FROM PIXELS TO ACTIONS – TOWARDS NATIVE VISION-LANGUAGE-ACTION POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action models are typically built on a simple premise: perception is first compressed by visual encoders, and action reasoning begins only afterwards. Yet manipulation often hinges on precisely the visual signals most vulnerable to this abstraction, e.g., local geometry, boundaries, depth, and subtle state changes. This raises a fundamental question: *should action policies inherit a fixed visual endpoint, or directly access evolving visual computation?* We introduce **NEO-VLA**, a structurally encoder-free vision-language-action framework. Images are tokenized by lightweight patch convolutional layers and evolve jointly with language throughout the backbone, allowing a flow-matching action expert to draw from instruction-conditioned visual representations at different depths. This simple change reveals a consistent pattern. Native architectures outperform encoder-based counterparts under controlled comparisons; full fine-tuning reduces but does not erase the gap, and additional scale alone cannot recover it. More strikingly, layer-wise probing shows that the action-critical signals often peak well before the final visual representation. Across diverse manipulation settings and distribution shifts, NEO-VLA consequently achieves stronger and more robust control. Together, these results suggest that embodied control benefits from access to visual representations across the computational hierarchy, rather than a fixed encoder output.

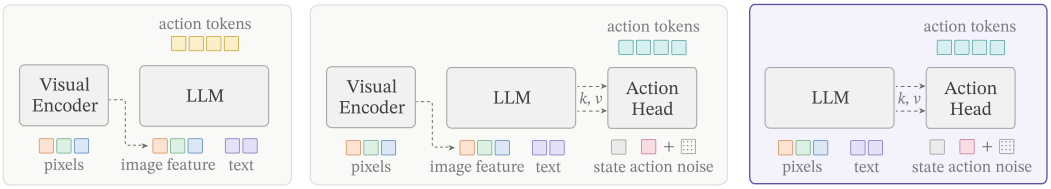
1 INTRODUCTION

Robotic manipulation demands both semantic understanding and fine-grained spatial reasoning. A policy must identify *what* an instruction refers to, infer *where* the target lies, and determine *how* the gripper should move relative to it. These requirements raise a fundamental question for vision-language-action (VLA) models: *should action prediction rely only on the final visual representation, or also on the intermediate computation that produces it?*

Most existing VLAs make this choice implicitly. Images are first processed by a separately pretrained vision encoder, such as CLIP, SigLIP, or DINOv2 (Radford et al., 2021; Zhai et al., 2023; Oquab et al., 2023), before their features are passed to the language backbone and action model (Brohan et al., 2023; Kim et al., 2024; Black et al., 2024; Bjorck et al., 2025). This interface exposes the encoder output while hiding the visual hierarchy that produced it. Recent studies increasingly point to this interface as a potential bottleneck: visual processing is critical to VLA performance (Zhang et al., 2026b), visual components are especially sensitive to pruning (Zhang et al., 2026a), and action fine-tuning can degrade visual representations (Kachaev et al., 2025). Besides, UAM (Zhang et al., 2026c) addresses this issue by adding a second visual pathway. We instead revisit the interface itself: rather than augmenting the representation delivered by a vision encoder, we allow the action expert to read directly from visual computation inside the backbone.

We introduce NEO-VLA, a structurally native VLA policy in Figure 1 built directly on pixels and words. Following encoder-free multimodal architectures (Bavishi et al., 2023; Diao et al., 2024; 2025; 2026a;c), a lightweight convolutional patch embedder maps raw pixels into the backbone token space, where image patches and instruction words are processed jointly within a single holistic architecture. A flow-matching action expert then reads cached keys and values from diverse backbone layers with different representation hierarchies to predict continuous action chunks. Here, *structurally native* refers strictly to the architecture: NEO-VLA uses no separately pretrained vision encoder, image codebook, or autoencoder, while the backbone itself is pretrained on web-scale image-text data.

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



(a) Encoder-based VLA + Quantised Actions (b) Encoder-based VLA + Action Flow **(c) Encoder-free VLA + Action Flow**

Figure 1: **Three VLA policy designs**: encoder-based vs. encoder-free backbones, and discrete vs. continuous actions. **(a)** A separately pretrained visual encoder converts pixels into image features, which are consumed by the language backbone together with text tokens; actions are generated as discrete tokens. **(b)** The same encoder-based visual interface is retained, while a separate action head predicts continuous actions conditioned on the backbone. **(c)** NEO-VLA removes the separately pretrained visual encoder: a lightweight patch embedder maps pixels directly into the backbone token space, while text tokens enter the backbone jointly. A narrow action expert then reads cached keys and values from selected backbone layers and predicts a flow-matching velocity field.

This design turns visual readout depth into an explicit policy choice. The action expert can access early representations, late representations, or combinations of both, without introducing a separate visual pathway. In conventional encoder-based VLAs, by contrast, even reading from different language-backbone layers leaves the policy downstream of a fixed visual endpoint. NEO-VLA therefore provides a direct way to ask which stages of visual computation matter most for control.

Our analysis reveals a clear depth-dependent pattern. With a frozen backbone, readout configurations that expose early layers consistently outperform those restricted to later representations, while after full fine-tuning, combining early and late layers yields the strongest performance. Targeted interventions further reinforce this trend: swapping early-layer activations across observations produces substantially larger prediction errors than perturbing later layers. These findings suggest that action-critical information emerges well before the final visual representation, where fine-grained spatial and geometric cues remain more directly accessible. Preserving access to these intermediate computations is therefore particularly valuable for precise and generalizable control.

We further compare it with Qwen3-VL-2B and Gemma-3-4B with comparable understanding capabilities, under the same action expert and robot-training corpus. It consistently achieves higher LIBERO success rates in both frozen-backbone and full-finetuning settings. This is particularly notable in the frozen setting, where policy learning must rely directly on the pretrained visual representations. While pretraining and capacity differences prevent isolating encoder removal alone, the comparison establishes the native backbone as a strong and transferable foundation for action learning.

We then evaluate NEO-VLA on LIBERO, LIBERO-plus, and RoboTwin 2.0, covering both standard manipulation and challenging distribution shifts in visual appearance, language, and environment. NEO-VLA achieves strong in-distribution performance while maintaining robust control under these shifts, with particularly pronounced gains under camera perturbations and image noise. These results complement the layer-wise analysis by showing that preserving access to intermediate visual computation is not only useful in controlled ablations, but also translates into practical robustness across diverse manipulation settings. Taken together, our findings point to a broader principle for embodied intelligence: effective control depends not only on what the backbone represents, but also on which stages of its visual computation remain accessible to action.

Our main contributions are summarised as follows.

- **A structurally native VLA policy.** We introduce NEO-VLA, which combines a lightweight pixel-to-token embedder, a shared vision-language backbone, and a flow-matching action expert with configurable access to backbone layers with diverse visual hierarchies.
- **A systematic empirical study of action readout depth.** Layer-map comparisons with matched expert capacity, together with attachment-point interventions, show that early visual representations are pretty effective for control, particularly when the backbone is frozen.
- **Evaluation across backbone families and manipulation benchmarks.** We compare native and encoder-based backbones using the same action-expert design and robot-training corpus, and show strong performance on LIBERO, LIBERO-plus, and RoboTwin 2.0.

2 Related Work

Vision-Language-Action Models. Existing VLA policies typically attach an action decoder to a pretrained vision-language model. Early systems emit actions as discrete text tokens (Brohan et al., 2023; Kim et al., 2024), while later work uses continuous diffusion or flow-matching decoders over backbone features (Black et al., 2024; Intelligence et al., 2025; Bjorck et al., 2025); Team et al. (2024) instead trains a transformer policy from scratch with a diffusion head. As shown in Figure 1, our design differs along two axes: whether the visual pathway introduces a separately pretrained vision encoder, and whether actions are represented discretely or continuously.

Most prior work has focused on the action side while leaving the visual pathway largely unchanged. Images are commonly processed by pretrained encoders such as CLIP, SigLIP, or DINOv2 (Radford et al., 2021; Zhai et al., 2023; Oquab et al., 2023), often at a fixed resolution and sometimes with the encoder frozen during policy training. Recent studies suggest that this front end can limit control: Zhang et al. (2026b) identify the visual module as a major bottleneck, Zhang et al. (2026a) find the vision encoder, language backbone, and projector among the components most sensitive to pruning, and Kachaev et al. (2025) show that action finetuning can degrade visual representations.

