Geometric correspondence (NAVI) and semantic correspondence (SPair-71k) are zero-shot evaluations based on cosine similarity between features, following the protocols of El Banani et al. [2024] and Siméoni et al. [2025]. SPair-71k is evaluated as PCK@0.1 across cross-instance pairs, matching bilinearly-interpolated features at the annotated keypoint locations. NAVI is evaluated on multi-view image pairs of the same object: we take the top-1000 cosine-similarity correspondences within the object mask, compute their 3D error from ground-truth camera pose and depth, and report recall averaged over 1, 2, and 5 cm thresholds.

C Additional ProbeBench Results

C.1 Comparison with State-of-the-Art Self-Supervised Features

In §2.2, we noted that despite improvements over their pre-MLLM encoder on many tasks, MLLM features still trail state-of-the-art self-supervised features. Table 3 reports DINOv3 ViT-7B [Siméoni et al., 2025], taken as the current SSL frontier for frozen-feature dense prediction, alongside the best MLLM result for each ProbeBench task. Best-MLLM scores are obtained by selecting the maximum probing score across the four MLLM families and five pipeline stages used in §2.2.

Table 3: **DINOv3 ViT-7B vs. best MLLM probing scores per task.** Best MLLM is the maximum score across the four MLLM families (PerceptionLM, SAIL-VL2, LLaVA-OneVision-1.5, Molmo2) and five pipeline stages (§2.2). DINOv3 leads on every task. Correspondence is evaluated zero-shot.

	Sem. Seg. (ADE20K) ↑	Sem. Corr. (SPair) ↑	Geom. Corr. (NAVI) ↑	Normals (NYUv2) ↓	Depth (NYUv2) ↓
DINOv3 ViT-7B	**53.2**	**59.4**	**59.4**	**28.4**	**0.295**
Best MLLM	45.8	45.9	51.5	29.5	0.355
Δ (DINOv3 advantage)	$+7.4$	$+13.5$	$+7.9$	$+1.1$	$+0.060$

DINOv3's advantage spans all five tasks, ranging from $+1.1^\circ$ on surface normals to $+13.5$ PCK on semantic correspondence. The semantic correspondence gap is particularly notable: even the best MLLM stage trails DINOv3 by more than 13 PCK points despite the language modeling objective sharpening semantic features (Finding 1). This indicates that current language-supervised vision encoders, even after extensive MLLM training, do not yet match self-supervised dense-prediction quality at scale.

C.2 Controlling for Projector Spatial Downsampling

Since all four studied MLLMs reduce each 2×2 block of vision encoder tokens to a single token in the projector, differences between the vision encoder and later stages could partly reflect this spatial downsampling. To separate it from changes in feature quality, here we pool vision encoder features to the projector's token grid with an operation matched to each projector. For PerceptionLM (adaptive average pooling) and Molmo2 (attention pooling), we apply average pooling. For SAIL-VL2 (pixel shuffle) and LLaVA-OneVision-1.5 (patch merging), which concatenate each block along the channel dimension before the projector's MLP (trained during MLLM training), we apply the same concatenation for trained probes. For zero-shot correspondence, we use average pooling for all models, since without the projector's MLP, concatenation places each patch in a position-dependent channel slot and breaks cosine matching.

Table 4: **Vision encoder performance with projector pooling.** While pooling lowers vision encoder performance in most model–task pairs, the pooled vision encoder still outperforms the projector in 15 of 20.

	Sem. Seg. (ADE20K) ↑	Sem. Corr. (SPair) ↑	Geom. Corr. (NAVI) ↑	Normals (NYUv2) ↓	Depth (NYUv2) ↓
PerceptionLM-8B					
Vision encoder	42.4	41.0	46.3	29.5	0.355
Vision encoder (pooled)	44.9	35.8	46.9	31.9	0.371
Projector	44.4	38.4	44.9	32.0	0.370
LLM late	43.6	33.8	42.8	33.1	0.387
SAIL-VL2-8B					
Vision encoder	36.0	27.8	41.1	32.9	0.426
Vision encoder (pooled)	39.7	22.7	39.9	34.2	0.425
Projector	37.1	28.2	41.9	35.1	0.440
LLM late	37.3	23.7	41.1	36.0	0.457
LLaVA-OneVision-1.5-8B					
Vision encoder	38.6	37.2	41.0	32.5	0.373
Vision encoder (pooled)	42.9	35.3	41.7	33.2	0.369
Projector	36.9	35.4	41.2	34.1	0.393
LLM late	41.0	35.1	40.1	34.7	0.396
Molmo2-O-7B					
Vision encoder	44.4	45.9	51.5	30.5	0.392
Vision encoder (pooled)	44.4	35.5	48.6	32.9	0.410
Projector	36.6	34.6	45.1	33.4	0.430
LLM late	35.7	26.3	41.4	35.0	0.443

