

Decoupling Intention from Trajectory: A Representational Deduction Framework for World Action Models

Xiangkai Ma¹, Yue Ma², Junjie Wang³, Sheng Xu³, Mingyang Li⁴, Han Zhang¹,
Yuzheng Zhuang⁵, Wenzhong Li^{1✉}, Zhihao Yuan^{5✉}

¹NJU ²HKUST ³CUHK-SZ ⁴THU ⁵Joy Future Academy, JD

<https://pilot-wam-2026.github.io/>

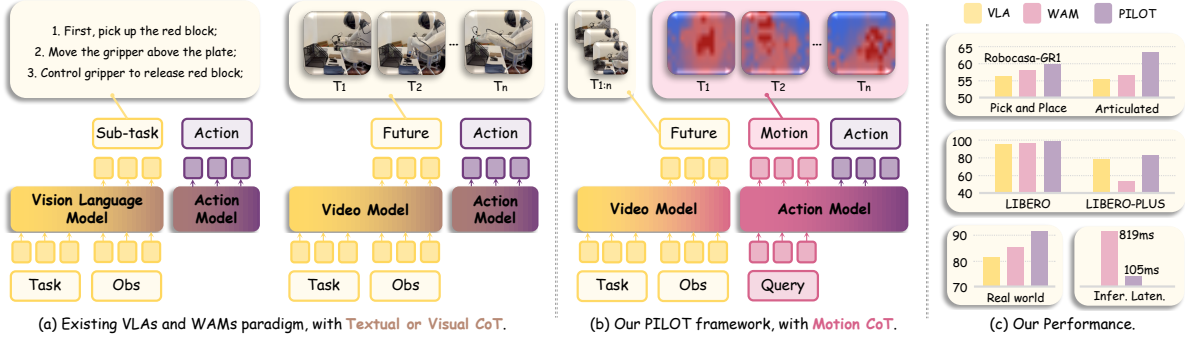


Figure 1. We propose **PILOT**, a novel framework that decouples high-level motion intention from low-level trajectory generation. A Representational Deduction (RD) mechanism is introduced, where latent state transition information is compressed into learnable motion-semantic tokens serving as Motion CoT, enabling the Action Model to focus on fine-grained trajectory refinement while alleviating representational entanglement.

Abstract

World Action Models (WAMs) aim to construct a unified architecture capable of understanding world state evolution and guiding to generative motion planning. However, existing visual branches focus on predicting static visual observation, rather than reflecting potential transition information that captures the evolution of world states under motion interactions. This leads to representational entanglement between high-level physical condition evolution and low-level action trajectory generation within the Action Model, creating a structural bottleneck while weakening the predictive capability of world evolution modeling for action generation. We propose **PILOT** (Physical Inference for Latent Optimized Trajectories), whose core Representational Deduction (RD) bridges this gap by integrating motion thought-of-chain (CoT) guidance as a native model capability. Specifically, RD aims to encourage the action branch to explicitly model potential state transition tokens, which are retained as CoT in the reasoning space to guide fine-grained motion trajectory. Experiments demonstrate that RD not only significantly improves the success rate and generalization ability of WAMs in complex robotic manipulation tasks but also enhances the model’s physical interpretability by decoupling high-level motion semantics from low-level trajectory details. Furthermore, the abundant state transition supervision signals introduced by RD effectively alleviate the sparse supervision in action generation, enabling it to serve as an efficient few-shot real-robot

fine-tuning strategy and demonstrating superior scalability for migration to mainstream WAM architectures.

1. Introduction

World Models [7, 20] and Embodied Intelligence [5, 6, 21, 24, 59, 75] are dedicated to “understanding how the physical world evolves during human interaction” and “planning the action trajectory based on current observations,” respectively. The unification of these paradigms represents a significant step toward General Spatial Intelligence [60]. Recent World Action Models (WAMs) [3, 29, 30, 39, 52] approach this goal by leveraging world models to predict future states and introducing the Inverse Dynamics Module (IDM) [15, 35, 70] to convert these predictions into trajectory. However, future states are confined to the visual latent space constructed by frozen VAEs [9, 45, 69]. This structural bottleneck causes the joint optimization objective of WAMs to primarily target visual reconstruction over trajectory generation. Guiding WAMs to focus on motion semantic representations to eliminate the visual bottleneck via an end-to-end framework remains a long-term goal.

Recent experiments [51, 62, 66] indicate that while explicit future prediction aids training, it provides limited foresight for trajectory generation during inference. We attribute this to the fact that static visual representations

*The corresponding authors are Wenzhong Li (lwz@nju.edu.cn) and Zhihao Yuan (yuanzhihao1@jd.com).