Recent unified systems also tend to preserve the visual front end, adding modality-specific experts for generation, foresight, or world modelling above a shared attention stream (Liang et al., 2024; Cai et al., 2026; Ma et al., 2026; Lv et al., 2025; Qu et al., 2025; Hu et al., 2026; Cen et al., 2025; Liang et al., 2026; Liu et al., 2026). Zhang et al. (2026c) instead adds a second visual pathway, arguing that one encoder may not serve both language grounding and control well. Our focus is different: we vary the visual pathway itself while keeping the action expert, the layers it reads, the training data.

Native Multimodal Models. A separate line of work removes the vision encoder itself, mapping pixels through a lightweight learned patch embedder directly into the language backbone (Bavishi et al., 2023; Diao et al., 2024; 2025; Luo et al., 2025; Diao et al., 2026a;c;d;b; Lab, 2026). In this literature, *native* can refer to different ideas: removing a separate vision module, training multimodally from the start (Tong et al., 2026; Han et al., 2026), or preserving the backbone's original format and objectives during policy training (Ma et al., 2026). We use the term only in the structural sense: no separately trained vision module, image codebook, or autoencoder.

Control models have evolved along related lines. Liu et al. (2025) feed raw image patches directly into a language backbone, Wang et al. (2026) represent vision, language, and action as unified discrete token sequences, and Xie et al. (2026) replace the transformer-based vision encoder with a convolutional alternative. Concurrent work studies understanding and generation without a separately trained vision module (Wu et al., 2026). Our focus is narrower: we compare visual pathways under a matched action expert and study which backbone depths the expert should read.

3 Model

3.1 Structurally native visual interface

Following NEO (Diao et al., 2026a), NEO-VLA adopts the structurally native visual interface, mapping raw images directly into the backbone token space through a lightweight two-stage convolutional embedder with an overall stride of 32. Each visual token therefore corresponds to a 32×32 image region. Visual and language tokens are processed jointly throughout the decoder backbone, while continuous actions are produced by a flow-matching expert rather than through action tokenization.

We also employ Native Rotary Position Embedding (Diao et al., 2026a) to factorize positions into temporal, height, and width coordinates with decoupled rotary frequencies. Image tokens retain explicit two-dimensional spatial coordinates, while videos additionally advance along the temporal axis, enabling a unified treatment of images, videos, and interleaved multimodal sequences.

The backbone further employs modality-aware multimodal attention: text follows causal attention, whereas visual tokens attend bidirectionally within each visual block. This allows dense spatial interaction while visual and linguistic representations evolve jointly across layers. As a result, the action expert can access different stages of the visual hierarchy, from fine-grained early representations to more semantic later features, rather than being constrained to a single encoder output.

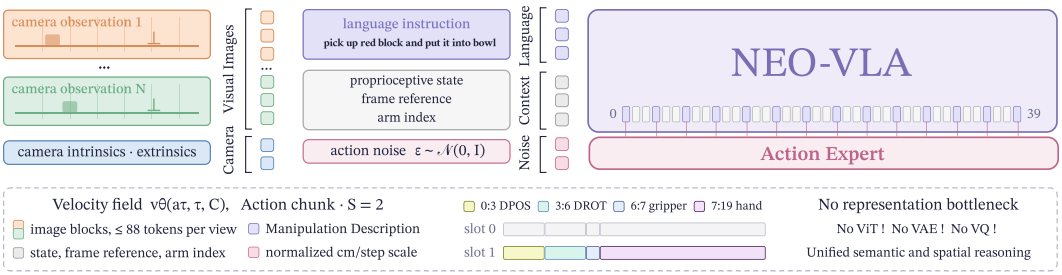


Figure 2: **NEO-VLA: pixels in, continuous actions out.** Camera views enter at their own resolution through a convolutional patch embedder of about 9M parameters (there is no vision transformer, no autoencoder, and no vector-quantised codebook) and become tokens in the same stream as the language instruction. One 2B, 40-layer decoder-only framework processes both, with each attention head's rotary positions split into time, height and width components so that an image keeps its two-dimensional structure. A separate, narrow, randomly initialised action expert reads the backbone's cached keys and values through joint per-layer attention and emits a continuous action chunk by flow matching. Nothing on the path from pixels to actions is quantised, as the dashed panel records.

Notably, we train a 2B, 40-layer decoder-only native vision-language backbone through a three-stage multimodal curriculum. We first establish visual perception with large-scale image-text pretraining while preserving the pretrained linguistic capabilities of the language backbone. Mid-training then strengthens vision-language alignment with diverse captioning, conversation, OCR, and detection data, followed by supervised instruction tuning on high-quality multimodal interactions. This curriculum progressively develops the model from visual perception to multimodal reasoning and instruction following, providing a strong foundation for subsequent action learning.

3.2 Action over the visual hierarchy

In Figure 2, NEO-VLA builds on the native multimodal backbone described above, where visual representations evolve jointly with language throughout the decoder stack. This exposes a hierarchy of visual features to the policy, from early layers that retain fine-grained appearance and geometry to deeper layers that increasingly integrate linguistic and task context. The action expert can read from multiple depths of the backbone rather than relying on a single final visual representation.

The visual pathway naturally supports variable-resolution inputs. Instead of resizing every image to a fixed square, we preserve its aspect ratio and downsample only when required by the token budget. Consequently, visual token grids vary across data sources, including 8×8 for OXE, 8×11 for AgiBot, and 6×10 for RH20T. This preserves the native spatial geometry of observations across heterogeneous robot datasets without imposing a fixed visual layout.

We further unify perception and action under the Native-RoPE setting. Visual tokens are indexed along temporal, height, and width axes with decoupled rotary frequencies, preserving spatial structure and temporal order within the backbone. Action tokens are aligned to the same temporal coordinate system, allowing visual observations, language, and actions to coexist within a unified positional framework. This establishes a direct transition from spatially structured perception to temporally ordered control, while keeping action prediction coupled to the evolving visual hierarchy.

3.3 Action readout across visual depths

The action expert reads from a configurable subset of backbone layers. Let $\mathcal{M} = (m_1, \ldots, m_d)$ denote the layer map, where expert layer j attends to the cached keys and values from backbone layer m_j. This formulation makes the depth of visual readout an explicit architectural choice. We evaluate four representative mappings:

- `first-28`: $\mathcal{M} = (0, \ldots, 27)$, reading only early backbone layers.
- `skip-28`: 28 layers sampled approximately uniformly across all 40 backbone layers.
- `end-28`: $\mathcal{M} = (12, \ldots, 39)$, reading only late backbone layers.
- `full-40`: all 40 backbone layers, paired with a 40-layer action expert.

Table 1: **Pretraining corpus.** We combine four public sources into 569 datasets, 1,440,644 episodes, 338,234,023 frames, and 1,475 camera views. *As held* is each source's share of stored frames, while *As sampled* is its expected share in a training batch. Sampling is deliberately rebalanced across sources, with RH20T and DROID oversampled and AgiBot undersampled relative to corpus size. DROID is reprocessed separately to retain both exterior cameras. We exclude 19 AgiBot dexterous-hand tasks whose 12-DoF end effectors do not match our action space. Camera calibration is available for 1,233 of the 1,475 views, but for none of the 240 Open X-Embodiment views.

		Corpus as built				Share of frames (%)	
Source	Arms	Datasets	Episodes	Frames	Views	As held	As sampled
AgiBot-World	2	198	158,102	282.7M	594	83.57	50.00
Open X-Embodiment	1	120	1,210,137	30.3M	240	8.95	10.00
RH20T	1	210	32,323	14.6M	518	4.30	30.00
DROID	1	41	40,082	10.7M	123	3.18	10.00
Total		569	1,440,644	338.2M	1,475	100.00	100.00

The first three configurations use the same 28-layer expert and identical hidden width, keeping parameter count and computation closely matched. Their differences therefore primarily isolate *where* the expert reads from the backbone rather than how much capacity it has. `full-40` increases expert depth and is thus treated separately as a higher-capacity configuration.

3.4 Continuous action policy modeling

Actions are generated by a lightweight, randomly initialized action expert that runs in parallel with the backbone and attends to its hidden states at each layer, following Black et al. (2024). The expert forms a separate, substantially smaller computation stream rather than mirroring the backbone, which keeps the pretrained backbone parameters decoupled from direct action prediction. We train the expert with flow matching. Given a ground-truth action chunk $\boldsymbol{a}$ and Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, the expert predicts the conditional velocity field as follows:

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{\boldsymbol{a},\epsilon,\tau}\left[\left\|v_\theta(\boldsymbol{a}_\tau, \tau, \mathcal{C}) - (\boldsymbol{a} - \epsilon)\right\|_2^2\right], \tag{1}$$

where $\mathcal{C}$ is the multimodal prefix and the time condition τ is drawn from a Beta distribution concentrated near the noisy end of the path. During the inference process, the chunk is produced by ten Euler steps with the prefix key–value cache held fixed, and without classifier-free guidance.

