SCALING VISION-LANGUAGE-ACTION MODELS WITH VIDEO GENERATIVE PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling vision-language-action (VLA) models requires substantial robot data and computation. Pretrained video generators provide visual and temporal representations, but rendering future frames adds iterative sampling and re-encoding. We introduce VID-POLICY, a video-generation-model-enhanced policy that transfers history-conditioned representations from a frozen video generator into a pretrained VLA. A projector maps these features to policy-side keys and values; gated additive attention combines them with native context while preserving the action generator. We assess source precision, history content, and closed-loop control in two policy families. In a fixed-seed, 24-task, five-episode-per-task DIAL development screen, BF16 video features yield 80.8% success, 4.2 percentage points above our reproduced Native DIAL reference. On the full 24-task, 50-episode-per-task RoboCasa-GR1-Tabletop benchmark, WALA-based VID-POLICY reaches 72.33% success, improving over our WALA re-implementation by about 1.83 percentage points. The results show that video representations can guide action without sampling future frames.

1 INTRODUCTION

Scaling model capacity and pretraining data has driven progress in language modeling (Kaplan et al., 2020; Hoffmann et al., 2022). Vision-language-action (VLA) models extend this trajectory to embodied control: larger pretrained models and more diverse robot demonstrations can broaden the behaviors available to a policy (Brohan et al., 2023; O'Neill et al., 2024; Kim et al., 2024). The appeal of scaling is clear, but its cost is particularly demanding in robotics. Robot demonstrations couple visual observations to embodiment-specific actions, and collecting them requires interaction with physical or simulated environments. Increasing policy capacity also increases the computation needed to adapt that capacity to control. These requirements motivate a complementary way to expand a VLA's resources: reuse knowledge that has already been learned outside robot action training.

Video generation provides a natural source of such knowledge. To model how a scene evolves, a video generator learns representations of appearance, motion, and interactions across frames (Ho et al., 2022; Agarwal et al., 2025; Wan et al., 2025). These are also relevant to manipulation, where an action changes object locations and the subsequent visual state. Video can be collected without robot action labels, giving video pretraining access to sequences beyond those available in demonstration datasets. A pretrained video generator therefore offers a source of visual and temporal representations that an existing policy need not learn entirely from robot data. Our aim is to make these resources available for action selection while retaining the policy's learned action-generation architecture. In this work, *scaling* refers to expanding the pretrained representation resources accessible to the VLA through this transfer.

Realizing that transfer requires deciding how the policy should access the video model. Explicit video planning samples future frames before selecting an action (Du et al., 2023; Zhou et al., 2024). This makes the predicted scene visible, but introduces iterative generation and a subsequent mapping from images to actions. VPP takes a feature-based route, aggregating video-predictor representations and training a diffusion action head (Hu et al., 2024). For an already pretrained VLA, the useful resource is the video model's internal representation; rendering it into pixels need not be an interme-

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.

diate step. An interface to those states can preserve the receiver's existing route from observations and language to actions.

We introduce VID-POLICY, a video-generation-model-enhanced policy that transfers history-conditioned states from a frozen video generation network into a pretrained VLA. The video network supplies features from observed frames. A compact projector maps these features to policy-side keys and values, and a gated attention branch retrieves them alongside the native context. Training updates the interface and optional low-rank policy adapters while keeping the source network frozen. The resulting policy uses video-derived context without sampling future frames or replacing its action generator.

We study this interface in DIAL and WALA through source-precision tests, adaptation comparisons, history-content controls, and closed-loop evaluation. Our contributions are:

- To scale pretrained VLA representation resources with knowledge learned from video, we introduce VID-POLICY, a policy interface that transfers frozen video generative priors while retaining the original action generator.
- To integrate the video prior while preserving native attention normalization, we propose gated additive attention aggregation: two attention branches retrieve native and video-derived context separately, then combine their outputs with a learned gate. A lightweight projector, optional low-rank adaptation (LoRA), and source quantization control adaptation cost.
- To assess transfer across policy families, we conduct experiments with two receiving models, covering visual quality, adaptation, history-content dependence, and closed-loop success.

2 RELATED WORK

Video-to-action and world-action models. UniPi and RoboDreamer use generated video for planning or inverse dynamics (Du et al., 2023; Zhou et al., 2024); LAPA learns latent actions from video (Ye et al., 2025). WALA, SG-WAM, and VLA-JEPA supervise action-relevant latent futures (Liu et al., 2026; Zhao et al., 2026; Sun et al., 2026). Other world-action models investigate joint video/action modeling, future-state inference, feedback, and dynamics adaptation (Ye et al., 2026; Zhou et al., 2026; Lei et al., 2026; He et al., 2026; Lyu et al., 2025; Yang et al., 2026a;b; Jing et al., 2026). VPP aggregates video-predictor features with a Video-Former and trains a diffusion action head (Hu et al., 2024). Our interface instead injects projected K/V into an existing policy while retaining its action generator.

Generalist VLA policies. RT-1, RT-2, and PaLM-E connect multimodal representations to control (Brohan et al., 2022; 2023; Driess et al., 2023); Open X-Embodiment, OpenVLA, and Octo broaden policy training across embodiments (O'Neill et al., 2024; Kim et al., 2024; Mees et al., 2024). Recent flow, diffusion, and token-based policies improve generality and action efficiency (Black et al., 2024; Physical Intelligence et al., 2025; Shukor et al., 2025; Bjorck et al., 2025; Pertsch et al., 2025; Liu et al., 2025; Chi et al., 2025; Zhao et al., 2023; Lee et al., 2024; Haldar et al., 2024). We augment these pretrained policy resources with video representations through a trainable interface.

Video and visual representations. Latent world models learn dynamics for control (Ha & Schmidhuber, 2018; Hafner et al., 2019b;a; 2025; Hansen et al., 2024; Micheli et al., 2022), while video foundation models scale temporal generation (Yan et al., 2021; Ho et al., 2022; Kondratyuk et al., 2023; Blattmann et al., 2023; Bar-Tal et al., 2024; Kong et al., 2024; Agarwal et al., 2025; Wan et al., 2025; Hu et al., 2023). Self-supervised image/video encoders and robotic visual representations offer complementary frozen-feature sources (Oquab et al., 2023; Assran et al., 2023; Bardes et al., 2024; Nair et al., 2022; Xiao et al., 2022; Ma et al., 2022; Karamcheti et al., 2023).

Adaptation and evaluation. Low-rank adaptation (LoRA) and quantization reduce adaptation and storage costs (Hu et al., 2021; Dettmers et al., 2023; Xiao et al., 2023; Dettmers et al., 2022); gated cross-attention and efficient attention designs inform our interface (Alayrac et al., 2022; Dao et al., 2022; Ainslie et al., 2023). VLA studies further explore low-resource adaptation and modular

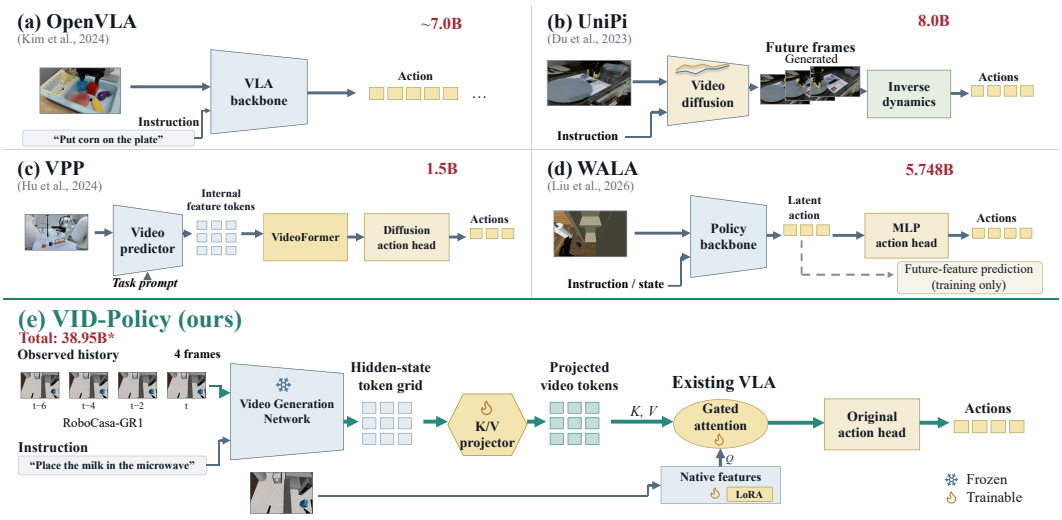

Figure 1: Comparison of policy architectures and component scales. OpenVLA predicts action tokens, UniPi renders future frames, VPP aggregates video-predictor features, and WALA learns action-relevant latent futures. VID-POLICY transfers frozen video-model states into an existing policy. The illustrated 81.40M budget corresponds to the WALA adaptation configuration. Each panel identifies the components counted in its illustrated scale.

skill merging (Ferchau et al., 2026; Omaisan & Mohamed, 2025; Fu et al., 2026). We evaluate closed-loop success on RoboCasa-GR1-Tabletop (Nasiriany et al., 2024), alongside cost and fitting diagnostics, since offline action error and task success capture different properties (Huang et al., 2026).