As shown in Table 4, pooling lowers vision encoder performance in most model–task pairs. Even so, the pooled vision encoder remains stronger than the projector in 15 of 20 model–task pairs, and four of the five exceptions are zero-shot correspondence, where average pooling is only a stand-in for the projector's learned merging. For Molmo2, the model with by far the largest drop from vision encoder to projector, a substantial gap remains between the pooled vision encoder and the projector on segmentation (44.4 vs. 36.6), geometric correspondence (48.6 vs. 45.1), and depth (0.410 vs. 0.430), so this drop is not explained by compression alone. Comparing the projector to the late LLM stage, which processes the same number of tokens, we see further degradation in 18 of 20 model–task pairs. The two exceptions are segmentation on SAIL-VL2 and LLaVA-OneVision-1.5, where the best representations emerge inside the LLM, in line with Finding 2.

C.3 Multi-Tile Input

Our main protocol sets each MLLM to a single tile, so that the MLLM and its pre-MLLM vision encoder process the same input resolution (§2.1). In practice, however, these models are often deployed with dynamic tiling. To test whether our pipeline-stage findings hold in this higher-resolution setting, we additionally probe PerceptionLM on depth estimation, where single-tile performance degrades across LLM stages, using 4 tiles plus the thumbnail (5 in total). Following PerceptionLM's native processing, the thumbnail passes through the full pipeline, and its features are upsampled and concatenated channel-wise with the stitched tile features. Probes follow Appendix B.2, with a learning-rate sweep of $\{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$.

Table 5: **Depth probing on PerceptionLM-8B with multi-tile input** (NYUv2 RMSE ↓). Increasing the tile count improves the vision encoder and projector, but the degradation across LLM stages persists.

Stage	1 tile	4 tiles + thumbnail
Vision encoder	0.355	0.346
Projector	0.370	0.347
LLM early (layer 7)	0.375	0.355
LLM mid (layer 15)	0.379	0.375
LLM late (layer 23)	0.387	0.386

As shown in Table 5, increasing the tile count improves depth estimation for the vision encoder and projector stages, but the drop-off across LLM stages persists: with 4 tiles plus the thumbnail, RMSE rises from 0.346 at the vision encoder to 0.386 at the late LLM stage, nearly matching the single-tile late stage.

C.4 Per-Stage Probing Across Language Model Sizes

In §2.3, we report the best probing score across pipeline stages for each (model, task) pair, finding that scaling the language model improves probed visual quality across all four families and all five tasks. Here, we provide the full per-stage breakdown that supports the claim that these gains extend across the full pipeline rather than being localized to LLM layers (where capacity grows directly with model size).