In Figure 2, each action chunk is a tensor of shape $[S, K, A]$ with $S = 2$ slots, $K = 30$ steps, and $A = 19$ dimensions. Slots 0 and 1 carry the arms. Within a slot, the columns are end effector translation deltas, a log map rotation delta, a gripper aperture, and hand joints. Translations are expressed in centimetres per step and passed through a single global normalizer.

4 Training

In Table 1, we construct a unified corpus from four public datasets: AgiBot-World (Bu et al., 2025), Open X-Embodiment (O'Neill et al., 2024), RH20T (Fang et al., 2024), and DROID (Khazatsky et al., 2024). Together, these sources provide broad coverage across robot embodiments, tasks, and visual conditions, while differing substantially in scale and episode structure.

The training corpus is highly imbalanced across data sources, motivating source-aware sampling during optimization. Although AgiBot accounts for 83.57% of all frames due to its substantially longer episodes, Open X-Embodiment contributes the majority of episodes while representing less than 10% of the total frames. We therefore sample each training batch with target proportions of 50% AgiBot, 30% RH20T, and 10% each from Open X-Embodiment and DROID. Relative to their stored proportions, this corresponds to approximately $7\times$ oversampling for RH20T and $0.6\times$ sampling for AgiBot. The datasets also differ in the visual information they preserve. Open X-Embodiment images are pre-resized to 256×256, so their original aspect ratios cannot be recovered, and camera calibration is unavailable for its 240 views. Other sources preserve aspect ratio and are only downscaled when needed, yielding token grids of 8×11 for AgiBot, 6×10 for RH20T, and 8×8 for Open X-Embodiment. More experimental details are provided in the Appendix.

Table 2: **Backbone comparison at a matched action expert.** Every row uses the same data, recipe and evaluation. FROZEN trains the action expert only; FINETUNED trains the whole stack; ENCODER UNFROZEN trains the vision tower and the expert while the language layers stay frozen. EXERT ACCESS ENCODER makes action expert can directly access intermediate vision-tower features.

Backbone	Backbone Adaptation	LIBERO suites ↑				**Average**
		Spatial	Object	Goal	Long	
Qwen3-VL (Bai et al., 2025), 2B	FROZEN	71.40	89.80	92.80	64.00	79.5
	FINETUNED	91.10	91.60	95.80	78.60	89.2
	ENCODER UNFROZEN	91.80	92.00	94.00	79.00	89.2
	EXPERT ACCESS ENCODER	92.15	93.50	93.80	82.00	**90.3**
Gemma-3 (Team, 2025), 4B	FROZEN	85.00	91.40	90.20	71.20	84.5
	FINETUNED	90.40	96.20	94.60	83.00	**91.0**
Our Native Model, 2B	FROZEN	89.20	95.80	93.20	85.40	90.9
	FINETUNED	95.80	98.40	97.00	86.00	**94.3**

5 EXPERIMENTS

We structure the experiments around three questions. First, does removing the separate vision encoder improve policy performance under matched control settings? Second, which backbone depths does the action expert rely on, and what information does it extract from them? Third, how does the resulting policy perform on standard in-distribution benchmarks and out-of-distribution benchmarks?

5.1 ENCODER-BASED BACKBONES AGAINST A NATIVE-INPUT ONE

We compare NEO-VLA with Qwen3-VL-2B (Bai et al., 2025) and Gemma-3-4B (Team, 2025) as shown in Table 2. All are pretrained with Open X-Embodiment (O'Neill et al., 2024) with the same training budget consistent with vanilla OpenVLA's setting; more details are listed in the appendix.

The native backbone leads with and without finetuning. With a frozen backbone, NEO-VLA achieves an average success rate of 90.90, compared with 79.50 for Qwen3-VL and 84.50 for Gemma-3. Full finetuning raises its score to 94.30, compared with 89.2 and 91.0, respectively. In the finetuned setting, NEO-VLA also performs best on each of the four LIBERO suites. The larger Gemma-3 backbone narrows the gap relative to Qwen3-VL, but does not eliminate it in either setting.

Direct access to intermediate visual features closes the gap. In Table 2, we surprisingly discover that unfreezing Qwen3-VL's vision encoder while keeping its language layers frozen raises its score from 79.5 to 89.2. Allowing the action expert to access intermediate vision-encoder features achieves 90.3, reducing the gap to frozen-backbone NEO-VLA from 11.40 to just 0.54 percentage points. This encoder-access variant also exceeds the fully finetuned Qwen3-VL configuration. These results suggest that where the expert reads visual features matters alongside which backbone parameters are updated. In particular, an encoder-based policy can recover much of the performance gap when its expert is no longer restricted to representations downstream of the encoder.

The native backbone supports strong control without full finetuning. Training only NEO-VLA's action expert achieves 90.9, exceeding fully finetuned Qwen3-VL and nearly matching fully finetuned Gemma-3 at 91.0. Updating the native backbone provides a further 3.40-point improvement, compared with 9.73 for Qwen3-VL and 6.50 for Gemma-3. Thus, the native backbone already provides representations that support strong policy learning without backbone adaptation. This is not a compute-efficiency claim or an isolated test of encoder removal: the backbones differ in language weights, tokenisers, pretraining corpora, input resolutions, and parameter counts. The comparison establishes the effectiveness of the native backbone under the evaluated recipe, while the encoder-access result highlights the importance of the visual interface.

Finding. The native backbone achieves the highest average success rates in both frozen and finetuned settings. Direct access to intermediate encoder features closes the frozen-backbone gap, highlighting the importance of which visual representations the action expert can reach.

Table 3: **Layer-map ablation.** We vary which backbone layers are accessible to the action expert. *Depth* denotes the number of backbone layers retained; thus, `first-28` and `first-14` also truncate the backbone. The two disjoint maps share no attachment points, providing a direct early-versus-late comparison. Each entry is evaluated over 2000 rollouts at step 30,000. Higher is better.

Adaptation	Layer map	Depth	LIBERO suites ↑				**Average**
			Spatial	Object	Goal	Long	
FROZEN	`first-28` [0..27]	28	94.8	98.4	97.6	88.8	**94.9**
	`skip-28` [uniform]	40	89.2	95.8	93.2	85.4	90.9
	`end-28` [12..39]	40	78.4	93.6	94.2	80.4	86.7
FROZEN, disjoint	`first-14` [0..13]	14	95.6	97.2	96.2	85.4	**93.6**
	`last-14` [26..39]	40	92.2	95.2	89.4	81.8	89.6
FINETUNED	`first-28`	28	95.0	96.8	96.6	90.0	94.6
	`skip-28`	40	95.6	97.8	96.0	92.0	**95.3**
	`end-28`	40	94.6	95.2	95.8	84.2	92.5

5.2 WHAT EACH EXPERT LAYER IS LOOKING AT?

Changing which layers the expert can access. The action expert attaches to a configurable set of backbone layers in Table 3, allowing us to vary where it reads from the backbone while keeping the training recipe and expert construction fixed. `skip-28` samples 28 layers approximately uniformly across all 40 backbone layers, whereas `end-28` restricts the expert to layers 12–39. To obtain a cleaner early-versus-late comparison, we additionally use two disjoint maps: `first-14` [0..13] and `last-14` [26..39], which share no attachment points. `first-28` reads only layers 0–27 and truncates the backbone at layer 27; it therefore changes backbone depth and compute in addition to readout position, and we do not treat it as a pure layer-access comparison.

With a frozen backbone, access to early layers is consistently important: `skip-28` outperforms `end-28`, and the disjoint `first-14` versus `last-14` comparison shows the same trend. The configurations that exclude the earliest layers perform worst. Together, these results indicate that the action expert benefits from reading the early backbone representations.

Which attachment points matter. The layer-map ablation shows that access to early backbone layers improves performance, but does not reveal which attachment points the action expert relies on most. We therefore intervene on one attachment point at a time. For the upper panel of Figure 3, we replace the backbone activation with that from a different observation in the same batch and measure the resulting prediction error. We repeat the intervention with matched-magnitude noise to distinguish the loss of scene-specific information from the generic effect of perturbing the representation.