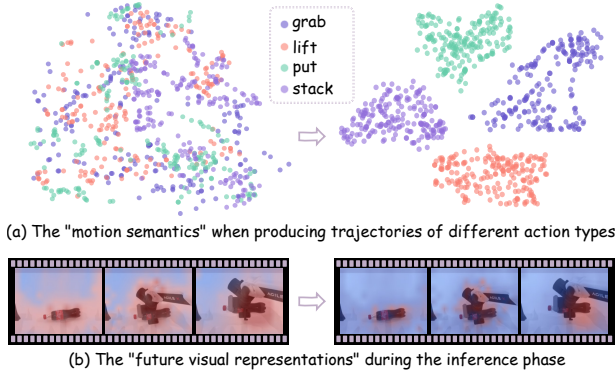


Figure 2. (a) t-SNE of hidden representation. Baselines (left) show entangled clusters due to background coupling, while PILOT(right) forms distinct action-type clusters, isolating motion intent from static details. (b) PCA overlay on future frames. Red highlights indicate high-variance regions. Baselines (left) are distracted by background noise, whereas PILOT (right) focuses on the interaction zone, capturing dynamic state transitions.

at a single moment fail to reflect the evolution process of physical states under action interactions. Without explicit modeling of state transitions, directly generating actions through the diffusion process leads to representational entanglement, which is visualized in Fig. 2. To address this, we propose that action model should simultaneously learn high-level motion intention structures and low-level manipulation details from the same inputs.

How to supervise the model to learn the motion semantic representations that contain state transition information? The existing WAM [10, 18, 19, 65] can precisely predict what will happen in the future, but still doesn't know how to do it. Most relevant works are Latent Action Models (LAMs) [12, 13, 57], which typically capture pixel-level differences and reconstruct future frames from the current frame via IDM [8, 48, 56, 64]. However, this kind of visual-aware-only supervision, fails to prompt the model to understand what action conditions led to the variation in the visual state. Instead of deriving latent motions directly from consecutive frames, we employ them exclusively as supervision, guiding the model to summarize physical state transitions and motion semantics within learnable tokens.

In this paper, we propose Physical Inference for Latent Optimized Trajectories (**PILOT**), an end-to-end framework grounded in the Representational Deduction (**RD**) mechanism that bridges the gap between static visual prediction and dynamic action manipulation by establishing motion semantic CoT reasoning as an innate capability of WAMs. Architecturally, PILOT decouples the World Model into understanding and generation pipelines to substitute explicit future image prediction with the Motion-CoT during inference, thereby accelerating deployment speed while improving both simulation and real-world. Specifically, action model should first distills high-level motion semantics CoT

from current and future predicted observations, where these conditional tokens embody physical causality and state transition information. Subsequently, we freeze the pre-trained VJEPAs-as-Encoder [2, 42, 43] to capture deterministic features tightly coupled with action conditions, and train a Causal Dynamics Engine (**CDE**) to predict future spatial positions, trajectories, shapes, and interaction relationships conditioned on motion semantic CoT, rather than probabilistic underlying visual rendering details. Meanwhile, these motion semantics are retained as context tokens in the reasoning space, allowing the subsequent diffusion process to concentrate on refining fine-grained end-effector trajectories and achieve representational decoupling. This brings an additional benefit: Motion-CoT serves as an extra supervision signal, alleviating the sparse supervision problem faced by the action model due to the low information density of action signals.

Our contributions can be summarized as follows.

- We propose PILOT, a novel WAM framework that decouples high-level motion intention from low-level trajectory generation via a Representational Deduction mechanism, and internalizes latent state transition information as Motion CoT to alleviate representational entanglement.
- We introduce a Causal Dynamics Engine with causally-decoupled attention, that endows the Action Model with explicit physical causality learned from VJEPAs latent spaces, while bypassing the computational bottleneck of pixel-level future frame reconstruction during inference.
- PILOT achieves state-of-the-art performance with 97.9% and 62.6% success rates on LIBERO and RoboCasa-GR1 benchmarks respectively, and attains an 83.1% average success rate in real-world manipulation tasks, demonstrating superior generalization and 90% lower inference latency compared to predict-then-act paradigms.

2. Related Work

2.1. World Models for Robotic Manipulations

Recently, WAMs [9, 10, 29, 30, 69, 71] have emerged as a promising paradigm that leverages world model to predict future, thereby guiding motion manipulation. Existing video-model-based WAMs [26, 55] generally follow two paradigms. The serial-paradigm first predicts future observation and then reconstructs actions via an inverse dynamic [15, 45, 73], while the parallel-paradigm jointly models the visual-action space through shared attention mechanism (MoT) [3, 39, 66] or joint denoising [19, 65]. Unlike generative pixel reconstruction strategies, joint embedding predictive frameworks (JEPAs) [2, 42, 43] mitigate uncertainty in pixel space by actively ignoring entropy-increasing stochastic texture details, thus focusing on predicting dynamic features such as object spatial positions and trajectories. Some studies [23, 49] incorporated JEPAs-as-encoder