Figure 1 contrasts the route from visual history to action across these models. Its component sizes describe pretrained resources; the 81.40M trainable budget of the illustrated WALA configuration comprises 66.06M low-rank adapter parameters in the vision-language model (VLM) backbone, denoted VLM-LoRA, and 15.34M projector/gate parameters. Supplementary Section F gives the counting scopes and implementation costs.

3 METHOD

Our approach scales the representation resources available to a pretrained VLA through a frozen video generator. We train the interface between the models while retaining the receiver's action generator. The interface projects observed-history hidden states into policy-side keys and values. Gated additive attention then combines the native context with the retrieved video prior. This transfers the source representation directly into action selection, bypassing the future-frame sampling and re-encoding used in explicit video planning (Du et al., 2023; Zhou et al., 2024).

Figure 2 follows the transfer from frozen video representations (Section 3.1) through K/V projection (Section 3.2) to gated attention inside the receiving VLM (Section 3.3). Section 3.4 specifies the trainable parameters and objective; Section 4 identifies the receiving policies.

3.1 VIDEO GENERATIVE PRIOR EXTRACTION

Let B denote batch size, h, w image height and width, d_s state dimension, T action horizon, and d_a action dimension. The current image $\mathbf{I}_t \in \mathbb{R}^{B\times 3\times h\times w}$, robot state $\mathbf{s}_t \in \mathbb{R}^{B\times d_s}$, and text instruction ℓ determine the action chunk $\widehat{\mathbf{A}}_t \in \mathbb{R}^{B\times T\times d_a}$:

$$\widehat{\mathbf{A}}_t = \mathcal{P}(\mathbf{I}_t, \mathbf{s}_t, \ell). \tag{1}$$

Here $\mathcal{P}(\cdot)$ denotes the pretrained receiving policy. The interface augments attention in its representation backbone, upstream of action generation; it therefore accommodates both diffusion and

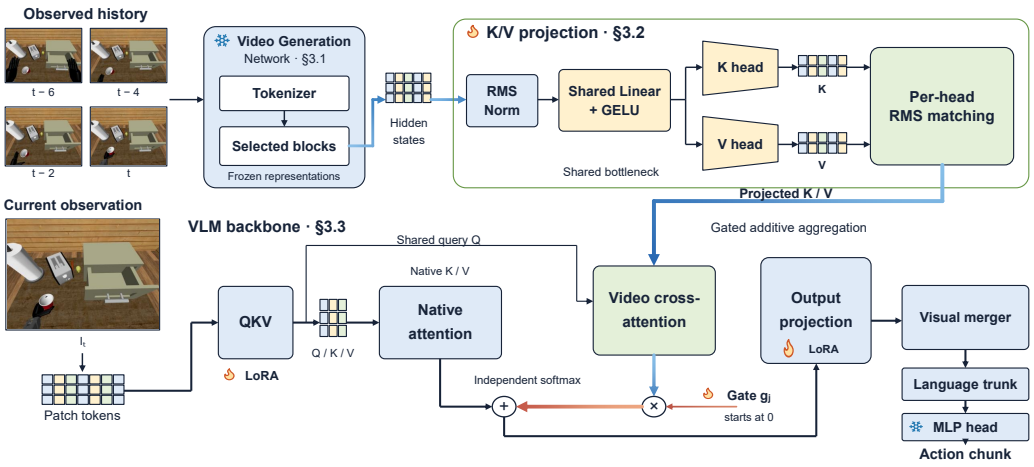

Figure 2: VID-POLICY: frozen video-prior extraction (Section 3.1), K/V projection and magnitude matching (Section 3.2), and gated additive attention aggregation (Section 3.3). The expanded path illustrates one placement in the VLM backbone; the same gated transfer principle applies to other receiving attention blocks. The original action head is retained. Flames and snowflakes mark trainable and frozen modules, respectively.

non-diffusion action heads when that conditioning path is available. We augment it with a frozen video representation while retaining the action parameterization:

$$\mathbf{F}_t = \mathcal{F}_{\text{vid}}(\mathbf{H}_t, \ell), \qquad \widehat{\mathbf{A}}_t = \mathcal{P}(\mathbf{I}_t, \mathbf{s}_t, \ell; \mathbf{F}_t). \tag{2}$$

The history tensor $\mathbf{H}_t \in \mathbb{R}^{B\times m\times 3\times h\times w}$ contains m observed frames. The frozen video generation network's feature extractor, $\mathcal{F}_{\text{vid}}(\cdot)$, returns $\mathbf{F}_t \in \mathbb{R}^{B\times L_v\times N_v\times d_v}$, where L_v is the number of selected source blocks, N_v their token count, and d_v their hidden width. Only the bridge, gates, and selected policy adapters are optimized.

Video-model states can contain appearance and generative detail unrelated to an action. The bridge therefore learns which source information to expose to a policy query.

3.2 K/V PROJECTION AND MAGNITUDE MATCHING

We form the source history from m observed frames:

$$\mathbf{H}_t = (\mathbf{I}_{t-\tau_{m-1}}, \ldots, \mathbf{I}_{t-\tau_1}, \mathbf{I}_t). \tag{3}$$

The offsets satisfy $0 < \tau_1 < \cdots < \tau_{m-1}$. The video generation network extracts history-conditioned states using the task prompt. We arrange the selected hidden states as N_v context tokens per source block and project them into policy-side keys and values. In the frame-wise extraction used here, states from the observed frames are concatenated along the context-token axis.

Let j index a receiving layer and $b_j \in \{0, \ldots, L_v - 1\}$ select an entry in the source-block list. The layer-specific selection function $\mathcal{B}_j(\cdot)$ extracts the corresponding hidden-state tensor:

$$\mathbf{F}_j = \mathcal{B}_j(\mathbf{F}_t) = \mathbf{F}_t[:, b_j, :, :] \in \mathbb{R}^{B\times N_v\times d_v}. \tag{4}$$

A compact bottleneck $\mathcal{P}_j$ followed by layer-specific linear heads produces

$$\overline{\mathbf{K}}_j^{\text{vid}} = \mathcal{P}_j(\mathbf{F}_j)\mathbf{W}_j^K, \qquad \overline{\mathbf{V}}_j^{\text{vid}} = \mathcal{P}_j(\mathbf{F}_j)\mathbf{W}_j^V \in \mathbb{R}^{B\times N_v\times D_{kv}}. \tag{5}$$

The bottleneck output has shape $B \times N_v \times r$, and $\mathbf{W}_j^K, \mathbf{W}_j^V \in \mathbb{R}^{r\times D_{kv}}$. The bottleneck uses conventional normalization, a linear map $\mathbf{W}_{s,j} \in \mathbb{R}^{d_v\times r}$, and GELU; biases are omitted here. The map can be shared across source blocks or instantiated per block; Supplementary Section H specifies the normalization and sharing choices. Separate heads distinguish the retrieval map from the transferred content (Vaswani et al., 2017).