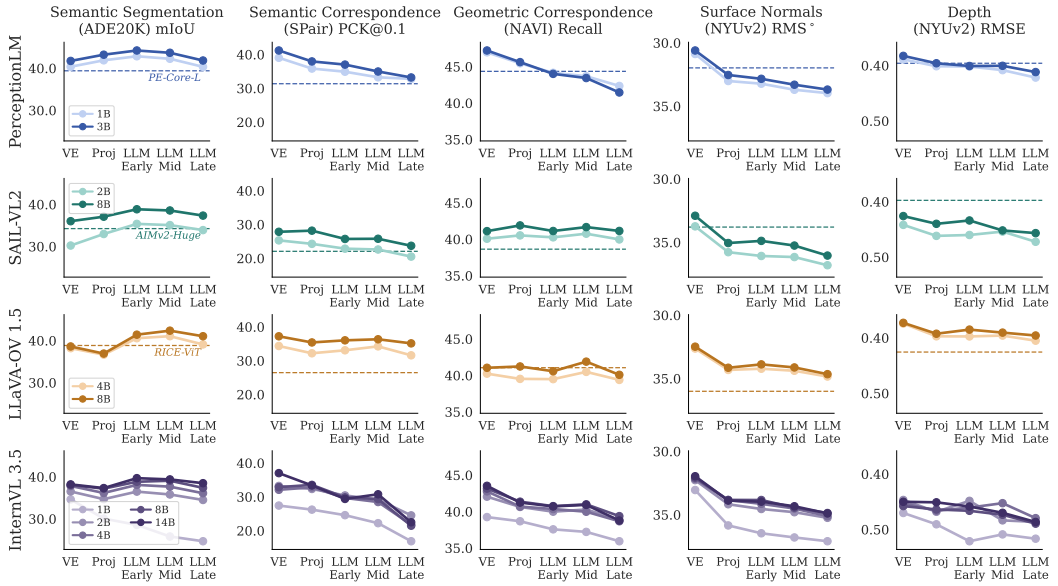

Figure 6: **Per-stage probing quality across the four scaling families.** Each row is one MLLM family, each column is one task. Within each subplot, lines trace probe scores across the five pipeline stages (vision encoder → projector → LLM Early/Mid/Late), with one line per LM size in that family (lighter to darker = smaller to larger LM). Larger LMs improve probed quality at nearly every pipeline stage, including the vision encoder itself, indicating that LM-scale gains propagate throughout the model rather than being confined to LLM-stage processing. Dashed lines mark the corresponding pre-MLLM vision encoder (PE-Core-L for PerceptionLM, AIMv2-H for SAIL-VL2, RICE-ViT for LLaVA-OneVision-1.5); InternVL3.5 has none, as its encoder was itself trained with next-token prediction within InternVL2.5.

We highlight two patterns. First, scaling the LM improves probed quality at nearly every pipeline stage, including the vision encoder itself. This is notable because the vision encoder architecture and initialization are held fixed across model sizes within each family, so any improvement at the vision encoder stage reflects gains the joint MLLM training propagates back to the underlying visual representations rather than additional LLM-side processing of fixed features. Second, the layerwise shape (the relative ordering of pipeline stages by probing score) remains broadly consistent within a family across model sizes, with LM scaling lifting the entire curve rather than reshaping which stage is strongest. Together, these patterns reinforce the §2.3 finding that LM capacity itself bottlenecks visual representation learning in MLLMs.

D Examining ProbeBench Relation to MLLM Benchmarking

MLLMs are trained and deployed to answer questions, so a natural question is how the representation-level effects measured by ProbeBench relate to end-to-end VQA performance, and in particular whether existing MLLM benchmarks already capture effects such as the degradation of monocular depth. While MLLM benchmarking has long been hindered by models exploiting linguistic priors and dataset biases, benchmarks such as CV-Bench [Tong et al., 2024] have emerged specifically to assess vision-centric MLLM capabilities. Given the feature-level degradation we identify in the probe setting, we would expect these models to suffer here too, at least on the depth subtask. As shown in Table 6, however, all evaluated MLLMs from §2.2 achieve strong end-to-end performance on this benchmark, including on Depth Order, which targets the same fine-grained depth abilities we found to degrade.

	CV-Bench				
Model	2D Spatial	2D Count	3D Depth	3D Distance	Average
PerceptionLM	89.4	70.9	88.3	75.7	81.1
SAIL-VL2	90.0	72.7	85.2	84.3	83.1
LLaVA-OV 1.5	84.0	71.2	84.3	82.3	80.5
Molmo2	90.3	76.5	84.8	82.5	83.5

Table 6: **Standard end-to-end MLLM evaluation on CV-Bench.** The studied models all achieve strong scores, even on 3D Depth, where we found feature degradation in the probing setting for $3/4$ models.