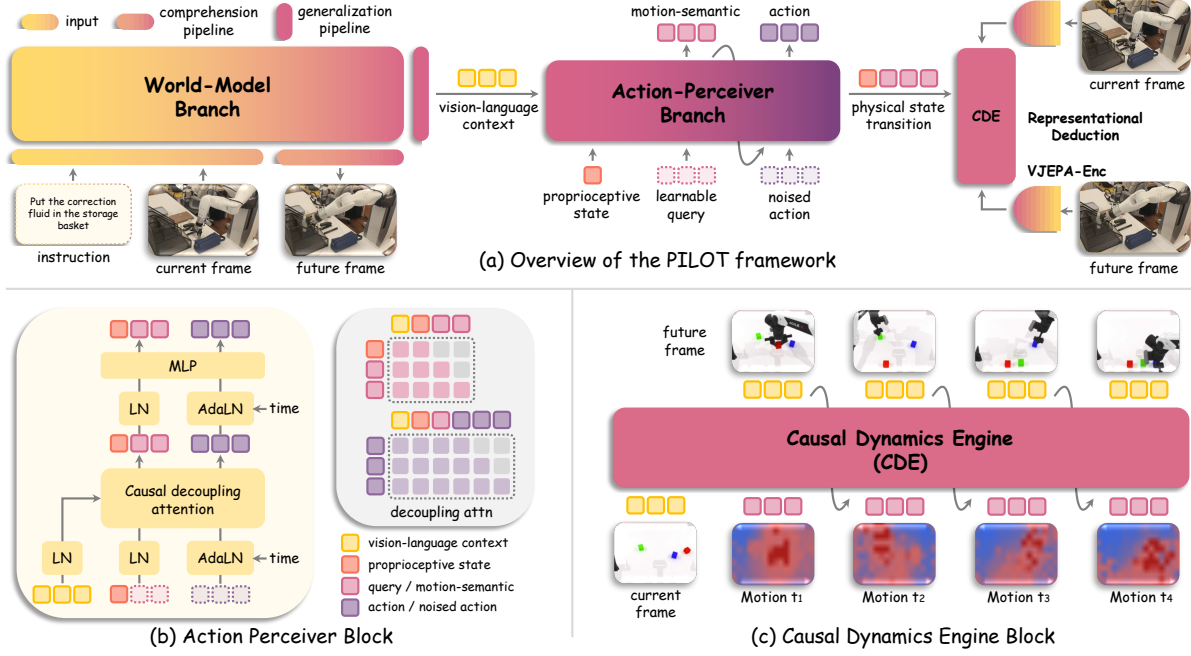


Figure 3. Overview of the **PILOT** framework. In the World-Model (a), current observations and instructions are encoded into a unified context. Subsequently, the Action-Perceiver employs learnable query tokens to distill motion-semantic context via a causally-decoupled attention mechanism (b), which guides the flow-matching action decoder. Simultaneously, we leverage a Causal Dynamics Engine (c) to supervise the motion semantics by learning physical state transition information.

to obtain abundant representations. Most relevant to our work is VLA-JEPA [51], which introduces JEPA to extract latent states, addressing the challenges of pixel reconstruction bias and future information leakage.

However, the purely visually supervision fail to provide state transition information. This inherent discrepancy causes the deviation of co-training from the motion semantics, resulting in fragile transferability and inefficient fine-tuning; conversely, completely removing visual supervision and relying solely on sparse action supervision leads to representation collapse of video model. To address this, we propose Representational Deduction, which bridges the aforementioned gap by incorporating additional motion semantics and physical state transition information.

2.2. Latent Action Learning for Robotics

LAMs [7, 35, 48, 56] aim to pre-train backbones on large-scale human videos lacking action labels, thereby establishing prior knowledge of latent motion. Existing approaches [8, 15, 61, 64, 70] typically follow a two-stage pipeline: (1) encoding visual observations from adjacent frames into discrete latent tokens via a Q-former [28], and (2) aligning vision and manipulation representations using a unified codebook [11, 15, 41, 68] to integrate robotic manipulation into the Next-Token-Prediction framework.

Despite the promise of LAMs, their performance is hindered by noise from camera-induced background variations and future frame leakage, which encourages triv-

ial encoding of future frames rather than learning state transitions [51]. These issues cause structural bottlenecks in learning control-oriented latent actions. Recent works mitigate this via latent constraints [37] or multi-encoder-query [49]. However, the LAMs inherently aligns latent actions with visual space rather than actionable control signals. PILOT fundamentally avoids these defects by ensuring that future visual information is used exclusively as a supervision target, not as a source for capturing latent motion representations, thereby guaranteeing that the model focuses on dynamic representations independent of camera motion and background changes.

3. Method

As illustrated in Fig. 3, PILOT consists of three cooperating components: (a) a World-Model branch built on a pre-trained Wan2.2 [55], which encodes the current observation and instruction into vision-language context and provides auxiliary future-frame supervision (Sec. 3.5); (b) a from-scratch trained Action Model [22] branch that distills motion-semantic context with learnable query tokens and decodes actions through causally-decoupled flow matching (Sec. 3.3); (c) a RD branch that supervises the motion semantics by predicting VJEP2-AC [2] future-state representations with a trainable CDE (Sec. 3.4).