We split projected K/V into n_{kv} heads of width d, with $D_{kv} = n_{kv}d$. If the receiver uses n_h query heads, grouped-query expansion repeats the K/V heads to shape $B \times n_h \times N_v \times d$. Native K/V have shape $B \times n_h \times N_n \times d$, where N_n is native context length; the query width is $D = n_h d$. In the illustrated VLM-backbone placement, we additionally match the projected key and value magnitudes to their native counterparts, separately per head:

$$\mathbf{U}_j^{\text{vid}} = \overline{\mathbf{U}}_j^{\text{vid}} \odot \min\left(c, \frac{\mathcal{R}(\mathbf{U}_j^N)}{\mathcal{R}(\overline{\mathbf{U}}_j^{\text{vid}}) + \epsilon}\right), \qquad c > 0. \tag{6}$$

Here $\mathbf{U}$ denotes either keys or values, and the superscript N identifies the corresponding native tensor. The function $\mathcal{R}(\cdot)$ returns per-head root-mean-square (RMS) magnitudes in $\mathbb{R}^{B\times n_h\times 1\times 1}$, broadcast over tokens and channels; $\epsilon > 0$ prevents division by zero. The scale is detached from the gradient. Matching key magnitude controls attention-logit scale, while matching value magnitude controls the video branch's contribution. The cap limits amplification of a near-zero projected feature. Learned projection supplies semantic correspondence; magnitude matching addresses numerical compatibility.

For receiver blocks with built-in query/key normalization, we reuse that normalization in the video branch. Source-block selection, normalization, and receiving-layer placement adapt the interface to each network pair; Supplementary Section H specifies our configuration. Joint permutation of projected key/value tokens leaves unmasked receiver attention unchanged; Supplementary Section G derives this property.

3.3 Gated additive attention in the VLM backbone

Let N_q be the number of receiving queries. The shared query tensor is $\mathbf{Q}_j \in \mathbb{R}^{B\times n_h\times N_q\times d}$. Native and video-prior contexts use separate attention normalizers:

$$\mathbf{O}_j^N = \mathcal{S}\left(\mathbf{Q}_j(\mathbf{K}_j^N)^\top/\sqrt{d}\right)\mathbf{V}_j^N, \tag{7}$$

$$\mathbf{O}_j^{\text{vid}} = \mathcal{S}\left(\mathbf{Q}_j(\mathbf{K}_j^{\text{vid}})^\top/\sqrt{d}\right)\mathbf{V}_j^{\text{vid}}, \tag{8}$$

$$\widetilde{\mathbf{O}}_j = \mathbf{O}_j^N + g_j\mathbf{O}_j^{\text{vid}}, \qquad g_j \in [0,1]. \tag{9}$$

The superscripts N and vid identify the native and video-prior branches. Here $\mathcal{S}(\cdot)$ is softmax over context tokens and transposition acts on the last two axes. Native and video-prior attention maps have shapes $B \times n_h \times N_q \times N_n$ and $B \times n_h \times N_q \times N_v$, respectively. Both outputs and their sum lie in $\mathbb{R}^{B\times n_h\times N_q\times d}$.

The $1/\sqrt{d}$ factor controls dot-product scale (Vaswani et al., 2017). Separate softmax operations prevent added video-prior tokens from competing for the native branch's probability mass. The two branches produce contexts in the same query space, and their outputs are added after the video branch is scaled by a learned gate. This follows gated cross-attention interfaces between frozen backbones and new modalities (Alayrac et al., 2022). The scalar gate g_j starts at zero, preserving native retrieval at initialization and allowing the new context to enter through learning.

The head-concatenation function $\mathcal{C}(\cdot)$ and native output matrix $\mathbf{W}_j^O \in \mathbb{R}^{D\times d_p}$ return query representations to the receiving backbone:

$$\widetilde{\mathbf{X}}_j = \mathcal{C}(\widetilde{\mathbf{O}}_j)\mathbf{W}_j^O \in \mathbb{R}^{B\times N_q\times d_p}. \tag{10}$$

Here d_p is the receiver hidden width. The gate controls how much video context enters the combined representation. The remaining receiver layers and original action-generation architecture complete the policy computation. The receiving queries depend on the injection site: visual-attention blocks use visual-token queries, whereas the VLM transformer trunk can provide action-query tokens.

3.4 Policy adaptation and training objective

The projector makes video features accessible to the receiver; optional VLM-LoRA adapts how the VLM backbone uses that context (Hu et al., 2021). We attach low-rank updates to selected VLM projections, optimizing them jointly with the projector and gates under the receiver's existing

objective. The video network and base policy weights remain frozen. Section 4 specifies the adapter locations for each receiving policy; Supplementary Section F gives the parameter budgets. Let θ contain the projector, gates, and enabled LoRA parameters. We minimize the receiving policy's objective over demonstrations $\mathcal{D}$:

$$\theta^\star = \arg\min_\theta \ \mathbb{E}_{\mathcal{D}}\left[\mathcal{L}_{\text{receiver}}\left(\theta; \mathbf{H}_t, \mathbf{I}_t, \mathbf{s}_t, \ell, \mathbf{A}_t, \mathbf{Z}_t\right)\right]. \tag{11}$$

The additional targets $\mathbf{Z}_t$ contain the receiver's bridge or future-feature supervision. We retain each receiving policy's original loss terms and weights without modification. For DIAL, we adopt its action and bridge objectives (Chen et al., 2026):

$$\mathcal{L}_{\text{DIAL}} = \tfrac{1}{2}\left(\mathcal{L}_{\text{action}} + \mathcal{L}_{\text{bridge}}\right). \tag{12}$$

For WALA, we retain its action, alignment, and decoding objectives (Liu et al., 2026):

$$\mathcal{L}_{\text{WALA}} = \mathcal{L}_{\text{action}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}} + \lambda_{\text{decode}}\mathcal{L}_{\text{decode}}. \tag{13}$$

Here $\mathcal{L}_{\text{receiver}}$ is the objective of the selected receiving policy. WALA's action term is mean absolute error (MAE) over the predicted action tensor $\widehat{\mathbf{A}}_t$ and target $\mathbf{A}_t \in \mathbb{R}^{B\times T\times d_a}$; its auxiliary terms align predicted and target transition representations and decode visual/depth changes. The weights λ_{align} and λ_{decode} follow the original WALA configuration.

4 Experiments

Development screens cover 24 RoboCasa-GR1-Tabletop tasks with five fixed episodes each unless stated otherwise; the full benchmark uses 50 episodes per task. SR denotes episode success rate, and differences are percentage points (pp) from unrounded counts.

Receiving policies and optimization. We instantiate the video generation network with MiniMax-H3 (H3) (MiniMax, 2026). H3 and pretrained receiver weights remain frozen. We optimize the projector, gates, and selected policy adapters using each receiver's released objective. Source quantization changes storage and feature computation while preserving the trainable parameter count. Historical fixed-gate training-loss diagnostics are reported in Supplementary Section C.

DIAL (Chen et al., 2026) supplies the first receiver. Its bridge and selected adapters are trained with BF16 H3 features; Section 4.1 fixes that policy and switches source precision at evaluation. The broader development sweep varies receiving layers and source mappings (Supplementary Section B).

WALA (Liu et al., 2026) combines a Qwen3-VL-4B backbone and a multilayer perceptron (MLP) action head. We compare projector/gate adaptation with additional VLM-LoRA (Hu et al., 2021). Native is the released policy; OFF retains jointly learned adapters and disables the video branch. The MLP predicts $\widehat{\mathbf{A}}_t \in \mathbb{R}^{B\times 32\times 47}$, with 29 executed coordinates per action. A separate WALA VLM-LoRA configuration uses rank-32 adapters in the VLM backbone. Supplementary Section F reports the parameter inventory for each configuration. Supplementary Section H specifies coordinate mappings and adapter scopes.

4.1 DIAL: Source Precision and the Resource Trade-off

Table 1: DIAL source-precision screen.

H3	SR (%) ↑	Δ vs. OFF (pp)
OFF	75.0	0.0
INT8	75.8	+0.8
BF16	**80.8**	**+5.8**

The DIAL policy was trained with features from a frozen BF16 H3 backbone. Table 1 keeps the policy and development episodes fixed while changing H3 at evaluation. BF16 reaches 80.8%, a 5.8-point gain over the same policy's OFF condition.

Test-time INT8 reaches 75.8%, five points below BF16. This test introduces a feature-distribution shift relative to the BF16 features used to train the policy.

In separate timing tests, INT8 reduces fixed-input H3 extraction time by $10.5\times$, policy-request time by $5.9\times$, and optimizer-update time by $5.4\times$. The training deployments differ in device placement and server configuration; Supplementary Section F gives the hardware and workloads for each ratio.