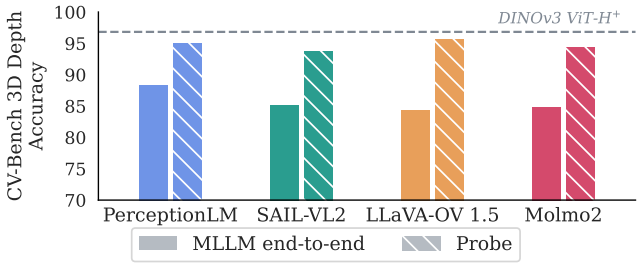

Figure 7: **Probe-based vs. end-to-end MLLM evaluation on CV-Bench 3D Depth.** A depth probe applied to MLLM features outperforms its end-to-end answer for every model, reaching close to 100% performance. This saturation does not reflect strong underlying depth features (see §2.2), but rather a limitation of CV-Bench 3D in assessing fine-grained depth ability.

In this section, we test the following contrasting explanations:

1. MLLMs process visual representations through computations that probing cannot recover, allowing for strong end-to-end performance through pathways our probe-level analyses miss.
2. Even a vision-centric benchmark like CV-Bench 3D does not assess fine-grained enough geometric information to expose the feature-level shortcomings we identified, allowing MLLMs with suboptimal depth representations to still succeed.

To distinguish these, we test whether probes from ProbeBench can match the MLLM's end-to-end performance on this benchmark. If probes fall short, that would indicate MLLMs are extracting depth through computations ProbeBench cannot capture. Conversely, if probes match or exceed end-to-end performance, that would demonstrate a limitation of the benchmark itself, validating ProbeBench as a diagnostic tool for visual representations rather than their downstream utility through existing benchmarks. Concretely, we focus on Depth Order from CV-Bench 3D: each example provides an image with two bounding-box-annotated objects and asks which is closer to the camera. We use the depth probe paired with each MLLM's best post-encoder layer to predict a per-patch depth map on the image, and answer each question by extracting the predicted depth values within each bounding box and selecting the box with the smaller median depth (following Fu et al. [2025]). Full details are in Appendix D.1 below.

As shown in Figure 7, the probe not only matches MLLM end-to-end performance but exceeds it for every model, with accuracy rising from the 84.3–88.3% end-to-end range to a 93.7–95.7% probed range. This gap does not reflect probes themselves being particularly strong: in §2.2 we showed that the underlying MLLM visual representations are degraded relative to the standalone vision encoder. Rather, it reflects that CV-Bench 3D does not adequately assess fine-grained depth ability.

These findings verify that the depth degradation surfaced by our probing analyses is not an artifact of probing failing to capture latent MLLM capability, but a real degradation that CV-Bench 3D fails

to expose. This indicates that solely measuring end-to-end accuracy does not provide a complete assessment of an MLLM's visual representation quality, confirming the utility of analyzing MLLMs as visual foundation models through ProbeBench.

D.1 CV-Bench 3D Depth Probe Setup

We provide additional details on the probe-based CV-Bench 3D Depth evaluation described above.

CV-Bench 3D Depth contains 600 questions in total, with 200 each drawn from Hypersim, SUN RGB-D, and nuScenes. The released benchmark images have the question's two ground-truth bounding boxes drawn directly on the image as colored rectangles, since this is the input format consumed by an MLLM during end-to-end evaluation. Applying our probes to these images would introduce a distribution shift at the annotated pixels, since the probes were trained on un-annotated dense-prediction inputs. To match the input distribution the probes were trained on, we recover the clean (un-annotated) source image for each question by mapping every CV-Bench example back to its source-dataset image identifier and loading the corresponding original image from Hypersim, SUN RGB-D, or nuScenes. The two bounding-box coordinates per question are obtained directly from the CV-Bench metadata, so the probe is given the same regions the MLLM is asked about, only without the rectangles drawn on the image.

For each MLLM, we use the best post-encoder probe layer identified in §2.2, paired with the ConvNet depth head trained at that layer. To match the visual domain of each question, we apply a domain-specific probe: probes trained on NYUv2 are used for the indoor questions (Hypersim and SUN RGB-D), and a probe trained on KITTI is used for the driving questions (nuScenes). The KITTI probes follow exactly the training recipe used for the NYUv2 probes in §2.2, with the only change being the training dataset. For the DINOv3 ViT-H$^+$ vision-foundation reference, we apply the same domain-specific probe selection so that all reported numbers are directly comparable.