3.1. Problem formulation and notation.

Let each episode provide, a current RGB observation o_t , a future RGB observation $o_{t+\Delta}$, a natural-language instruction $\ell \in \mathbb{R}^L$, a proprioceptive state $s_t \in \mathbb{R}^{d_s}$, and a target action chunk $\mathbf{a}_t = (a_t, \dots, a_{t+H-1}) \in \mathbb{R}^{H \times d_a}$, where H is the future action horizon. The pair $(o_t, o_{t+\Delta})$ is the only extra supervision that RD requires and is obtained from teleoperation trajectories. Throughout, d_s denotes the state dimension and d_a the action dimension. We model instruction-following manipulation as a goal-conditioned controlled Markov process. The target policy factorizes as:

$$p_{\Theta}(\mathbf{a}_t \mid o_t, \ell, s_t) = \int p_{\Theta}(\mathbf{a}_t \mid \mathbf{m}, s_t) \times p_{\Theta}(\mathbf{m} \mid o_t, \ell, s_t) d\mathbf{m}, \quad (1)$$

where $\mathbf{m} \in \mathbb{R}^{K \times d}$ is a latent *motion-semantic* variable that we introduce explicitly. Our central thesis is that the quality of a world-action model hinges on whether $\mathbf{m}$ captures the *action-conditioned state transition* $o_t \rightarrow o_{t+\Delta}$.

3.2. Motivation of Representation Entanglement

Prior WAMs typically inject future information by reconstructing future observations in a frozen VAE latent space. Although this supervision helps the video backbone acquire physical priors, it biases joint optimization toward visual reconstruction rather than manipulation. Moreover, single future visual latent describes what the scene may look like, but does not explicitly model how the world state changes under action. Consequently, the action diffusion process must infer high-level state evolution and generate fine-grained motor commands from the same entangled context, yielding an unstable transition prior at inference.

PILOT addresses this issue through a hierarchy centered on the motion-semantic variable $\mathbf{m}$ (Eq. 1). Causally-decoupled attention (Eq. 3) prevents learnable tokens from absorbing action noise, encouraging $\mathbf{m}$ to encode the state transition $o_t \rightarrow o_{t+\Delta}$. Asymmetric information flow (Eq. 4) then allows $\mathbf{m}$ to condition action diffusion while blocking reverse noise leakage, enabling model (Eq. 7) to focus on trajectory refinement. The resulting representational deduction loss $\mathcal{L}_{\text{RD}}$ (Eq. 11) further provides transition-level supervision, alleviating the sparsity of action-only training.

3.3. Motion-CoT from Learnable Queries

Vision-language context. In the comprehension pipeline, we repurpose the Wan2.2 as a vision-language encoder. The video tokenizer encodes observation o_t into clean latents and the text encoder embeds instruction ℓ ; a single conditional forward pass yields the context sequence

$$\mathbf{c} = \text{WM}_{\theta}(o_t, \ell) \in \mathbb{R}^{N_c \times d_c}, \quad (2)$$

where WM_{θ} denotes the trainable world-model parameters, and N_c and d_c are the number and dimension of context

tokens. The pre-trained video prior in WM_{θ} provides visual and physical cues for downstream action generation.

Learnable motion-semantic queries. We introduce K learnable query tokens $\mathbf{Q} \in \mathbb{R}^{K \times d}$ to distill motion-semantic representations from $\mathbf{c}$ before action decoding. The action model is instantiated as a Perceiver-style flow-matching head. At a sampled flow time τ , the noised action chunk $\mathbf{a}_t^{\tau} = (1 - \tau)\epsilon + \tau\mathbf{a}_t$ is embedded into action tokens $\mathbf{A}^{\tau} \in \mathbb{R}^{H \times d}$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The state s_t is mapped into a state token $\mathbf{S} \in \mathbb{R}^{1 \times d}$, and the latent sequence $[\mathbf{S}; \mathbf{Q}; \mathbf{A}^{\tau}]$ attends to $\mathbf{c}$ through a stack of L blocks.