The upper panel reveals a clear depth-dependent pattern. Replacing activations at early attachment points causes a substantial increase in prediction error, whereas perturbing later layers has only a limited effect. In contrast, matched-magnitude noise remains close to the unperturbed baseline, indicating that the degradation is driven by the loss of scene-specific information rather than by perturbation strength alone. Together, these results suggest that early backbone representations carry disproportionately important visual cues for action prediction.

The lower panel asks what information the expert uses at those depths by hiding one input at a time. On the robot-frame arm used in our main experiments, the instruction matters most at the first attachment point, the wrist-camera image is strongest around layers 6–10, and the instruction becomes important again around layer 17. Camera geometry has little effect on this arm. To verify that this null is not caused by a failed intervention, we also plot the same wrist-camera extrinsics from a separate arm whose actions are expressed in the wrist-camera frame. There, the wrist pose produces a large response at the first attachment point and almost none afterwards, as expected when the pose defines the action coordinate frame. Together, the two panels show both *where* the expert relies on the backbone and *what* information it reads at those depths.

Finding. The action expert relies primarily on early backbone representations. Across depth, it first uses the instruction, then the camera image, and finally the instruction again. The wrist-frame control confirms sensitivity to camera geometry when action prediction depends on it.

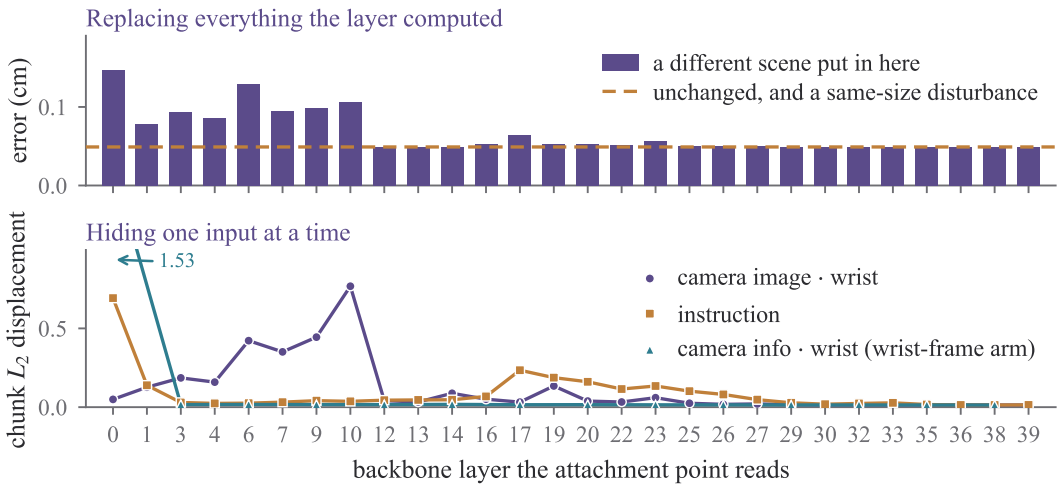


Figure 3: **Two views of the same 28 attachment points.** *Top:* replacing early-layer activations with those from another scene sharply increases prediction error, while replacing later layers has little effect; matched-magnitude noise produces almost no change. *Bottom:* hiding one input at a time shows an instruction–image–instruction pattern across depth. The camera-info curve is a positive control from a separate wrist-frame arm, where wrist pose matters strongly at the first attachment point; on the robot-frame arm used elsewhere, camera geometry has negligible effect.

Table 5: **LIBERO-plus by perturbation axis.** *Open* indicates whether both code and weights are publicly available, checked 2026-09-24; other results follow Yuan et al. (2026). The total is task-weighted rather than the mean of the seven axes; **bold** marks the best value in each column.

Model	Open	Camera	Robot	Language	Light	Background	Noise	Layout	**Total**
π_0	✓	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
StarVLA	✓	52.5	49.8	**88.5**	95.7	95.7	73.0	76.9	74.1
ABot-M0	✓	60.4	67.9	86.4	96.2	91.6	86.4	82.6	80.5
Cosmos-Policy	✓	75.8	63.3	81.7	96.5	88.9	92.7	82.2	82.2
$\pi_{0.5}$	✓	78.4	73.6	80.8	96.2	94.1	89.0	84.5	84.4
Qwen-RobotManip-scratch	×	70.4	44.9	88.1	95.8	95.5	84.4	79.1	78.3
Being-H0.7	×	82.0	59.0	82.8	**97.8**	90.0	93.5	**88.5**	84.8
Qwen-RobotManip	×	87.2	**75.5**	85.6	96.6	**97.7**	**97.7**	87.3	**89.0**
NEO-VLA	✓	**94.1**	73.2	87.3	96.3	92.3	95.3	82.2	88.7

5.3 Standard Benchmark Results

Simulator versions, camera inputs, training data, and evaluation protocols vary across prior work, so the results below provide benchmark context rather than controlled comparisons. We additionally report whether both code and weights are publicly available, since this determines whether a result can be independently reproduced.

LIBERO (Liu et al., 2023). In Table 4, NEO-VLA reaches an average success rate of 98.6. Here EO-1 and MMaDA-VLA obtain similarly high scores, at 98.4 and 98.0, respectively, while Qwen-RobotManip reports a single aggregate score of 99.1 without a per-suite breakdown. The small differences among the strongest methods make LIBERO increasingly difficult to use for distinguishing current policies. We therefore report it mainly to show that NEO-VLA retains strong in-distribution performance rather than to draw conclusions from small cross-paper differences.

Table 4: **LIBERO**, 2,000 episodes per suite at step 27,500. Here *Open* is whether both code and weights are publicly available, checked 2026-09-24; **bold** marks the best value per column.

Model	Open	Spatial	Object	Goal	Long	**Average**
OpenVLA	✓	84.7	88.4	79.2	53.7	76.5
$\pi_{0.5}$	✓	98.8	98.2	98.0	92.4	96.8
MMaDA-VLA	✓	98.8	99.8	98.0	95.2	98.0
EO-1	✓	**99.7**	99.8	99.2	94.8	98.4
Qwen-RobotManip	×	—	—	—	—	**99.1**
NEO-VLA	✓	99.4	**100.0**	**99.3**	**95.6**	98.6

LIBERO-plus (Fei et al., 2025) provides a more informative picture by evaluating seven types of distribution shift separately. As shown in Table 5, NEO-VLA reaches an overall score of 88.7, the highest among the models in the table with both code and weights publicly available. Its performance is particularly strong under camera perturbations, where it reaches 94.1, and under image noise, where it reaches 95.3. The gains are not uniform across all shifts: performance is lower under robot and layout changes, which alter the embodiment or scene configuration rather than only the visual observation. Thus, the main robustness advantage appears under changes to how the scene is observed, rather than under every form of distribution shift. This pattern is consistent with the layer analysis in Section 5.2, where the action expert relies strongly on early visual representations.

RoboTwin 2.0 (Chen et al., 2025). Different from LIBERO-plus, RoboTwin 2.0 allows us to vary the training distribution. We therefore report two settings in Table 6. The *official setting* follows the benchmark budget of 2.5K clean episodes and evaluates clean-to-randomized transfer through *Clean2Rand*. The *data-scaling setting* uses approximately 27K episodes and trains separate models on the clean and randomized splits, allowing us to study the effect of broader training coverage.

Under the official 2.5K setting, the main difference appears under distribution shift. NEO-VLA scores 78.32 on the clean split and retains 68.05 when evaluated under randomization. Among models with public code and weights, this is the strongest *Clean2Rand* result in the table, while several clean-trained policies drop sharply under the same shift. The result suggests that NEO-VLA preserves substantially more of its clean performance when the environment changes.

Table 6: **RoboTwin 2.0.** We report two training settings: the *official setting* with 2.5K episodes and a *data-scaling setting* with approximately 27K episodes. *Clean* and *Rand* evaluate models trained on the corresponding data split, while *Clean2Rand* trains on clean data and evaluates under randomization. *Open* indicates whether both code and weights are public.