At inference, each model takes the clean source image as input and produces a feature map at its native patch resolution, which the depth probe converts to a per-patch depth prediction in metric units. We bilinearly upsample the resulting depth map to the original image resolution to obtain a per-pixel depth prediction over the full image. For each of the question's two ground-truth bounding boxes, we then extract the predicted depth values at the pixel coordinates falling inside the box and aggregate them by taking the median predicted depth within the box.

We resolve each question of the form "which of regions A or B is closer to the camera?" by selecting the region with the smaller median predicted depth, producing a binary answer that we compare against the CV-Bench ground-truth answer to obtain per-question accuracy. We report CV-Bench 3D Depth accuracy as the unweighted average over all 600 questions (n=200 per source).

E SLIM Architecture, Training, and Additional Results

Latent grid and stages. The spatial latent grid matches each model's ViT patch grid: 32×32 for PerceptionLM-8B and SAIL-VL2-8B (448×448 input, patch size 14), and the variable, aspect-preserving grid of LLaVA-OneVision-1.5's native-resolution encoder. The latent dimension is D=512. We aggregate over 9 stages: 5 uniformly-spaced ViT layers, the projector, and 3 LLM layers. For PerceptionLM-8B these are ViT layers $\{9, 18, 27, 36, 46\}$ (depth-matched to its 47-layer ViT) and LLM layers $\{7, 15, 23\}$; for SAIL-VL2-8B and LLaVA-OneVision-1.5-8B, they are ViT layers $\{4, 9, 14, 19, 23\}$ of their 24-layer encoders and LLM layers $\{8, 17, 26\}$.

Cross-attention head. Each cross-attention block uses 8 heads, head dimension 64, residual + LayerNorm. The shared projection W is implemented as one $\mathrm{Linear}(d_{\text{stage}} \to D)$ per input dimension class (e.g., one shared projection for all ViT stages at 1536d, one for all LLM stages at 4096d), and per-stage FiLM parameters (γ_s, β_s) are applied immediately after.

Adaptive depth gating. A single gate is placed in the SLIM stack between the ViT-stage cross-attention and the projector + LLM-stage cross-attention. The gate is a linear projection of the latent $Q_{i,j}$ followed by a sigmoid, trained with a straight-through estimator (forward hard, backward soft). The sparsity loss is $\mathcal{L}_{\text{sparse}} = \lambda \cdot \mathrm{mean}(g)$, where g is the soft gate. We use λ=0.03 for segmentation and semantic correspondence (0.01 for LLaVA-OneVision-1.5 correspondence) and λ=0.01 for depth and surface normals (0.03 for LLaVA-OneVision-1.5 normals, where λ=0.01 learned no pruning); Appendix E.1 additionally uses λ=0.10 for segmentation. At inference, the gate is hard-thresholded and pruned tokens are masked out of the LLM's self-attention so they cannot be attended to.

Training. SLIM is trained end-to-end with the task loss plus the sparsity regularizer, using AdamW with batch size 16 across 4 GPUs. Ungated models sweep learning rate $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$ and weight decay $\{10^{-4}, 10^{-3}\}$ (surface normals use learning rates $\{5 \times 10^{-3}, 5 \times 10^{-2}, 0.5\}$, as in §2). Gated models use learning rate 3×10^{-4} for segmentation and 5×10^{-2} for surface normals, with weight decay 10^{-4}, and the full sweep for depth. We train for 20 epochs on ADE20K and 25 epochs on NYUv2 depth and surface normals. The MLLM backbone is frozen throughout; only the SLIM head, gate, and task decoder are trained.

Correspondence protocol. For semantic correspondence, we follow the supervised protocol of Luo et al. [2023]. The SLIM head is trained on SPair-71k training pairs with a symmetric cross-entropy objective: the latent sampled at a source keypoint is matched by cosine similarity (temperature 0.07) against all latents on the target grid, mirroring the argmax-cosine rule used at evaluation. We train for 3 epochs of 8,000 pairs at 448×448 with the same learning-rate and weight-decay sweep, select the best head on held-out validation pairs, and report PCK@0.1 on 200 test pairs per category. For geometric correspondence, the trained head is applied zero-shot to NAVI. The single-stage baselines (pre-MLLM vision encoder, MLLM best stage, and general-purpose encoders) are trained and evaluated identically; for the MLLM best stage we train on the vision encoder stage, which under zero-shot evaluation is the strongest MLLM stage for correspondence or within 1 point of it (Figure 1). For gated models, evaluation uses the same two-pass protocol as the other tasks, with pruned tokens removed from the LLM's forward pass.