Causally-decoupled attention mechanism. To prevent motion semantics from absorbing diffusion noise, we decouple the attention inside each Perceiver block. Denoting the non-action latents as $\mathbf{Z} = [\mathbf{S}; \mathbf{Q}]$, each block applies

$$\mathbf{Z} \leftarrow \mathbf{Z} + \text{Attn}(\mathbf{Z}, [\mathbf{c}; \mathbf{Z}]), \quad (3)$$

$$\mathbf{A}^{\tau} \leftarrow \mathbf{A}^{\tau} + \text{Attn}(\mathbf{A}^{\tau}, [\mathbf{c}; \mathbf{Z}; \mathbf{A}^{\tau}]), \quad (4)$$

where $\mathbf{Z}$ attends only to itself and the vision-language context, while action tokens additionally attend to $\mathbf{Z}$. This asymmetric flow lets motion semantics condition action decoding while blocking reverse noise leakage. Accordingly, time-conditioned AdaLN is applied only to action tokens, and $\mathbf{Z}$ uses plain layer normalization to remain invariant to the flow time τ . After the final block, we read out the query slots as the motion-semantic context:

$$\mathbf{O} = \text{Act.Per.}(\mathbf{Z}, \mathbf{A}^{\tau}, \mathbf{c}; \tau) \in \mathbb{R}^{(1+K+H) \times d}, \quad (5)$$

$$\mathbf{m} = \mathbf{O}[:, 1:K+1, :] \in \mathbb{R}^{K \times d}. \quad (6)$$

Flow-matching action decoding. Action tokens after the last block are projected back to the action space and supervised with the conditional flow-matching velocity objective:

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{\tau, \epsilon} \|\text{Proj}_{\text{act}}(\mathbf{O}[:, K+1:, :]) - (\mathbf{a}_t - \epsilon)\|^2, \quad (7)$$

where Proj_{act} is a MLP mapping action tokens to the action dimension d_a . At inference, Euler integration starts from Gaussian noise, conditioned on the query-produced $\mathbf{m}$.

3.4. Representational Deduction with a CDE

The motion-semantic context $\mathbf{m}$ becomes meaningful only when it is explicitly constrained to encode *what changes in the world state*. We provide such supervision via CDE that models state transitions in the representation space closely associated with physical dynamics.

Frozen state representations. Given the current and future observations $(o_t, o_{t+\Delta})$, the frozen VJEP2-AC encoder E_{ψ} produces per-frame patch representations:

$$\mathbf{r}_t = E_{\psi}(o_t), \quad \mathbf{r}_{t+\Delta} = E_{\psi}(o_{t+\Delta}), \quad (8)$$

where each representation lies in $\mathbb{R}^{P \times d_r}$, with P patch tokens and token dimension d_r . Unlike the appearance-oriented visual latents, VJEPA2-AC representations are learned through a joint-embedding predictive objective and are therefore organized around predictable, action-relevant structure rather than pixel-level reconstruction. This enables RD to supervise motion rather than texture.

Causal dynamics engine. The CDE G_ξ is initialized from the pre-trained VJEPA2-AC predictor and remains trainable. Instead of using the original action and state inputs of the predictor, we condition the CDE on the learned motion semantics and robotic state. Two lightweight linear projectors map them into the VJEPA2-AC latent space:

$$\tilde{\mathbf{m}} = \mathbf{W}_m \mathbf{m} \in \mathbb{R}^{K \times d_r}, \quad \tilde{\mathbf{s}} = \mathbf{W}_s \mathbf{s}_t \in \mathbb{R}^{1 \times d_r}. \quad (9)$$

Conditioned on $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{s}}$, the CDE predicts the future representation from the current one:

$$\hat{\mathbf{r}}_{t+\Delta} = G_\xi(\mathbf{r}_t; \tilde{\mathbf{m}}, \tilde{\mathbf{s}}) \in \mathbb{R}^{P \times d_r}. \quad (10)$$

Internally, $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{s}}$ are prepended to the current-frame patch tokens as conditioning tokens, and only the predicted frame tokens are returned.

representational deduction loss. We supervise the CDE by matching its prediction to the frozen encoder representation of the ground-truth future frame:

$$\mathcal{L}_{\text{RD}} = \text{SmoothL1}(\hat{\mathbf{r}}_{t+\Delta}, \mathbf{r}_{t+\Delta}). \quad (11)$$

Gradients from $\mathcal{L}_{\text{RD}}$ propagate through the CDE and the projector $\mathbf{W}_m$ to the learnable queries, encouraging $\mathbf{m}$ to encode transition information that is predictive of future VJEPA2-AC representations. Because the supervision target is representational rather than pixel-level, RD teaches the model *how the state evolves under the action*, rather than merely *what the future looks like*.

3.5. World-Model Branch and Training Objective

Future-frame latent prediction. In the generation pipeline, the world model retains its generative capability through an auxiliary future-frame objective. We encode the short clip $[o_t, o_{t+\Delta}]$ into clean video latents, using the current-frame latents as conditions and the future-frame latents as prediction targets. Flow-matching noise is applied only to the target latents, and the world model is trained to predict the corresponding target velocity:

$$\mathcal{L}_{\text{wm}} = \mathbb{E}_{\tau, \epsilon} \left\| \text{WM}_\theta(\mathbf{z}^\tau; o_t, \ell, \tau) - (\epsilon - \mathbf{z}^*) \right\|_{\odot \mathbf{M}_{\mathbf{z}^\tau}}^2, \quad (12)$$

where WM_θ denotes the trainable world-model parameters, $\mathbf{z}^*$ are the clean target latents, $\mathbf{z}^\tau$ is their noised version,