Model	Open	Clean	Rand	Clean2Rand
OFFICIAL SETTING: 2.5K EPISODES				
FastWAM	✓	77.80	—	1.90
StarVLA-OFT	✓	46.52	—	3.16
$\pi_{0.5}$	✓	70.70	—	46.00
OLA-Sem	×	75.12	—	67.56
Qwen-RobotManip	×	**89.58**	—	**68.12**
NEO-VLA	✓	78.32	—	68.05
DATA-SCALING SETTING: 27K EPISODES				
StarVLA-OFT	✓	88.18	88.32	—
Qwen-RobotManip	×	**93.10**	**93.50**	—
NEO-VLA	✓	87.88	88.54	—

The data-scaling setting asks a different question: what happens when the training data itself includes randomization? At approximately 27K episodes, NEO-VLA reaches 87.88 on clean data and 88.54 on randomized data, with almost no gap between the two splits. StarVLA-OFT shows a similar pattern, while Qwen-RobotManip achieves the highest scores in this block. Because the two settings use very different training budgets, we compare models within each setting rather than across them.

Together, the two settings show that robustness on RoboTwin depends strongly on the training distribution. A policy trained only on clean data can lose substantial performance under randomization, whereas training directly on randomized data largely removes this gap.

6 CONCLUSION

consistently outperforms encoder-based alternatives, with the largest gains when the backbone is frozen; full finetuning narrows, but does not eliminate, this gap. Layer-wise analyses further reveal that action prediction depends disproportionately on early visual representations: perturbing early-layer activations substantially alters the predicted actions, whereas later-layer interventions have far smaller effects. Input ablations suggest a complementary depth-wise pattern, with the model first grounding the instruction, then relying more strongly on visual observations, and finally returning to higher-level task context. These findings are reflected in downstream evaluation, while also exposing the limits of current benchmarks: LIBERO is already close to saturation, and RoboTwin performance remains sensitive to training scale and data distribution. Overall, our results point to a broader design principle for embodied intelligence: removing the separate vision encoder allows the policy to access a richer hierarchy of visual computation, rather than reasoning only from a fixed perceptual endpoint.

AI Assistance Disclosure

In this work, we used generative AI tools for language editing, including polishing paragraphs to improve clarity and logical flow and correcting grammatical errors. We have not used generative AI tools for idea generation, method design, experimental design, or data analysis, and the remaining required disclosure tasks are not applicable to this work. Additionally, we did not use generative AI tools for any other tasks requiring recommended disclosure. We have reviewed all AI-assisted text to ensure that it accurately reflects the authors' intended meaning and scientific claims. The ideas, methodology, experimental design, results, and analyses presented in this work are entirely the authors' own. We take responsibility for the final content of this work, including any text produced with the aid of generative AI.

Ethics statement

This work studies vision-language-action models for robotic manipulation using existing robotic datasets and simulation benchmarks. We do not collect new human-subject data, conduct user studies, or introduce a new dataset. As our models are trained on existing multimodal and robotic data, they may inherit biases, annotation errors, privacy concerns, or other limitations present in the source datasets. Like other learned robotic policies, NEO-VLA may behave unpredictably under distribution shift or in environments not represented during training. Such failures could lead to unsafe actions when deployed on physical robots. Our experiments are therefore intended for research purposes and are conducted primarily in controlled benchmark settings. Practical deployment would require extra safety mechanisms, validation under diverse real-world conditions, and appropriate human oversight.

Reproducibility statement

We provide detailed descriptions of the model architecture, action expert, training objectives, optimization settings, data mixtures, preprocessing procedures, and evaluation protocols in Appendix **??**. Upon publication, we will release the model implementation, training and evaluation code, configuration files, data-processing pipeline, and trained checkpoints. For third-party datasets, we will provide the exact data configurations, preprocessing procedures, sampling ratios, and instructions for obtaining the original data subject to their respective licenses.

References

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. *arXiv preprint arXiv:2511.21631*, 2025.

Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. Introducing our multimodal models, 2023. URL `https://www.adept.ai/blog/fuyu-8b`.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Junhao Cai, Zetao Cai, Jiafei Cao, Yilun Chen, Zeyu He, Lei Jiang, Hang Li, Hengjie Li, Yang Li, Yufei Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Bohan Hou, Yuming Jiang, Jiayan Guo, Xin Li, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Haiwen Diao, Yufeng Cui, Xiaotong Li, Yueze Wang, Huchuan Lu, and Xinlong Wang. Unveiling encoder-free vision-language models. *Advances in Neural Information Processing Systems*, 37: 52545–52567, 2024.

Haiwen Diao, Xiaotong Li, Yufeng Cui, Yueze Wang, Haoge Deng, Ting Pan, Wenxuan Wang, Huchuan Lu, and Xinlong Wang. Evev2: Improved baselines for encoder-free vision-language models. *arXiv preprint arXiv:2502.06788*, 2025.

Haiwen Diao, Mingxuan Li, Silei Wu, Linjun Dai, Xiaohua Wang, Hanming Deng, Lewei Lu, Dahua Lin, and Ziwei Liu. From pixels to words–towards native vision-language primitives at scale. In *International Conference on Learning Representations*, volume 2026, pp. 109909–109929, 2026a.

Haiwen Diao, Jiahao Wang, Chenjing Ding, Hanming Deng, Jiangnan Chen, Ruixi Zhang, Ruohui Wang, Wenwen Tong, Xiangyu Fan, Yubo Wang, et al. Sensenova-u1. 5: Towards native unified visual intelligence. *arXiv preprint arXiv:2609.11929*, 2026b.

Haiwen Diao, Jiahao Wang, Penghao Wu, Yuhao Dong, Yuwei Niu, Yue Zhu, Zhongang Cai, Weichen Fan, Linjun Dai, Silei Wu, et al. From pixels to words–towards native one-vision models at scale. *arXiv preprint arXiv:2605.28820*, 2026c.

Haiwen Diao, Penghao Wu, Hanming Deng, Jiahao Wang, Shihao Bai, Silei Wu, Weichen Fan, Wenjie Ye, Wenwen Tong, Xiangyu Fan, et al. Sensenova-u1: Unifying multimodal understanding and generation with neo-unify architecture. *arXiv preprint arXiv:2605.12500*, 2026d.

Hao-Shu Fang, Hongjie Fang, Zhenyu Tang, Jirong Liu, Chenxi Wang, Junbo Wang, Haoyi Zhu, and Cewu Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 653–660. IEEE, 2024.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Junlin Han, Shengbang Tong, David Fan, Minghao Chen, Philip Torr, Filippos Kokkinos, and Mike Lewis. Towards physics of multimodal pretraining: Knowledge flow, modality synergy, early unification, and recipes. *arXiv preprint arXiv:2608.05000*, 2026.

Yucheng Hu, Jianke Zhang, Yuanfei Luo, Yanjiang Guo, Xiaoyu Chen, Xinshu Sun, Kun Feng, Qingzhou Lu, Sheng Chen, Yangang Zhang, et al. Bagelvla: Enhancing long-horizon manipulation via interleaved vision-language-action generation. *arXiv preprint arXiv:2602.09849*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Nikita Kachaev, Mikhail Kolosov, Daniil Zelezetsky, Alexey K Kovalev, and Aleksandr I Panov. Don't blind your vla: Aligning visual representations for ood generalization. *arXiv preprint arXiv:2510.25616*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Thinking Machines Lab. Interaction models: A scalable approach to human-ai collaboration. *Thinking Machines Lab: Connectionism*, May 2026. doi: 10.64434/tml.20260511. https://thinkingmachines.ai/blog/interaction-models/.

Haotian Liang, Xinyi Chen, Bin Wang, Mingkang Chen, Yitian Liu, Yuhao Zhang, Zanxin Chen, Tianshuo Yang, Yilun Chen, Jiangmiao Pang, et al. Mm-act: Learn from multimodal parallel generation to act. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35080–35090, 2026.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Yang Liu, Pengxiang Ding, Tengyue Jiang, Xudong Wang, Wenxuan Song, Minghui Lin, Han Zhao, Hongyin Zhang, Zifeng Zhuang, Wei Zhao, et al. Mmada-vla: Large diffusion vision-language-action model with unified multi-modal instruction and generation. *arXiv preprint arXiv:2603.25406*, 2026.

Zhuoyang Liu, Jiaming Liu, Jiadong Xu, Nuowei Han, Chenyang Gu, Hao Chen, Kaichen Zhou, Renrui Zhang, Kai Chin Hsieh, Kun Wu, et al. Mla: A multisensory language-action model for multimodal understanding and forecasting in robotic manipulation. *arXiv preprint arXiv:2509.26642*, 2025.

Gen Luo, Xue Yang, Wenhan Dou, Zhaokai Wang, Jiawen Liu, Jifeng Dai, Yu Qiao, and Xizhou Zhu. Mono-internvl: Pushing the boundaries of monolithic multimodal large language models with endogenous visual pre-training. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24960–24971. IEEE, 2025.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025.