E.1 Comparison with VisionZip

We compare SLIM's learned gate with VisionZip [Yang et al., 2025b], a strong MLLM token-pruning method that does not require text-guided selection. Starting from the ungated SLIM head, we prune LLM image tokens per image using attention scores from the penultimate ViT block, at the same fraction of LLM tokens that the gated SLIM model prunes, and evaluate with the same two-pass protocol. We run this comparison on ADE20K semantic segmentation, where we find that LLM-stage features are most useful.

As shown in Table 7, at moderate pruning (λ=0.03, $\sim$60% of LLM tokens pruned), SLIM matches or exceeds VisionZip, and at aggressive pruning (λ=0.10, $\sim$90%), it outperforms VisionZip on all three models, by 0.7–5.8 mIoU. The gap widens as the token budget shrinks, indicating that a task-trained selector becomes increasingly important under aggressive pruning.

Table 7: **SLIM's learned gate vs. VisionZip at matched prune rates** on ADE20K semantic segmentation (mIoU ↑). Parentheses give the fraction of LLM tokens pruned by SLIM, which VisionZip matches.

	PerceptionLM-8B	SAIL-VL2-8B	LLaVA-OneVision-1.5-8B
SLIM (ungated)	50.9	48.5	49.5
λ=0.03			
SLIM	**50.6** (60%)	48.2 (58%)	**49.3** (58%)
VisionZip	50.5	**48.3**	48.2
λ=0.10			
SLIM	**49.5** (89%)	**47.3** (93%)	**47.8** (89%)
VisionZip	48.8	45.4	42.0

Gate interpretability. To assess whether the learned gate is task-informed, we analyze the gated PerceptionLM SLIM model for ADE20K segmentation (λ=0.03) and compute per-class prune rates over the full validation set, assigning each LLM-token position the majority ground-truth class of its region (requiring at least half the region to belong to that class). Restricting to classes covering more than 1,000 positions, the most-pruned classes are sky (97.5%), skyscraper (88.0%), ceiling (87.5%), building (85.3%), and tree (80.5%), all large and visually homogeneous regions, while the least-pruned are display case (1.5%), pedestal (2.6%), seat (3.3%), shelf (8.3%), and box (8.6%), which are compact and object-centric. Aggregated by category, the gate keeps object ("thing") regions twice as often as "stuff" regions (55.6% vs. 27.9% of positions kept), indicating that it learns an interpretable spatial pattern that routes object-centric regions through the LLM. Figure 8 shows qualitative examples.

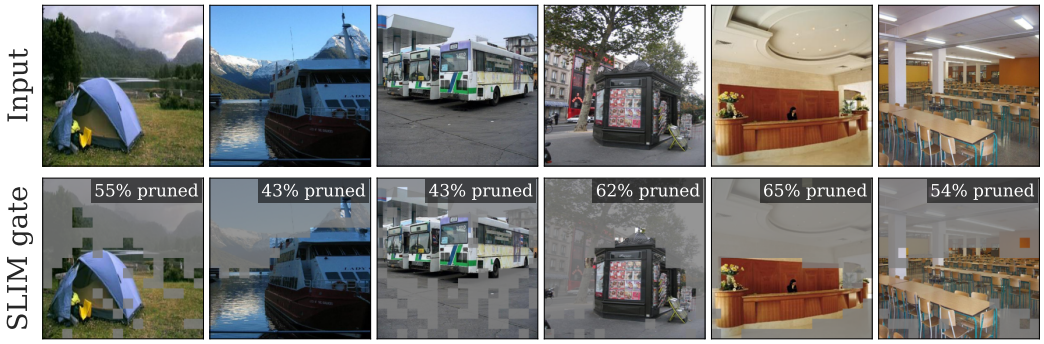

Figure 8: **Qualitative SLIM gate decisions** (PerceptionLM-8B, λ=0.03, ADE20K val). Pruned LLM tokens are masked in gray; kept tokens are shown in full color. The gate prunes large, homogeneous regions such as sky, trees, buildings, and ceilings, and keeps tokens covering objects.