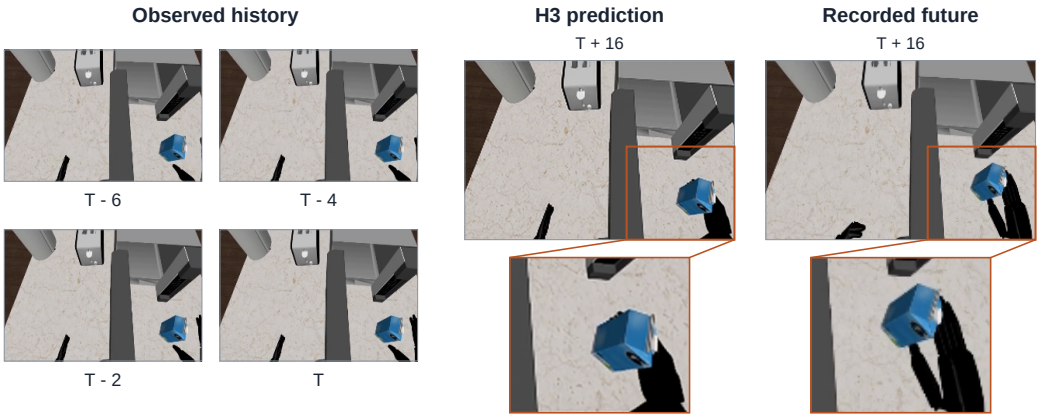

Figure 3: RoboCasa-GR1 placement case: observed history (left), H3 output and recorded future at $T + 16$ (right). Identical crops remove letterbox borders; boxes mark matched object regions enlarged below. Metrics use the original full frames.

Table 2: Image quality in the RoboCasa-GR1 case study.

Input / prediction	MAE ↓	PSNR ↑
Copy current T	8.497	17.553
H3, four-frame BF16	**7.802**	**19.447**

Table 3: WALA adaptation configurations. 24 tasks × 5 episodes.

Variant	Proj.	Gate	LoRA	SR (%) ↑
Native	×	×	×	70.8
+ Projector/gate	✓	✓	×	71.7
+ VLM-LoRA	✓	✓	✓	**73.3**

We therefore use INT8 features during both training and evaluation in the subsequent adaptation experiments. The bridge learns from the source representation available at deployment, with reduced weight storage and feature-extraction cost.

4.2 Visual quality of H3 futures

We compare a BF16 H3 prediction with the recorded future in a RoboCasa-GR1 placement case.

Figure 3 pairs four observed frames ($T-6, T-4, T-2, T$) with the prediction and recorded $T + 16$ frame. Enlarged object regions show spatial differences. Table 2 reports MAE and peak signal-to-noise ratio (PSNR) on the original full frames.

H3 reduces MAE from 8.497 to 7.802 and raises PSNR from 17.553 to 19.447 relative to copying $\mathbf{I}_T$ in this case. Copying the current frame provides a static reference: it retains the visible scene without predicting subsequent motion. The full-frame metrics complement the enlarged crops, which make local object-placement differences easier to inspect.

4.3 Adaptation configurations

Table 3 compares three WALA configurations on the 24-task development set.

Native uses the released WALA policy and a reused local reference. The adapted rows use selected checkpoints from separate training runs. The projector/gate configuration scores 71.7%; the configuration with additional VLM-LoRA in visual-attention blocks scores 73.3%. The first configuration adapts the video-to-policy interface while keeping the receiver weights frozen. VLM-LoRA also adjusts how the receiver processes its context. Both retain the pretrained action generator, so the comparison concerns the scope of adaptation within the same policy family.

Closed-loop cases. Figure 4 illustrates successful placements with H3 and VLM-LoRA through paired initial and terminal observations.

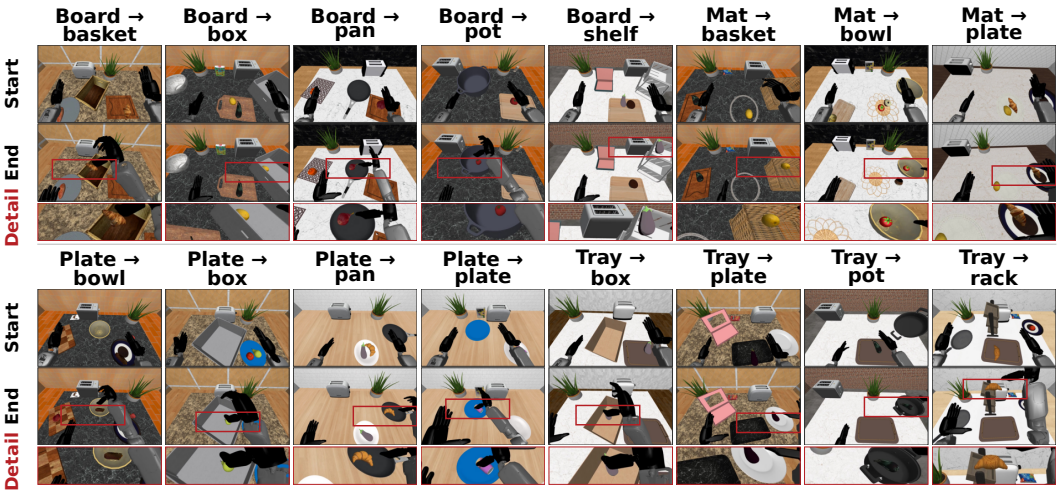

Figure 4: Successful object placements. Each column shows the initial state, terminal state, and enlarged terminal view from one recorded episode.

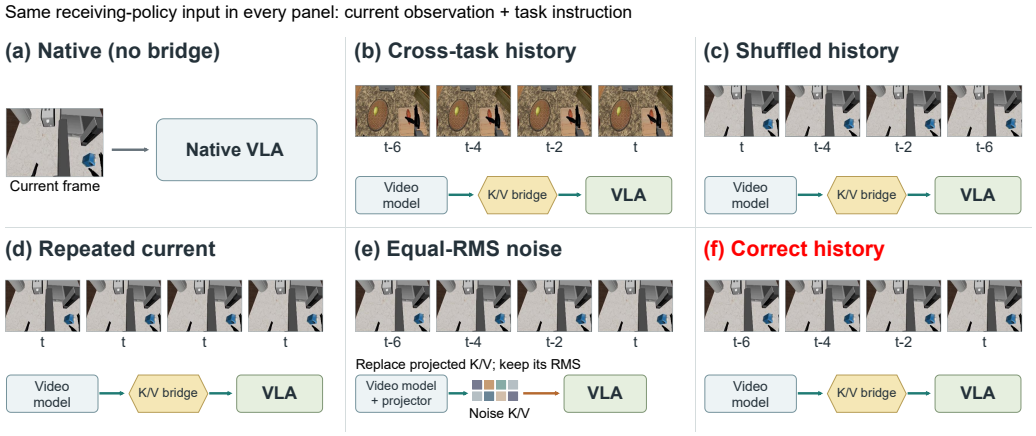

Figure 5: History-content interventions in Table 4, illustrated with recorded RoboCasa-GR1 frames. Cross-task input substitutes donor history and its prompt in the video branch; the receiver keeps the current observation and task. Shuffling uses order $(T, T-4, T-2, T-6)$; repetition uses four current frames. Equal-RMS noise replaces projected K/V. Native uses the released policy without a bridge. Frames illustrate these operations rather than the exact evaluated episodes.

4.4 History-content controls

We perturb the injected history while keeping the adapted policy fixed. The exploratory test covers six tasks with five paired episodes each. Correct-history and Native outcomes reuse the same task–seed pairs; the other conditions are new rollouts.

Table 4: History-content screen. SR (%); Δ is relative to Native.

Input condition	SR ↑	Δ (pp)
Native (no bridge)	70.0	–
Cross-task history	66.7	-3.3
Shuffled history	70.0	0.0
Repeated current	66.7	-3.3
Equal-RMS noise	66.7	-3.3
Correct history	**76.7**	**+6.7**

Correct history yields the highest success rate (76.7%) in this six-task screen. The paired exact McNemar tests give $p = 0.25$–0.625. Cross-task input changes the video branch's history and prompt; noise preserves projected K/V RMS. Input shuffling probes the extraction route because the receiver is invariant to jointly permuting K/V. Supplementary Section B records paired outcomes and selection details.

4.5 RoboCasa benchmark and deployment cost

Benchmark comparison. With rank-32 VLM-LoRA, WALA-based VID-Policy reaches 72.33% SR, compared with 70.50% for our WALA re-implementation. The evaluation covers 24 tasks with 50 episodes per task; the 1.83-point gain corresponds to 22 additional successes, with gains on 12 tasks. Supplementary Section A gives the protocol and per-task results.