Haoxiang Ma, Junhao Cai, Xiaoxu Xu, Hao Li, Yuyin Yang, Yang Tian, Jiafei Cao, Hongrui Zhu, Zherui Qiu, Yuqiang Yang, et al. Internvla-a1. 5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

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.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Dong Wang, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, et al. Eo-1: An open unified embodied foundation model for general robot control. *arXiv preprint arXiv:2508.21112*, 2025.

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

Gemma Team. Gemma 3. 2025. URL `https://goo.gle/Gemma3Report`.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Shengbang Tong, David Fan, John Nguyen, Ellis Brown, Gaoyue Zhou, Shengyi Qian, Boyang Zheng, Théophane Vallaeys, Junlin Han, Rob Fergus, et al. Beyond language modeling: An exploration of multimodal pretraining. *arXiv preprint arXiv:2603.03276*, 2026.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *International Conference on Learning Representations*, volume 2026, pp. 80929–80944, 2026.

Penghao Wu, Haiwen Diao, Weichen Fan, Lewei Lu, Dahua Lin, and Ziwei Liu. Uncovering understanding-generation synergy in native unified multimodal models: From representation, task to system. *arXiv preprint arXiv:2609.01607*, 2026.

Haozhe Xie, Beichen Wen, Jiarui Zheng, Zhaoxi Chen, Fangzhou Hong, Haiwen Diao, and Ziwei Liu. Dynamicvla: A vision-language-action model for dynamic object manipulation. *arXiv preprint arXiv:2601.22153*, 2026.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

Fengnian Zhang, Tao Huang, Siyu Xu, Zhong Jin, and Chang Xu. Revisiting parameter redundancy in vision-language-action models: Insights from vlm-to-vla adaptation. *arXiv preprint arXiv:2606.31382*, 2026a.

Jianke Zhang, Xiaoyu Chen, Qiuyue Wang, Mingsheng Li, Yanjiang Guo, Yucheng Hu, Jiajun Zhang, Shuai Bai, Junyang Lin, and Jianyu Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. *arXiv preprint arXiv:2601.03309*, 2026b.

Jianke Zhang, Yuanfei Luo, Yucheng Hu, Xiaoyu Chen, Yanjiang Guo, Ziyang Liu, Hongbin Xu, Tian Lan, and Jianyu Chen. Uam: A dual-stream perspective on forgetting in vla training. *arXiv preprint arXiv:2605.15735*, 2026c.

A Implementation Details

Table 7 summarises the evaluation protocol, action representation, image processing, and corpus sampling settings.

Simulator settings. All controlled LIBERO comparisons use the same simulator versions: `mujoco` 3.2.3 and `robosuite` 1.4.1. In our tests, changing the `mujoco` version changed the four-suite average by 12.4 percentage points. We therefore keep these versions fixed and treat results from other papers as reference numbers rather than matched comparisons. For RoboTwin, we use the benchmark's SAPIEN configuration without modification, including its task-specific step limits.

Evaluation budgets. LIBERO ablations use 500 episodes per suite, while the benchmark results use 2,000 episodes per suite to reduce evaluation noise. Unless stated otherwise, rollouts use a replan interval of 12, a horizon of 20, and 10 flow-matching steps. The encoder-unfrozen Qwen3-VL experiment is an exception: it uses a replan interval of 4 and a horizon of 8, as noted in Table 2.

Table 7: **Implementation settings.** Evaluation, action representation, image processing, and corpus sampling settings used in our experiments. Rollout values are defaults; exceptions are stated with the corresponding results. The per-suite episode counts apply to LIBERO.

Setting	Value
EVALUATION PROTOCOL	
Replan interval	12
Rollout horizon	20
Flow-matching steps (NFE)	10
LIBERO simulator	`mujoco` 3.2.3, `robosuite` 1.4.1 (pinned)
RoboTwin simulator	SAPIEN, benchmark-provided per-task configurations
RoboTwin step limits	nine distinct task-specific limits, unchanged
Episodes, controlled runs	500 per suite (§5.1, §5.2)
Episodes, benchmark results	2,000 per suite (§5.3)
RoboTwin evaluation	50 tasks × 100 episodes, seed 0
RoboTwin rendering	ray traced, `action_type: ee`
ACTION CHUNK	
Chunk shape	$[S, K, A] = [2, 30, 19]$
Slots	0 and 1, one per arm
Columns within a slot	translation deltas, log-map rotation delta, gripper aperture, hand joints
Translation units	centimetres per step
Normaliser	one global asinh; actions expressed per step
VISUAL PATH	
Resize rule	downscale only; each side snapped to a multiple of 32 px
Token grid, Open X-Embodiment	8×8
Token grid, AgiBot-World	8×11
Token grid, RH20T	6×10
Tokens per image	60 to 88
Position encoding	rotary, split into time, height, and width
CORPUS SAMPLING	
Batch share rule	weight × frame count, normalised
AgiBot-World	50% expected share
RH20T	30%
Open X-Embodiment	10%
DROID	10%
Relative to stored frame share	RH20T: $\sim 7\times$; AgiBot-World: $\sim 0.6\times$

A.1 MODEL CONFIGURATION

Table 8 lists the backbone and action-expert settings. In the backbone comparison, NEO-VLA and Qwen3-VL use 28-layer action experts, each with 443.6M parameters. Gemma-3 uses a 34-layer expert with 538.0M parameters so that the expert can read all 34 backbone layers. The models therefore share the same expert construction, but not the same expert capacity; the Gemma-3 configuration has the larger expert.

The `skip-28` configuration has 2,777.4M parameters in total. Its measured latency is 244.6 ms for an eight-step chunk. The `full-40` configuration reads every backbone layer and uses a 40-layer expert, increasing both parameter count and computation.

A.2 INPUT TEMPLATE

Table 9 shows input template. All inputs live in a single token stream: image patches, camera identity and geometry, language, robot state, and the action chunk being denoised. They differ in how they are embedded, not in where they enter the model. Nothing bypasses the backbone before reaching the action expert.

Table 8: **Model configuration.** Total parameter count and latency are reported for `skip-28`. The 34-layer expert used with Gemma-3 in Table 2 has more parameters than the 28-layer expert.

Component	Configuration
BACKBONE	
Type	decoder-only multimodal, 40 layers, 2B
Separate vision encoder	none; no ViT, CLIP/SigLIP/DINO, or VAE
Image input	convolutional patch embedder, $\sim$9M parameters
Patch size	16, downsample ratio 0.5
Pixel budget	65,536 to 16,777,216 per image
Canonical resolution	none; native resolution, downscale only
Position encoding	per-head QK split $[t{:}64 \mid h{:}32 \mid w{:}32]$
Rotary base	5×10^6 for time, 1×10^4 for height and width
ACTION EXPERT	
Attachment	joint per-layer attention over cached keys and values
Layers, `skip-28`	28, trainable parameters: 443.6M
Layers, 34-layer variant	34, width 1024, trainable parameters: 538.0M
Initialisation	random; no backbone weights are copied
Objective	flow matching, conditional velocity prediction
Inference	10 Euler steps, fixed prefix KV cache, no CFG
Time distribution	Beta, concentrated near the noisy end
COMPLETE MODEL	
Total parameters	2,777.4M
Latency, 8-step chunk	244.6 ms

Table 9: **The input template**.

Block	Rendered as	Tokens
PREFIX, ONCE PER CAMERA VIEW		
Image patches	<img> [IMG] </img>	60–88
Camera identity	<cam$_k$>	
Intrinsics	one column each: f_x, f_y, c_x, c_y, w, h	6
Extrinsics	one column per pose component	6
PREFIX, LANGUAGE, ONCE PER SAMPLE		
View naming	"Image N is the ⟨name⟩ view.", per view	
Camera geometry	ordinal trust word then digits: *pose exact*, *close*, *rough*, or *unknown* with no digits	none in robot frame
Action frame	"Actions are expressed in the frame of ⟨name⟩ (image N)."	camera frame only
Task	"A video recorded from a robot's point of view executing the following instruction: ⟨task⟩."	
Control rate	"Control frequency: ⟨rate⟩ Hz."	
	text budget	≤ 256
SUFFIX, ONCE PER SLOT, THREE SLOTS		
Frame reference	<frame_ref>	1
Arm identifier	<arm$_i$>; the value is the block's index	1
State	[state]	1
Action chunk	[Actions], $K = 30$ steps	30
	33 per slot, three slots	99

The action frame is specified once in language by naming the visible camera view whose frame defines the actions. Changing that reference is the only difference between the robot-frame and wrist-frame policies used in Section B; the architecture and the rest of the input format stay unchanged.