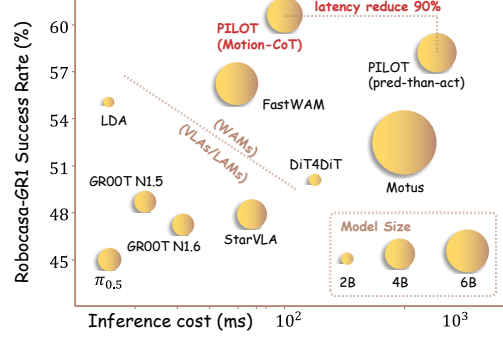


Figure 4. Inference latency of existing VLAs and WAMs on H20 GPU, along with success rates on RoboCasa-GRI. We further compare the inference overhead of PILOT under the “Motion-CoT” and “Predict-then-Act” paradigms with identical architectures. Motion-CoT reduces latency by 90%.

and $\mathbf{M}_{\mathbf{z}^\tau}$ restricts the loss to future-frame positions. This objective preserves the instruction-grounded future prior of the pre-trained generator, while the generation pipeline is disabled during inference to avoid the visual bottleneck.

Overall objective. The three branches are trained jointly with a single weighted objective:

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda_{\text{wm}} \mathcal{L}_{\text{wm}} + \lambda_{\text{RD}} \mathcal{L}_{\text{RD}}, \quad (13)$$

where λ_{wm} and λ_{RD} balance the auxiliary future-frame and Representational Deduction losses against the action objective. In practice, the VJEPA2-AC encoder is frozen; the world model and CDE are fine-tuned from their pre-trained weights; and the Action Model together with the two projectors is trained from scratch.

Inference. At test time, only the understanding–action pathway is used: the world model encodes (o_t, ℓ) into context $\mathbf{c}$, the learnable queries produce the motion-semantic context $\mathbf{m}$, and the flow-matching head outputs the action chunk conditioned on $\mathbf{m}$. The RD branch and future-frame prediction serve only as training-time supervision and incur no additional inference cost; the generative decoder can be optionally invoked for future-frame visualization.

4. Experiments

4.1. Implementation Details

Our codebase is built upon the StarVLA project [63]. PILOT is implemented with three cooperative branches. The **World-Model branch** builds on a pre-trained Wan2.2 video diffusion transformer, which simultaneously serves as a visual-language encoder (providing a 196×2048 context) and a trainable latent future-frame generator. The **Action-Model branch** adopts learnable query tokens ($K = 64$,

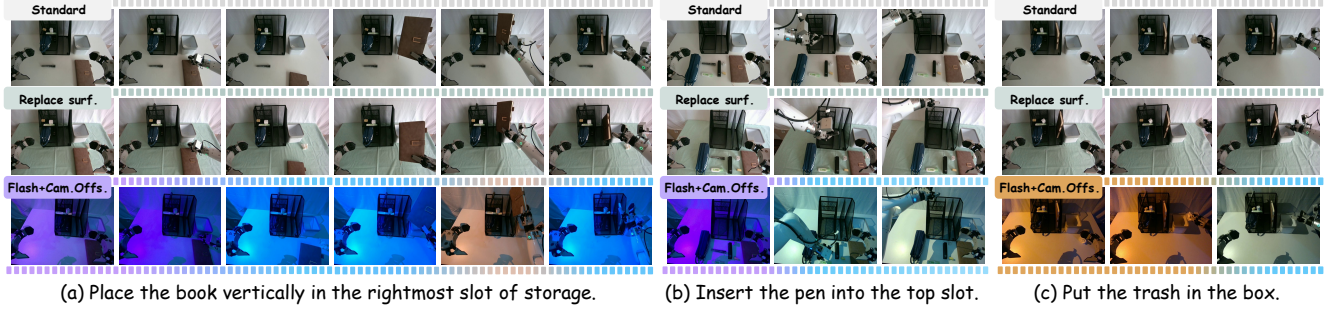


Figure 5. Real-world evaluation suite on the Agibot-G1 humanoid robot.

$d = 1024$) that distill motion-semantic information, followed by a DiT-based flow-matching action decoder with 4 denoising steps. The **Representational Deduction branch** employs a frozen VJEP2-AC encoder that produces current and future patch-level representations (256×1408), and a lightweight Causal Dynamics Engine that predicts future representations conditioned on motion semantics and proprioceptive state. Please refer to the appendix for complete architecture hyper-parameters, training hyper-parameters, and algorithm pseudocode details.

4.2. Simulation Benchmarks

We evaluate PILOT on two simulation benchmarks spanning different embodiments, action spaces, and visual modalities.