Table 5: RoboCasa-GR1-Tabletop success rate (%).

Method	SR (%) ↑
StarVLA Qwen3-PI (Community, 2026)	43.90†
GR00T-N1.6 (NVIDIA GEAR Team, 2025)	47.60†
StarVLA Qwen3-OFT (Community, 2026)	48.80†
DiT4DiT (Ma et al., 2026)	50.80†
LDA-1B (Lyu et al., 2026)	55.40†
FrameSkip (Yu et al., 2026)	59.50†
DIAL (Chen et al., 2026)	70.20†
WALA (Liu et al., 2026)	75.20†
WALA (our re-implementation)	70.50‡
VID-Policy	**72.33**‡

† Published results use their reported settings; ‡ denotes our evaluations.

The task-level results show how this aggregate gain is distributed. Seven tasks have lower success rates with VID-Policy, while five are unchanged. The largest increase is 22 percentage points on task 23, followed by 12 points on task 10 and 10 points on task 16. Because every task contributes the same number of episodes, the overall score weights these task-level changes equally. The improvement therefore combines gains and losses across the benchmark, with the complete distribution shown in Supplementary Figure S1.

Deployment cost. Section 4.1 reports inference and update times; Supplementary Section F gives hardware and workload details.

The frozen H3 DiT has 33.123B logical weights (61.73 GiB BF16; 31.70 GiB INT8, 48.6% less storage at unchanged logical count). WALA-A trains 15.335M projector/gate parameters; WALA-B trains 16.515M total, including 1.180M VLM-LoRA. The separate rank-32 configuration in Figure 1 trains 81.40M parameters. Feature extraction adds inference cost; timing and memory depend on workload, precision, and placement. Logical parameter count records the size of the pretrained representation resource, whereas weight storage depends on its numerical precision. Trainable parameter count determines the scope of adaptation but does not include the frozen source weights that must remain available for feature extraction. The bridge thus reduces the number of parameters updated during adaptation while retaining access to a much larger source model. Evaluating this trade-off requires considering source execution together with receiver inference, rather than interpreting the adaptation budget as the total deployed model size.

5 Conclusion & Discussion

VID-Policy transfers frozen video representations to pretrained VLAs through projected keys and values with gated additive attention, preserving the action generator while avoiding future-frame sampling. A compact bridge, optional VLM-LoRA, and source quantization keep receiver adaptation separate from the scale of the frozen video model. In the fixed-seed DIAL precision screen, BF16 features reach 80.8% success, 5.8 percentage points above the same adapted policy's OFF condition. On the full RoboCasa-GR1-Tabletop benchmark, WALA-based VID-Policy reaches 72.33%, about 1.83 percentage points above our WALA re-implementation. In the six-task history-content screen, correctly paired history yields 76.7% success versus 66.7–70.0% across controls. These DIAL and WALA results come from different policy families and protocols, providing complementary evidence rather than a direct cross-family ranking.

Discussion. The full-benchmark result comes from simulated RoboCasa-GR1-Tabletop; DIAL precision and history-content findings come from focused screens. The task-level variation also motivates studying when video context is useful to the receiving policy. Extending the current evaluation can distinguish benefits that persist across tasks from those tied to particular observation histories. Source precision and interface placement provide further choices for balancing predictive features against the resources required to extract them.

Reproducibility statement

We record task and seed manifests, model and statistics identities, checkpoint hashes, trainable-parameter scopes, action mappings, exception counts, videos, and per-step diagnostics. The supplementary material specifies the controls, evaluation freeze, historical result provenance, and efficiency measurements needed to reproduce each claim. Final artifacts will include an anonymous code snapshot and the exact evaluation contract.

AI use statement

We used AI tools solely to improve the language and readability of the manuscript. All research contributions, experimental work, and analyses were carried out by the authors. The authors reviewed all AI-assisted edits and take full responsibility for the final manuscript.

References

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025. 1, 2

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023. 2

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022. 2, 5

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15619–15629. IEEE, 2023. 2

Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 conference papers*, pp. 1–11, 2024. 2

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024. 2

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. 2

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. 2

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 2

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 2

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. 1, 2

Yi Chen, Yuying Ge, Hui Zhou, Mingyu Ding, Yixiao Ge, and Xihui Liu. Dial: Decoupling intent and action via latent world modeling for end-to-end vla. *arXiv preprint arXiv:2603.29844*, 2026. 6, 9

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. 2

StarVLA Community. StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing. *arXiv preprint arXiv:2604.05014*, 2026. 9

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022. 2

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022. 2

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023. 2

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023. 2

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023. 1, 2, 3

Finn Ferchau, Daniel Pommer, and Cristian Axenie. On the Efficiency of LoRA Fine-Tuning for Vision-Language-Action Models in Industrial Robotic Manipulation. In *Artificial Neural Networks and Machine Learning–ICANN 2026: 35th International Conference on Artificial Neural Networks, Padua, Italy, September 14–17, 2026, Proceedings, Part V*, pp. 191. Springer Nature, 2026. 3

Yuxia Fu, Zhizhen Zhang, Yuqi Zhang, Zijian Wang, Zi Huang, and Yadan Luo. Mergevla: Cross-skill model merging toward a generalist vision-language-action agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22335–22347, 2026. 3

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018. 2

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a. 2

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019b. 2

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. 2

Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. BAKU: An Efficient Transformer for Multi-Task Policy Learning. *arXiv preprint arXiv:2406.07539*, 2024. URL https://arxiv.org/abs/2406.07539. 2

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024. 2

Haotian He, Zeyu Yan, Qipeng Liu, Ning Guo, and Wenzhao Lian. FAWAM: Force-Aware World Action Models for Closed-Loop Contact-Rich Manipulation. *arXiv preprint arXiv:2606.08555*, 2026. 2

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022. 1, 2

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in neural information processing systems*, 35:30016–30030, 2022. 1

Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. GAIA-1: A Generative World Model for Autonomous Driving. *arXiv preprint arXiv:2309.17080*, 2023. URL https://arxiv.org/abs/2309.17080. 2

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2, 5, 6

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. *arXiv preprint arXiv:2412.14803*, 2024. 1, 2

Haoxu Huang, Tongsam Zheng, Yifan Chen, Jiacheng You, and Yang Gao. Critical Interval MSE: Toward Reliable Offline Validation for Robot Manipulation Policies. *arXiv preprint arXiv:2606.29898*, 2026. 3

Dong Jing, Jingchen Nie, Tianqi Zhang, Jiaqi Liu, Huaxiu Yao, Zhiwu Lu, and Mingyu Ding. Tempovla: Learning speed-controllable vision-language-action policies. *arXiv preprint arXiv:2606.06491*, 2026. 2

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

Siddharth Karamcheti, Suraj Nair, Annie S Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023. 2

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. 1, 2

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023. 2

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024. 2

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024. 2

Fenghao Lei, Zhixiong Huang, Long Yang, Jiabao Chen, Peilin Huang, Han Fu, Zhuo Li, and Xiaoxue Ren. DELE-w0. 5: Inferring Action from Future Latent State for Robotic Manipulation. *arXiv preprint arXiv:2608.22067*, 2026. 2

Jiahao Liu, Zhongpu Xia, Shuai Tian, Huangrui Li, Yuhang Zheng, Ning Ma, Xin Fu, Xiaotian Liu, Jing Li, Yixian Li, et al. WALA Learning Executable Latent Actions from Action-Labeled Demonstrations and Action-Free Videos. *arXiv preprint arXiv:2607.11397*, 2026. 2, 6, 9

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025. 2

Jiangran Lyu, Ziming Li, Xuesong Shi, Chaoyi Xu, Yizhou Wang, and He Wang. Dywa: Dynamics-adaptive world action model for generalizable non-prehensile manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11058–11068. IEEE, 2025. 2

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026. 9

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026. 9

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022. 2

Oier Mees, Dibya Ghosh, Karl Pertsch, Kevin Black, Homer Rich Walke, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *First workshop on vision-language models for navigation and manipulation at ICRA 2024*, 2024. 2

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022. 2

MiniMax. MiniMax-H3: Official model card. Hugging Face model card, 2026. URL https://huggingface.co/MiniMaxAI/MiniMax-H3. Accessed September 25, 2026. 6