F Image Generation Setup

We provide architectural and training details for the Representation Autoencoder (RAE) image generation experiments described in §4.1, complementing the main-paper setup.

Background on representation autoencoders. Conventional latent diffusion models diffuse in the latent space of a VAE [Rombach et al., 2022], whose encoder is trained purely for reconstruction and produces low-dimensional, semantically weak features. RAEs [Zheng et al., 2025] instead pair a frozen pretrained representation encoder (e.g., DINOv2, SigLIP2, MAE) with a trained ViT decoder, preserving the semantic structure of the underlying encoder while still permitting high-fidelity reconstruction. Diffusion transformers trained on RAE latents converge faster and reach lower FID than their VAE-based counterparts, making RAE a natural framework for evaluating any frozen vision encoder as a generative substrate. RAE has so far been instantiated only with self- or language-supervised vision encoders trained on image-only or image-text contrastive objectives; MLLM vision encoders — i.e., the vision backbone of an MLLM after end-to-end multimodal training — are an unexplored but natural candidate, since stronger semantics may aid conditional generation while the loss of fine-grained geometric structure could harm reconstruction fidelity.

Stage 1: ViT decoder. The frozen encoder (PE-Core-L or PLM-VE) produces $32 \times 32 \times 1024$ latents from 448×448 inputs (patch size 14). We train a ViT-B decoder with patch size 8 to reconstruct 256×256 ImageNet images, optimizing ℓ_1 + LPIPS + GAN losses; the GAN discriminator is activated only after epoch 4 to allow stable initial reconstruction. We additionally use noise-augmented decoding at $\tau{=}0.8$ to robustify the decoder to small latent perturbations introduced by stage 2. Training runs for 8 epochs at global batch size 64.

Stage 2: Diffusion transformer. We train a class-conditional diffusion transformer on the stage-1 latents using flow matching with a dimension-dependent noise schedule, linear interpolation between data and noise, and logit-normal time sampling. We adopt the DiT^{DH} architecture [Wang et al., 2025a, Zheng et al., 2025]: a DiT-S backbone augmented with a wide, shallow DDT head (2 layers, hidden dimension 2048). Training runs for 80 epochs at global batch size 256. At sampling time, we use Euler ODE integration with 50 steps.

Compute caveat. The above configuration is downscaled relative to Zheng et al. [2025] (smaller decoder and DiT backbone, fewer epochs, smaller batch sizes), so our results are not directly comparable to their headline numbers; the goal is the *controlled, relative* comparison between PE-Core-L and PLM-VE under an identical recipe.

Evaluation. We report reconstruction FID (rFID) at the end of stage 1 and generative FID (gFID) at the end of stage 2, both on 4096 samples drawn from random ImageNet class labels (no guidance). We additionally report gFID with classifier-free guidance at scale 1.5 applied identically to both encoders.

G Inverse Dynamics Modeling Setup

We provide additional details on the Minecraft VPT inverse dynamics modeling experiments described in §4.2.

Dataset. We use a 300-recording subset of the publicly released Minecraft VPT contractor data [Baker et al., 2022], which after filtering for corrupted or incomplete recordings yields ~ 241 usable trajectories. Each recording is roughly 5 minutes of gameplay at 20 fps ($\sim$ 6,000 frames per recording). The ~ 241 recordings ($\sim$ 1.4M frames, $\sim$ 20 hours) are split $90/10$ into train and validation, giving ~ 217 recordings ($\sim$ 1.3M frames, $\sim$ 18 hours) for training. We additionally use a separately-released held-out test set of 30 recordings ($\sim$ 180K frames, ~ 2.5 hours) on which all reported AUROC numbers are computed; the train/val pool never sees these recordings. Each example is a T=16 contiguous-frame clip sampled uniformly from a recording, with frames resized to 448×448 to match the input resolution used by PerceptionLM and PE-Core in the rest of the paper. The target is the 22-dimensional VPT action vector at each timestep: 20 binary keyboard/mouse-button channels and 2 continuous mouse camera deltas $(\Delta x, \Delta y)$.