Outputs are organised by slot rather than as one flat action sequence. Each slot contains its own frame reference, arm identifier, state, and 30-step action chunk. Arm identity is positional: the slot index determines which arm it represents. One slot is reserved for a mobile base, which is unused by the embodiments in this paper but kept to preserve a consistent slot ordering across datasets.

Camera pose can be provided either through dedicated structured columns or written into the instruction text. Section C compares these two forms; the structured version shown here is the one used by the masking analysis.

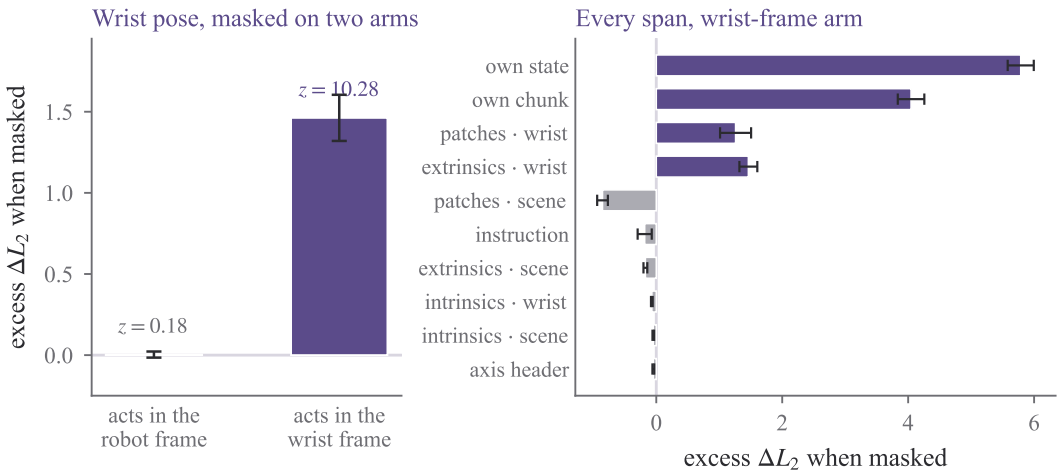


Figure 4: **Validating the masking analysis with robot-frame and wrist-frame policies.** *Left:* the same wrist-pose intervention is applied to two policies. Masking the pose has little effect when actions are expressed in the robot frame, but a large effect when actions are expressed in the wrist-camera frame, where the pose defines the output coordinate system. *Right:* we mask each input span of the wrist-frame policy separately. Values are excess chunk L_2 displacement over a matched-random control. The strong response to wrist-camera extrinsics confirms that the masking procedure detects camera geometry when the policy depends on it.

A.3 Pre-training

Table 10 gives the stage-1 pre-training settings. The training sources differ substantially in size and episode length. AgiBot-World contains 83.57% of the stored frames, with roughly 1,800 frames per episode. Open X-Embodiment contains about four fifths of the episodes but less than a tenth of the frames. We therefore reweight the sources rather than sample uniformly from all stored frames. The expected batch shares are listed in Table 7.

The image data also have two limitations. Open X-Embodiment images have already been resized to a fixed square, so we cannot recover their original aspect ratios. None of its 240 camera views include calibration, compared with 1,233 of the remaining 1,235 views. These limitations restrict the aspect-ratio and camera information available during pre-training.

Table 10: **Pre-training protocol.** Corpus composition is given in Table 1, and sampling shares are listed in Table 7. Fine-tuning checkpoints are specified separately for each experiment: step 27,500 for Table 4 and step 30,000 for Table 3.

Setting	Stage-1 pre-training
Steps	75,000
Hardware	64 H100 GPUs
Time per step	$\sim$3.4 s, sustained
Training time	$\sim$71 h, plus $\sim$11.5 min per checkpoint save
Corpus	569 datasets, 1,440,644 episodes, 338.2M frames, 1,475 views
Excluded tasks	19 AgiBot dexterous-hand tasks with 12-DoF end effectors
Precision	fp32 trainable parameters with bf16 autocast

B Validating the Masking Analysis

In Section 5.2, camera geometry has almost no effect on the robot-frame policy. This could mean that the policy truly does not need the wrist-camera pose, but a null result could also come from a faulty masking procedure. We therefore repeat the same intervention on a second policy whose actions are expressed in the wrist-camera frame. For this policy, the wrist pose defines the output coordinate system and should therefore matter.

Robot-frame versus wrist-frame policy. The left panel of Figure 4 applies the same wrist-pose intervention to both policies. For the robot-frame policy used in our main experiments, masking the wrist pose has almost no effect on the predicted action. For the wrist-frame policy, the same intervention produces a large change, because the wrist pose directly defines the coordinate frame of the action. This provides a positive control: the masking procedure detects camera geometry when the policy actually depends on it.

Masking individual input spans. The right panel tests the wrist-frame policy in more detail. We divide the prefix into named spans, including image patches, instruction tokens, camera intrinsics and extrinsics, and the policy's own state and action history. For each span, we replace its values with the corresponding values from a different observation in the same batch and measure the resulting change in the predicted action chunk.

To account for the fact that any replacement can disturb the prediction, we also replace the same number of randomly selected prefix columns. We report the target-span displacement minus this matched-random baseline, which we call *excess displacement*. The large response to wrist-camera extrinsics in the right panel is therefore not simply caused by perturbing the activation; it reflects information that the wrist-frame policy actually uses.

The depth-resolved analysis in Figure 5 uses the same two policies, but reports raw chunk L_2 displacement because a matched-random baseline is not available at every attachment point.

Checking that the wrist-pose mask works. To verify that the near-zero response in the robot-frame policy is not caused by a faulty intervention, we repeat the same experiment on a second policy trained with all actions expressed in the wrist-camera frame, so the wrist pose directly defines the coordinate system of the output. All other settings are kept the same, and we mask the same six wrist-pose columns in both policies.

As shown in Figure 4, masking the wrist pose has almost no effect on the robot-frame policy, but causes a large change in the wrist-frame policy. The excess displacement is 0.0034 ± 0.0188 for the robot-frame policy and 1.4622 ± 0.1422 for the wrist-frame policy, with corresponding z values of 0.18 and 10.28. The masking procedure therefore detects the wrist pose when the action representation depends on it. This makes the near-zero response of the robot-frame policy easier to interpret: the pose has little effect there because the policy does not need it to define the action frame, rather than because the masking procedure failed.

Where the wrist pose is used. Figure 5 repeats the masking experiment layer by layer for both policies. Here we report the raw change in the predicted action chunk after masking an input, measured by chunk L_2 displacement.

The camera image behaves similarly in the two policies: masking it has the largest effect around backbone layers 6–9. The wrist pose shows a different pattern. For the robot-frame policy, masking the wrist pose has almost no effect at any depth. For the wrist-frame policy, however, masking the wrist pose produces a large change at the first attachment point (1.53) and almost no change afterwards. This shows that when the action representation depends on the wrist pose, the expert uses that information immediately at its first attachment point.

Span-level and depth-wise masking. Figure 4 measures whether a given input span has a larger effect than a same-sized random replacement. We therefore subtract the displacement caused by replacing the same number of randomly selected prefix columns and report the resulting *excess displacement*.

Figure 5 measures where that input matters across backbone depth. Because no matched-random value is available at every attachment point, it reports the raw chunk L_2 displacement instead. The two figures therefore report different quantities and are used for different purposes.