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022. 2

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024. 3

NVIDIA GEAR Team. GR00T N1.6: An improved open foundation model for generalist humanoid robots. NVIDIA Research, December 2025. URL https://research.nvidia.com/labs/gear/gr00t-n1_6/. Released December 15, 2025; accessed September 25, 2026. 9

Abdullah Yahya Abdullah Omaisan and Ibrahim Sheikh Mohamed. Towards Accessible Physical AI: LoRA-Based Fine-Tuning of VLA Models for Real-World Robot Control. *arXiv preprint arXiv:2512.11921*, 2025. 3

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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024. 1, 2

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. 2

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. 2

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. 2

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. 2

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *European Conference on Computer Vision*, pp. 478–497. Springer, 2026. 2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4, 5

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 1, 2

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023. 2

Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 2

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021. 2

Fan Yang, Yuting Su, Xiaobo Wang, Yuncheng You, Fugui Fan, Yuting Wu, Minghui Wu, Chenxu Zhao, JiaHong Ning, and Peiguang Jing. LiLa-WAM: Lightweight Latent Reasoning World-Action Model for Robotic Manipulation. *arXiv preprint arXiv:2608.03701*, 2026a. 2

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026b. 2

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, et al. Latent action pretraining from videos. In *International Conference on Learning Representations*, volume 2025, pp. 28213–28239, 2025. 2

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. 2

Bin Yu, Shijie Lian, Xiaopeng Lin, Zhaolong Shen, Yuliang Wei, Changti Wu, Hang Yuan, Haishan Liu, Bailing Wang, Cong Huang, et al. FrameSkip: Learning from Fewer but More Informative Frames in VLA Training. *arXiv preprint arXiv:2605.13757*, 2026. 9

Ruiteng Zhao, Zhengshen Zhang, Yue Su, Wenshuo Wang, Jiahui Li, Zhiyuan Yang, Francis EH Tay, Marcelo H Ang Jr, and Haiyue Zhu. SG-WAM: Self-Guided World Modeling in Geometry-Aware Policy Space. *arXiv preprint arXiv:2608.01397*, 2026. 2

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 2

Pengfei Zhou, Shengcong Chen, Di Chen, Jiaxu Wang, Rongjun Jin, Bingwen Zhu, Yike Pan, Songen Gu, Kuanning Wang, Shufeng Nan, Xingyu Qiu, et al. τ_0-WM: A Unified Video-Action World Model for Robotic Manipulation. *arXiv preprint arXiv:2606.01027*, 2026. URL https://arxiv.org/abs/2606.01027. 2

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*, 2024. 1, 2, 3

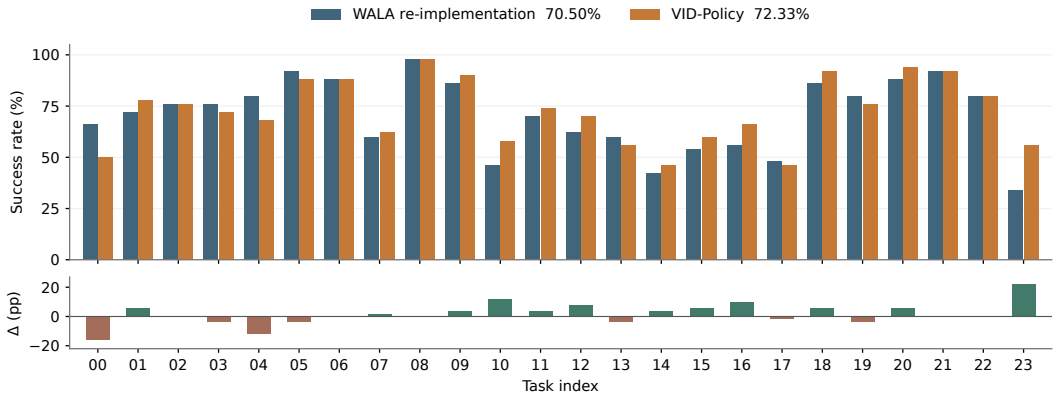

Figure S1: Full-benchmark comparison on RoboCasa-GR1-Tabletop. Each task uses 50 episodes for each method. The upper panel shows success rates, and the lower panel shows VID-POLICY minus WALA in percentage points. Task indices follow the evaluation manifest.

A EVALUATION PROTOCOLS

The WALA re-implementation and WALA-based VID-POLICY were evaluated on all 24 RoboCasa-GR1-Tabletop tasks with 50 episodes per task (1,200 total per method). Both runs used the same fixed seeds (0–49), task-specific initial scenes, GPU evaluation setup, and 256×256 observation resolution. Each episode starts from a fresh environment. WALA succeeded in 846 episodes (70.50%), and VID-POLICY succeeded in 868 (72.33%), yielding a gain of 1.83 percentage points. Both evaluations completed without exceptions. Figure S1 shows the per-task success rates and their differences.

The largest task-level gain is 22 percentage points on task 23, followed by 12 points on task 10 and 10 points on task 16. Seven tasks have lower success rates with VID-POLICY, while 12 improve and five are unchanged. Reporting all 24 tasks makes the distribution behind the aggregate score visible. These full-benchmark results use the rank-32 VLM-LoRA configuration; the development screens below retain their original checkpoints and episode counts.

B ADDITIONAL EVALUATION RESULTS

Supplementary Section A reports the local full-benchmark results and their evaluation conditions. Table 5 presents the full-benchmark comparison alongside published results. The following development screens probe individual design choices on smaller evaluation sets.

The DIAL comparison in Section 4.1 uses policy checkpoint 270, trained with the frozen full BF16 H3 diffusion transformer (DiT) and the original objective. On the same 24-task, five-episode development set, OFF succeeds on 90 episodes, test-time INT8 on 91, and BF16 on 97. The separately released Native DIAL succeeds on 92 episodes. OFF refers to the adapted policy with its video branch disabled; all scores in this comparison use the development set.

Table S1 summarizes evaluations across source mappings, precision settings, and adaptation configurations. WALA ON/OFF comparisons use the same checkpoint. Fixed-gate runs span changes in preprocessing, statistics, adapter scope, and code.

At the matched 1→2 checkpoint 1020, BF16 obtains 95/120 and INT8 obtains 90/120. The variants agree on 101 episodes; BF16 alone succeeds on 12 and INT8 alone on seven. On the recovered appearance-OOD set, Native obtains 65/90 and step-770 INT8 obtains 64/90. Each comparison uses one checkpoint under the stated precision or appearance condition.

For the six-task Scheme-A content screen, the correct-only/control-only discordant counts are 3/1 against Native, 4/1 against mismatch, 2/0 against shuffle, 3/0 against repeated current, and 4/1

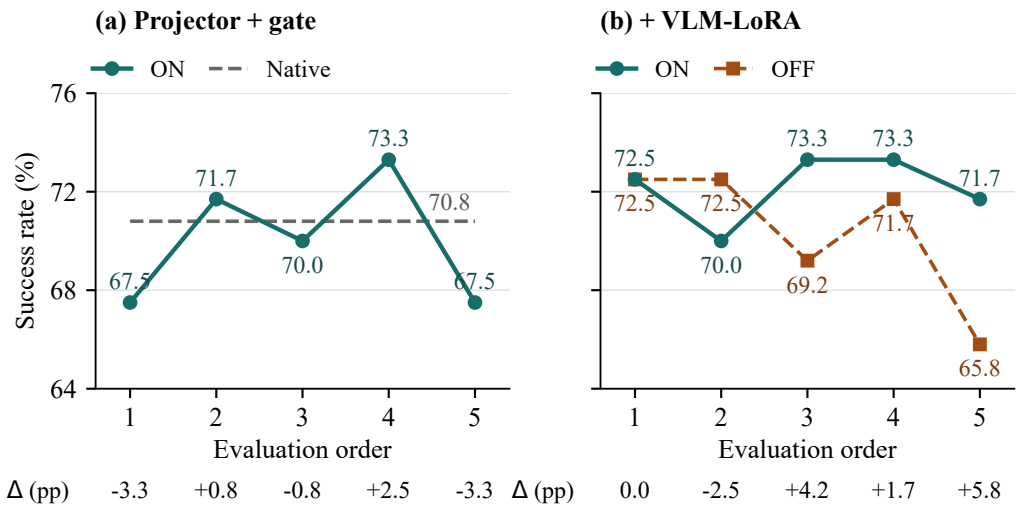

Figure S2: WALA adaptation screens on 24 tasks with five episodes each. Five checkpoint evaluations are shown in their original order; horizontal spacing indicates evaluation order, not training distance. (a) Projector/gate ON versus the reused Native reference (70.8%). (b) Additional VLM-LoRA, with ON/OFF evaluated at the same checkpoint. Labels give success rates; Δ gives the recorded ON-minus-reference difference in percentage points.