Frozen-backbone protocol. Following §2, the visual model is frozen throughout. For each clip, we extract per-frame features $F \in \mathbb{R}^{T \times N \times D}$, where N is the number of spatial tokens and D is the feature dimension at the chosen pipeline stage. PerceptionLM stages requiring multi-frame LLM context (LLM-Early/Mid/Late) are run in video mode, where the LLM processes the T frames jointly within its image-token sequence; the vision-encoder and projector stages are run per-frame.

Head architecture. The head matches the bidirectional "non-causal IDM" protocol of Baker et al. [2022]. (i) *Per-frame pooling*: a V-JEPA-style attentive pooler with a single learnable query $q \in \mathbb{R}^{1 \times D_{\text{pool}}}$ cross-attends to the N spatial tokens of each frame, producing one pooled token per frame. (ii) *Temporal transformer*: a 4-layer bidirectional transformer (d_{model}=512, 8 heads, MLP ratio 4, dropout 0) over the T pooled tokens, with a learned 1D positional embedding ($T_{\max}$=32). (iii) *Action head*: a per-position MLP that maps each temporal-transformer output to the 22-dimensional action vector (20 key logits + 2 mouse deltas). Only the head is trained; the backbone is frozen.

Loss. Mouse-delta targets are normalized using per-channel mean and standard deviation computed on the training split (keys are unnormalized). The training loss is

$$\mathcal{L} = \text{BCE}(\text{key_logits}, \text{key_targets}) + 0.1 \cdot \text{MSE}(\text{mouse_pred}_{\text{norm}}, \text{mouse_target}_{\text{norm}}),$$

with all per-frame losses masked by per-clip valid-frame indicators and averaged over valid frames. The mouse weight (0.1) keeps the BCE key signal from being swamped by mouse-channel gradients.

Training. We train each (model, stage) configuration for 1 epoch with AdamW (learning rate 3×10^{-4}, weight decay 10^{-4}), batch size 1 per GPU across 4 GPUs (effective batch 4), and 4 data-loading workers.

Evaluation. We report mean AUROC across the 20 binary key channels on the held-out test split (the 30 separately-released recordings described above), computed per-channel and then averaged. To quantify uncertainty in the comparison between configurations, we use a paired bootstrap (B=10,000) clustered at the recording level: each resample draws test recordings with replacement, so that correlated clips from the same recording are kept together, and both configurations are scored on the same resampled recordings. We report the 95% confidence interval of the resulting difference in mean AUROC. Mouse-delta accuracy is not reported as a primary metric: in our relatively compute-limited setup, we found that all setups performed near-randomly on these continuous channels, with no meaningful separation across pipeline stages or against the PE-Core baseline. The reported AUROC therefore focuses on the discrete keyboard/mouse-button actions, where signal does emerge.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We have detailed all claims made in the abstract and introduction in §2, §3, and §4

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss limitations of our work in the discussion section (Appendix A).

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: We do not include theory assumptions or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have provided code for reproducing our experimental results in the supplement.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will publicly release all code to reproduce results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All the experimental settings and details are contained in the main paper and Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: In line with foundation model probing literature (e.g., DINOv2 [Oquab et al., 2023], DINOv3 [Siméoni et al., 2025], Probe3D [El Banani et al., 2024]), we do not report error bars. The benchmark involves training a probe for each (model, pipeline stage, task) combination across 4 MLLM families, 5 pipeline stages, and 5 tasks, plus self-supervised and pre-MLLM encoder baselines, which makes multi-seed reruns of every configuration prohibitive in our compute setting. To ensure reported numbers are not artifacts of a single hyperparameter choice, we sweep over multiple learning rates and weight decays per configuration and select the best-validation head.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We include information about compute resources in Appendix B.2.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The paper is foundational research analyzing the visual representations inside multimodal LLMs; there is no direct path to specific deployments or applications with predictable societal impact.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Our work only uses publicly released MLLMs and data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have properly credited all assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We have documented how to setup ProbeBench and SLIM in the code.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

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