LIBERO. As shown in Table 1, PILOT achieves **97.9%** average success rate with only 5.4B parameters, surpassing all baselines including Motus (8B, 97.7%) and $\pi_{0.5}$ (3.5B with internet-scale pre-training, 96.2%). Notably, PILOT excels on the challenging Goal (98.1%) and Long-horizon (97.2%) suites, where understanding the *intent* of manipulation is critical for long-horizon planning. This validates our central thesis: decoupling motion semantics from trajectory generation via Representational Deduction effectively alleviates representational entanglement.

LIBERO-Plus. As reported in Table 2, PILOT achieves **81.0%** total success rate, surpassing the previous best PokeVLA (79.3%) by 1.7% absolute. The improvement is particularly pronounced on Robot (70.0% vs. 46.1%) and Layout (82.0% vs. 77.2%), suggesting that the motion-semantic tokens learned by Representational Deduction encode *transition dynamics* rather than superficial appearance, making the policy inherently more robust to structural perturbations.

RoboCasa-GR1. As shown in Table 3, PILOT achieves **62.6%** average success rate, outperforming all baselines including FastWAM (56.7%, 6B) and LDA (55.4%, 7B). The performance gap is largest on PnP Novel From Tray (74.8%

Method	Size	Spatial	Object	Goal	Long	Avg.
<i>Vision-Language-Action Models</i>						
$\pi_{0.5}$ [21]	3.5B	98.8	98.2	98.0	92.4	96.2
GROOT-N1.6 [44]	3.3B	97.7	98.5	97.5	94.4	97.0
OpenVLA-OFT [25]	7B	97.6	98.4	97.9	94.5	97.1
<i>Latent Action Models</i>						
LAPA [64]	7B	73.8	74.6	58.8	55.4	65.7
UniVLA [8]	7B	96.5	96.8	95.6	92.0	95.2
Mantis [62]	5.8B	98.8	99.2	94.4	94.2	96.7
VLA-JEPA [51]	3B	96.2	99.6	97.2	95.8	97.2
<i>World-Action Models</i>						
F1 [38]	4B	98.2	97.8	95.4	91.3	95.7
Fast-WAM [66]	6B	98.2	100.0	97.0	95.2	97.6
Motus [3]	8B	96.8	99.8	96.6	97.6	97.7
PILOT (Ours)	5.4B	97.1	99.2	98.1	97.2	97.9

Table 1. **Evaluation on LIBERO [34].** Success rates (%) are reported for four task suites and the average.

Method	Size	Cam.	Rob.	Lang.	Light	Back.	Noise	Layout	Avg.
<i>Vision-Language-Action Models</i>									
ECot [67]	3B	0.3	26.8	40.2	42.6	16.4	10.2	36.9	24.3
Spatial Forcing [27]	3.3B	20.1	13.4	40.9	29.1	33.4	25.7	39.3	29.1
π_0 -FAST [46]	3.3B	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
OpenVLA-OFT [24]	7B	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
<i>Latent Action Models</i>									
UniVLA [8]	7B	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
PokeVLA [72]	7B	84.7	46.1	84.8	94.6	82.6	89.8	77.2	79.3
<i>World-Action Models</i>									
WorldVLA [10]	7B	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
Fast-WAM [66]	6B	16.4	44.5	68.9	78.2	53.7	37.7	60.7	51.5
PILOT (Ours)	5.4B	76.0	70.0	84.0	92.0	85.0	78.0	82.0	81.0

Table 2. **Evaluation on LIBERO-Plus [17].** Success rates (%) on seven perturbation types with zero-shot from standard LIBERO.

vs. 55.1%) and PnP Close (59.7% vs. 56.3%), which require precise spatial reasoning under novel object configurations. We attribute this to the Motion-CoT mechanism: by distilling transition dynamics into dedicated semantic tokens, PILOT implicitly captures spatial generalization that guides the action decoder through unseen arrangements. Full per-task results are provided in the Appendix.

Task Category	GR00T N1.5	GR00T N1.6	DiT4DiT	UWM-XL	StarVLA	LDA	VP-VLA	PhysBrain	LangForce	FastWAM	Motus	PILOT
Pick and Place 6 Tasks	45.3	24.2	50.3	30.7	50.3	56.3	54.3	52.7	55.7	52.7	49.3	59.7
Articulated from Cuttingboard 5 Tasks	46.4	56.9	57.6	17.4	52.8	64.2	60.8	52.0	53.2	60.6	56.6	61.2
Articulated from Placemat 4 Tasks	45.5	51.9	39.0	10.0	38.0	47.8	54.5	48.0	48.0	47.2	43.8	56.0
Articulated from Tray 5 Tasks	48.8	55.1	46.4	17.2	39.2	53.4	46.0	46.0	47.2	54.8	50.8	74.8
Articulated from Plate 4 Tasks	56.5	57.5	60.0	16.2	58.5	53.0	53.5	62.5	58.5	58.5	54.5	60.5
Average	48.2	47.6	50.8	19.2	47.8	55.4	53.8	50.0	52.6	56.7	53.1	62.6

Table 3. **Evaluation on RoboCasa-GR1 [4].** Success rates (%) across five task categories and the overall average on 24 manipulation tasks with the GR1 humanoid. Full per-task results are provided in the Appendix.