Table S1: Experiment families. Conditions count reported configurations, not independent training seeds.

Family	Conditions	Status
DIAL+H3	19	24×5 historical screens
DIAL appearance OOD	2	90 episodes per condition
WALA corrected	15	A-ON, B-ON/OFF at five checkpoints
WALA fixed Gate= 1	12+	diagnostic; several partial runs
Predictive-content screen	6	six tasks, five seeds; reference reuse
Training traces	>1,300 steps/run	training-batch diagnostics

Table S2: Additional DIAL screens retained for provenance.

Run	Step	Success	SR	Run	Step	Success	SR
1→1 BF16	320	83	69.17	late INT8 S1	910	87	72.50
1→1 INT8	480	83	69.17	late INT8 S1	1130	89	74.17
1→1 INT8	710	89	74.17	late INT8 S2	890	81	67.50
1→1 INT8	770	87	72.50	late INT8 S2	1110	88	73.33
block16–36 INT8	690	86	71.67	1→1 INT8	1530	87	72.50
1→2 INT8	950	87	72.50	1→2 INT8	970	86	71.67
1→2 INT8	980	90	75.00	–	–	–	–

against equal-RMS noise. These are the pairs used in the exploratory exact McNemar tests in Section 4.4.

C Fixed-Gate Diagnostics

Early WALA experiments fixed the gate at one while other implementation factors changed. These diagnostics are limited to training batches; held-out executed-action and auxiliary-target errors are not assessed here.

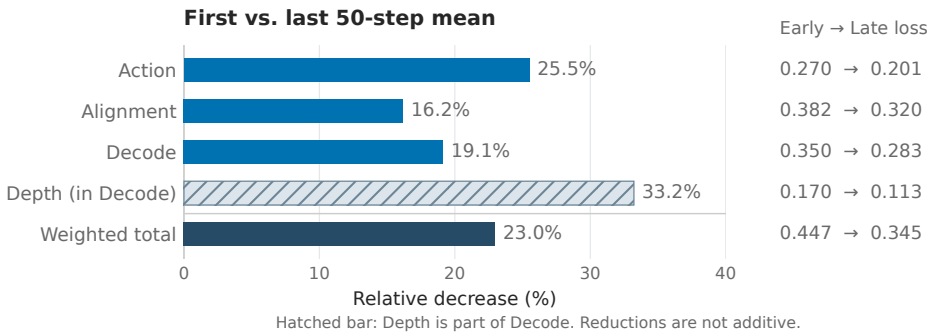

Figure S3: Historical loss reductions: $100(1 - L_{\text{late}}/L_{\text{early}})$, using H3-ON post-update microbatch means over steps 1–50 and 932–981. Depth is part of Decode; reductions are not additive. Means: Table S4.

Table S3: Recovered fixed-Gate diagnostics. Partial rows include completed episodes only.

Generation	Condition	Success	Total
v4 task-00	Native / H3 / OFF	4 / 0 / 0	5 each
v5 task-00	Action-LoRA only	0	5
v5 task-00	VLM-LoRA only	4	5
v5 task-00	Projector only	2	5
v5 task-00	H3 + VLM-LoRA	3	5
v6 task-00	step919/1020 ON; step1020 OFF	0/0/0	5 each
v3 partial	Projector / paired Native	8/12	15
v3 partial	H3+VLM-LoRA / paired Native	14/15	23
Historical full	Native / adapted policy, step 919 OFF	86/11	120

Table S4: Fixed-Gate training losses: first and last 50-step means.

Metric	First 50	Last 50	Change
Action	0.2704	0.2014	-25.5%
Alignment	0.3820	0.3202	-16.2%
Decode	0.3503	0.2833	-19.1%
Depth decode	0.1699	0.1135	-33.2%
Total	0.4474	0.3446	-23.0%

D Adaptation Comparison Details

In Table 3, the WALA native reference is reused (85/120). The projector/gates run at checkpoint 600 scores 86/120, and the projector/gates+VLM-LoRA run at checkpoint 1000 scores 88/120. The adapted rows come from separate training runs and use the selected checkpoints listed above. Figure S2 retains the broader checkpoint results.

In the WALA ON/OFF comparison, both settings retain the jointly trained weights, including VLM-LoRA; OFF disables the video branch at inference.

E Visual Generation Evidence

The example in Figure 3 uses trajectory 989 from the RoboCasa-GR1 milk-to-microwave task. Its four history indices are 64, 66, 68, and 70, and the recorded target is index 86. The prediction and recorded target both correspond to $T + 16$. Image metrics in this section assess generated RGB frames; policy success is evaluated through the closed-loop tests in Section 4.5.

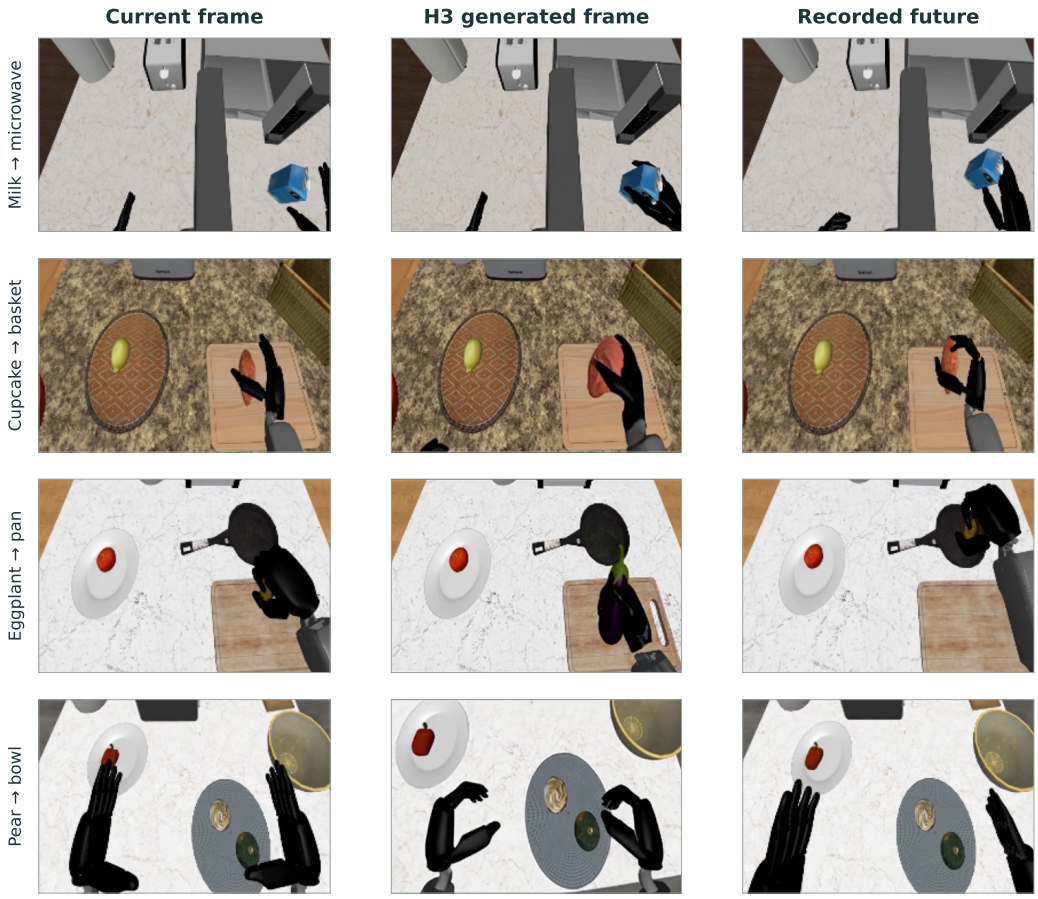

Figure S4: All four RoboCasa standalone H3 predictions: current observation, generated image, and recorded future. The scored generated frame is 0.792 s after the current frame, 8.3 ms before the 0.8 s target. Letterbox borders are cropped only for display; reported metrics use the complete frames. This single-frame probe is separate from the four-history-frame example in Figure 3.