C Camera Pose: Structured or Prompted

We compare two ways of providing camera extrinsics. Both model is purly trained with Libero data without pretrain under Libero's official setting with the same training budget. ***CaPE*** assigns each

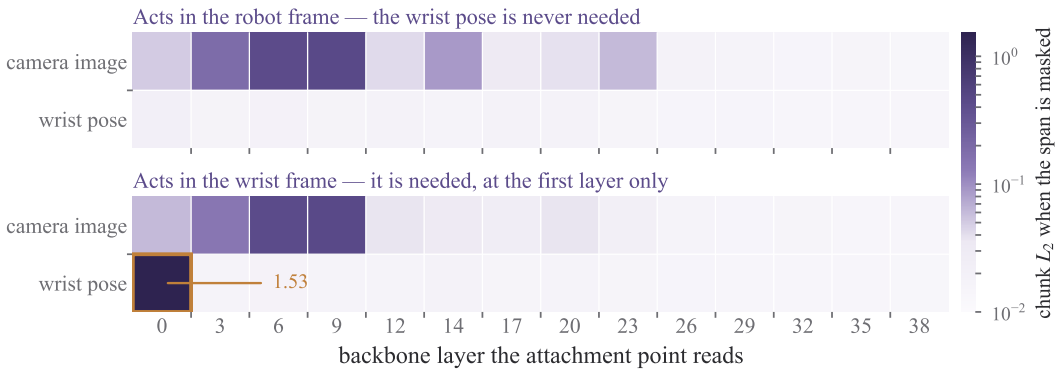


Figure 5: **Camera-image and wrist-pose dependence across depth.** Each cell replaces one input span at one attachment point and measures the resulting change in the predicted action chunk. Wrist-pose replacement has little effect in the robot-frame model, but a strong effect at the first attachment point in the wrist-frame model. Both models respond most strongly to the camera image around backbone layers 6–9. Colours use a logarithmic scale and show raw chunk L_2 displacement, not centimetres or excess over a random control. We show the 14 even-indexed attachment points recorded for both models.

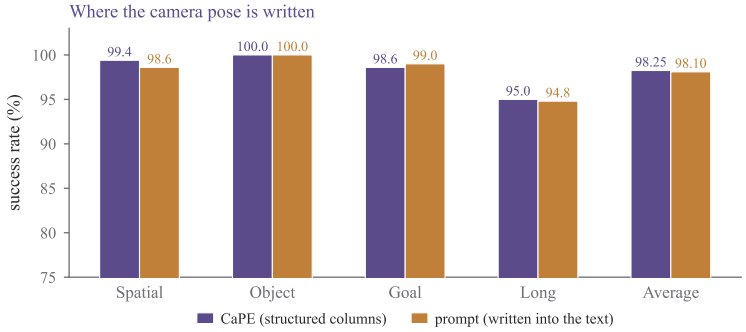


Figure 6: **Structured and prompted camera pose on LIBERO.** CaPE stores each camera's extrinsics in six dedicated prefix columns; the prompt variant writes the same values into the instruction. The two formats achieve similar success rates, averaging 98.25 and 98.10, respectively. Differences across individual suites are at most 0.8 percentage points. We choose the prompt format for its input flexibility rather than an accuracy gain. The vertical axis starts at 75%.

view six dedicated prefix columns containing its pose values. This is the structured format used in the masking analysis in Section B. The prompt variant writes the same six values into the instruction as ordinary text tokens, without reserving separate pose columns.

Figure 6 shows similar LIBERO performance for the two formats. CaPE reaches a four-suite average of 98.25, compared with 98.10 for the prompt variant. No suite differs by more than 0.8 percentage points: the prompt variant scores higher on Goal, lower on Spatial and Long, and the same on Object. Neither format performs better across all four suites.

We use the prompt format for its flexibility rather than for an accuracy gain. A planner or another component that generates the instruction can also include the camera pose in that text, without requiring a separate pose-input interface. The LIBERO comparison supports this choice as a practical alternative to structured columns, but does not establish an advantage on other benchmarks or unseen camera poses.

D Input Ablation on Real-Robot Data

The masking analysis in Section B studies where different inputs matter inside the model, one attachment point at a time. Here, we ask a simpler question: how does the predicted action respond

Table 11: **Input ablation on real-robot data.** Offline evaluation on logged observations from a PiPER/Pika rig ($n = 6$). We report the displacement statistic over an eight-step action chunk together with the repeat-noise floor for each condition. The instruction edits serve as diagnostic controls.

Condition	Displacement (mm)	Noise floor (mm)
Unaltered observation	6.52	0.577
IMAGE		
Blanked image	7.15	1.07
Mirrored image	4.08	0.64
INSTRUCTION		
Object noun substituted	0.24	0.84
Verb substituted	4.14	0.97

when the image or instruction itself is changed? We evaluate this offline using observations recorded on a real PiPER/Pika rig.

Setup. For each observation, the policy predicts an eight-step action chunk without executing it on the robot. We then modify either the image or the instruction and record the resulting displacement statistic over the predicted chunk. Each condition also has a repeat-noise floor, obtained by running the same input repeatedly. This gives a reference for the variation introduced by sampling alone.

Image edits. Changing the image produces a clear response in the predicted action. Blanking the image gives a displacement of 7.15 mm against a 1.07 mm noise floor, while mirroring it gives 4.08 mm against a 0.64 mm floor. These measurements show that the image affects the policy output, although the logged statistic does not retain signed, per-axis action changes. We therefore use it only as a coarse measure of sensitivity rather than to infer how a particular action component changes.

Instruction edits. The two language edits produce very different responses. Replacing the object noun gives 0.24 mm, below its 0.84 mm noise floor, whereas replacing the verb gives 4.14 mm against a 0.97 mm floor. In this small set of examples, the predicted action is therefore much more sensitive to the verb than to the object noun. The purpose of these rows is mainly to check that the measurement can distinguish between input changes that have little effect and those that substantially alter the prediction.

Results. Overall, the offline ablation shows that the policy is sensitive to both visual input and parts of the instruction on real-robot observations. Image edits change the predicted action well beyond the sampling floor, while the language response depends strongly on what is changed: replacing the verb has a clear effect, whereas replacing the object noun does not exceed the noise floor. These results provide a coarse but direct verification that the policy conditions on the current observation and instruction, rather than producing nearly identical actions regardless of the input.

E LAYER-WISE DECODABILITY OF DEPTH AND CORRESPONDENCE

The layer-map experiments show that giving the action expert access to early backbone layers improves policy performance. We next ask what spatial information is available at different depths. To do this, we probe depth and dense correspondence from every layer of the native backbone and the two encoder-based backbones used in our comparison.

Protocol. Each backbone is frozen and evaluated on a held-out image set. At every layer, we extract the hidden states associated with image tokens, reshape them to their spatial token grid, and train a separate linear probe. One probe predicts depth, while the other predicts dense correspondence to a second view. Only the probe is trained; the backbone remains fixed, and no robot actions or policy training are involved.

We deliberately use a linear probe rather than a larger decoder. The goal is to measure how directly the spatial information is available at each layer. A high score means that the information can be recovered with a simple linear mapping.

Table 12: **Layer-wise linear decodability of spatial information.** We train linear probes on frozen backbone representations. *Peak* reports the best score across layers, with its relative depth in parentheses. *Read-layer mean* averages the scores over the layers available to the action expert. Higher is better.

	Depth (δ_1, %) ↑		Correspondence (PCK@0.1, %) ↑	
Backbone	Peak (rel. depth)	Read-layer mean	Peak (rel. depth)	Read-layer mean
ENCODER-BASED				
Qwen3-VL-2B	78.2 (0.22)	54.6	61.5 (0.28)	38.2
Gemma-3-4B	80.1 (0.25)	57.9	63.7 (0.30)	41.0
NATIVE-INPUT				
our model (ours)	79.4 (0.30)	**73.8**	62.8 (0.32)	**58.1**

Depth is evaluated with δ_1, the percentage of pixels whose predicted depth lies within a factor of 1.25 of the ground truth. Dense correspondence is evaluated with PCK@0.1, the percentage of matches within 0.1 of the image size. Higher is better for both metrics. Since the backbones have different numbers of layers, we report layer locations using relative depth in $[0, 1]$.

Peak versus accessible representations. Table 12 reports two quantities. *Peak* is the best probe score obtained at any layer, with the relative depth of that layer shown in parentheses. *Read-layer mean* averages the probe scores over the layers exposed to the action expert in the corresponding backbone configuration.

This distinction separates what a backbone represents somewhere in its computation from what the action expert can directly access.

Results. All three backbones reach similar peak performance, and their best layers appear relatively early, between 0.22 and 0.32 of the backbone depth. The difference becomes much larger at the layers available to the action expert. For depth, the read-layer mean is 73.8 for our model, compared with 54.6 for Qwen3-VL and 57.9 for Gemma-3. For correspondence, the corresponding values are 58.1, 38.2, and 41.0.

Interestingly, our model does not have the highest peak score on either task. Its advantage is instead that much more of this spatial information remains readily decodable at the layers the action expert can read. This matches the layer-map results: the important difference is not simply whether spatial information appears somewhere in the backbone, but whether it remains accessible to the policy at the points where action prediction reads from it.