A separate four-case standalone BF16 H3 probe uses a single current frame as input and scores the generated frame nearest the recorded $T + 16$ target. Figure S4 shows every tested case. In the milk-to-microwave case H3 improves full-frame MAE and PSNR over copying the current frame (0.03138 versus 0.03332; 18.78 versus 17.55 dB). Across all four cases, copying the current frame scores better on average (MAE 0.06290 versus 0.08606; PSNR 15.92 versus 15.10 dB).

F Efficiency and Resource Reporting

Parameter accounting. OpenVLA's 7B value is the reported policy scale. UniPi's simulated setting reports 1.7B base video, 1.7B temporal super-resolution, and a 4.6B T5 encoder; their 8.0B subtotal excludes the inverse-dynamics model. VPP reports a 1.5B SVD foundation, excluding its added conditioning, Video-Former, and action head. Figure 1 summarizes these component scales under the counting scopes above.

Our released receiver checkpoints contain 5,518,820,352 tensor elements for DIAL and 5,748,023,158 for WALA. The full H3 DiT has 33,122,992,912 logical weights; INT8 scale tensors are excluded from this logical count. WALA's stored checkpoint includes training-only world-model modules, so its inventory exceeds the weights needed solely for action inference. The architecture inventory adds receiver weights, H3 DiT weights, and adapters; it excludes H3's text encoder and variational autoencoder (VAE). Quantization changes storage per weight, not this logical count.

Table S5: Policy-side adaptation budgets. Current configurations match the component inventory described above; historical rows document the earlier diagnostic settings.

Configuration	Projector/gates	LoRA	Total trainable
DIAL (historical)	21,080,556	16,777,216	37,857,772
WALA-A	15,335,436	0	15,335,436
WALA-B	15,335,436	1,179,648	16,515,084
WALA VLM-LoRA	15,335,436	66,060,288	81,395,724

Table S6: Archived timing and memory measurements. Each row reports its recorded workload; the training deployments also differ in device placement and server configuration.

Measurement	BF16	INT8	BF16/INT8
Four-frame H3 extraction (s)	10.028	0.952	10.53×
Process peak allocated (GiB)	56.85	49.37	1.15×
Policy request, four-task rollout (s)	9.168	1.547	5.93×
Optimizer update, deployment mean (s)	402.675	75.269	5.35×

The WALA VLM-LoRA row uses rank 32. Its trainable inventory agrees with the optimizer log (81.396M total); VLM-LoRA alone is 66.060M. Its instantiated policy reports 5.441B parameters, a different counting scope from the released checkpoint inventory above. The expanded attention diagram and earlier WALA A/B controls use visual-attention blocks. This WALA configuration injects at language-attention layers 12, 14, . . ., 34 and applies rank-32 VLM-LoRA throughout the VLM transformer trunk.

Precision and timing. For N logical weights, the ideal payload is $Nb/8$ bytes at b bits. Applied to the full H3 DiT, this gives 61.70 GiB (BF16) and 30.85 GiB (INT8). These analytic values cover weight payloads. Runtime memory also depends on quantization scales, retained higher-precision tensors, activations, workspace, offload, and the co-resident receiver.

The recorded full-backbone weight files occupy 61.73 GiB (BF16) and 31.70 GiB (INT8), a 48.6% storage reduction including file overhead and retained higher-precision tensors. BF16 and INT8 label the backbone weight formats; exported features may use a different dtype.

Fixed-input probe. Both configurations process the same four RGB frames and prompt through blocks 0, 2, . . ., 22, with native DIAL co-resident in a single Ascend process. We average three warm calls after discarding the first. Conditioning-cache hits exclude text-encoding cold start; memory management/offload remains enabled. Both runs use the same server class and launcher; physical-card identifiers were not recorded. The allocation counter remains active through loading and warm calls, so peak allocated memory is the process high-water mark across both phases. It counts process allocations rather than total device occupancy.

Rollout requests. The four-task comparison fixes policy checkpoint 460, tasks, seeds, history settings, and prompt handling; each precision supplies 176 steady requests with feature caching disabled. Request times are measured during rollouts, where observations can diverge as the policies choose different actions. BF16 succeeds on four tasks and INT8 on three.

Training deployments. Five optimizer updates (461–465) use effective batch 96. The BF16 deployment has four policy workers plus four dedicated H3 service devices; INT8 co-locates policy and H3 on four devices on another server. The observed 5.35× ratio compares these two deployments, including their precision, placement, communication, and server differences. Both use the block-22 extraction route. The DIAL precision screen in Section 4.1 uses block 36 and separate H3 services for BF16 evaluation.

Standalone explicit-generation cost. Table S7 reports the four-case H3 image-generation probe shown in Figure S4. The forward timer is device-synchronized and excludes model loading, image saving, and transfer. Each request uses one A100 80 GB GPU, batch size one, BF16 weights, CPU offload for the DiT and text encoder, and 20 resident DiT blocks.

Table S7: Standalone BF16 H3 generation on four RoboCasa cases. Forward time excludes startup; peak allocated memory is measured on each single-GPU instance. The mean gives each case equal weight.

Case	Synchronized forward (s)	Peak allocated (GiB)
Milk to microwave	308.33	36.02
Cupcake to basket	314.74	36.02
Eggplant to pan	309.33	36.02
Pear to bowl	308.07	36.02
Mean	310.12	36.02

G Receiver Permutation Invariance

Consider one batch element and attention head. Let $\mathbf{P} \in \{0,1\}^{N_v \times N_v}$ be a token-permutation matrix, $\mathbf{Q} \in \mathbb{R}^{N_q \times d}$ the fixed query, and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{N_v \times d}$ the keys and values. Define $\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathcal{S}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d})\mathbf{V} \in \mathbb{R}^{N_q \times d}$. For unmasked attention without token-position terms,

$$\mathcal{A}(\mathbf{Q}, \mathbf{P}\mathbf{K}, \mathbf{P}\mathbf{V}) = \mathcal{S}(\mathbf{Q}\mathbf{K}^\top\mathbf{P}^\top/\sqrt{d})\mathbf{P}\mathbf{V} \tag{S1}$$

$$= \mathcal{S}(\mathbf{Q}\mathbf{K}^\top/\sqrt{d})\mathbf{P}^\top\mathbf{P}\mathbf{V} = \mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}). \tag{S2}$$

Row-wise softmax commutes with a column permutation. Token-wise projection and permutation-invariant RMS statistics preserve this property. It describes the receiver, whereas input shuffling can also change the extraction route. An order-dependent extractor or explicit temporal codes are needed for a shuffle test to isolate temporal reasoning.

H Implementation Details

For WALA Schemes A and B, H3 blocks $(12, 14, 16, 19, 21, 23, 25, 27, 29, 32, 34, 36)$ map in order to visual layers $12, \ldots, 23$. Four-frame history uses offsets $(6, 4, 2, 0)$ ($m = 4$); each of $L_v = 12$ source blocks ($d_v = 5376$) provides $N_v = 64$ tokens. A shared projection maps $5376 \to 512$, and layer-specific K/V heads map $512 \to 1024$; the bottleneck is $r = 512$, $D = 1024$, and the magnitude cap is $c = 20$. Gates initialize at zero, stay in $[0, 1]$, and use FP32 master parameters without weight decay; projected values synchronize to active-device parameters. Scheme A trains the bridge and gates; Scheme B also trains VLM-LoRA in visual-attention blocks. WALA emits 32-step chunks with 47 coordinates per step; the controller executes the first 29, while the remaining 18 supervise auxiliary wrist targets.

WALA VLM-LoRA configuration. Both WALA projector placements preserve native attention, add separately normalized video context, and use token-wise RMS normalization with a shared input map. The rank-32 VLM-LoRA run injects into 12 language-attention layers $(12, 14, \ldots, 34)$ and applies adapters throughout the VLM trunk. Receiver width is $d_p = 2560$ ($D = 4096$ concatenated query width); the video branch reads 32 action queries per sample. The bridge maps $5376 \to 512$, followed by $512 \to 1024$ K/V heads; eight 128-wide heads repeat to match 32 query heads. Receiver-normalized queries and keys feed separate attention, whose gated output returns to action-query positions. Unlike the illustrated placement, this run omits capped native-RMS rescaling.