4.3. Real-World Evaluation

We deploy PILOT on an Agibot-G1 dual-arm humanoid robot to assess real-world manipulation capabilities, as shown in Fig.5. The evaluation comprises three settings:

Standard Tasks. We define 8 single-step pick-and-place tasks on a clean tabletop: picking up a *pen*, *eraser*, *correction fluid*, *charger*, *pencil case*, *stapler*, *book*, and *trash* item, each evaluated over 50 rollouts. As shown in Table 4, PILOT achieves **83.1%** average success rate, surpassing Fast-WAM (73.3%) and $\pi_{0.5}$ (71.3%). On the most challenging task (*pencil case*, 55%), PILOT still outperforms Fast-WAM by 11% absolute, confirming that motion-semantic tokens effectively handle spatially complex grasps.

Generalization Transfer. We further evaluate on 3 transfer scenarios that perturb the training distribution: **Flash+Cam.Offs.** (apply a strobe-light background and offset the robot’s head-mounted camera angle), **Desk** (replaced table surface), and **Color** (altered object appearance/color). PILOT achieves **68.3%** generalization average, compared to 50.0% for Fast-WAM and 48.0% for $\pi_{0.5}$. The 18.3% gap over Fast-WAM indicates that the Representational Deduction branch encodes *transition dynamics* rather than superficial visual appearance, making the policy inherently robust to distributional shifts.

Few-Shot Fine-Tuning. Finally, we fine-tune with only 10% of the full training data. PILOT retains **62.4%** average success rate, a modest 20.7% relative drop from the full-data 83.1%, whereas Fast-WAM drops from 73.3% to 40.8% (44.3% relative decline). This demonstrates that the frozen VJEPa2-AC encoder provides general-purpose physical representations that transfer across embodiments, and the Causal Dynamics Engine needs only to adapt its transition predictions to the new dynamics. This makes PILOT particularly suitable for deployment scenarios where collecting extensive real-world demonstrations is prohibitively expensive.

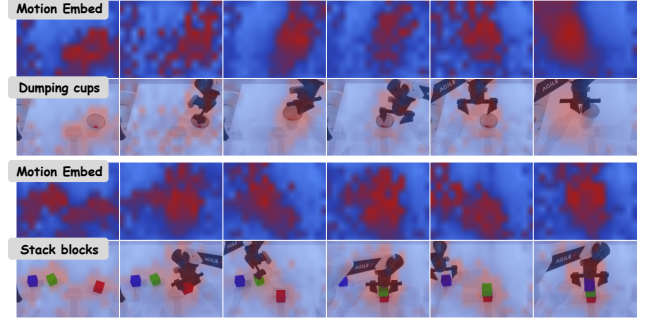


Figure 6. PCA RGB pseudo-color visualization of Causal Dynamics Engine predicted future representations.

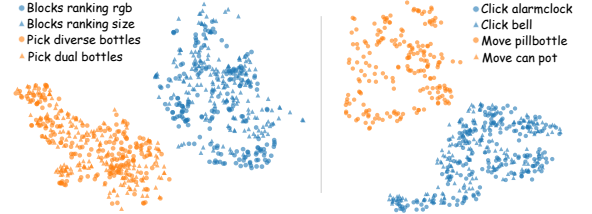


Figure 7. t-SNE visualization of Motion-CoT token embeddings across some RoboCasa-GR1 tasks, colored by task category. The two colors denote two action types, each containing two sub-tasks represented by triangles and circles.

4.4. Ablation Studies

We conduct cumulative ablation experiments that incrementally add components to a minimal baseline, isolating each contribution. Results are reported on LIBERO and RoboCasa-GR1.

Cumulative component ablation. Table 5 shows the results of progressively adding components. Starting from a bare backbone without future prediction or Motion-CoT adding future-frame prediction improves LIBERO by +2.3% and RoboCasa by +3.6%, confirming that the change-weighted future-frame loss provides useful visual grounding. Replacing future-frame prediction with Motion-CoT alone yields a larger gain (+3.2% / +5.5% over the baseline), indicating that distilled semantic tokens are a more effective information bottleneck than pixel-level generation. Adding the Representational Deduction branch with Causal Dynamics Engine supervision further boosts

Method	Size	Standard Tasks								Stand. Avg.	Generalization				Gen. Avg.	Few-Shot Avg. (10%)
		Pen	Eraser	Corr. Fluid	Charger	Pencil Case	Stapler	Book	Trash		Flash+Cam.Offs.	Desk	Color			
π_0 [5]	3.3B	70	82	46	78	32	62	52	72	61.8	38	42	36	38.7	28.4	
OpenVLA [24]	7B	64	76	40	72	26	56	46	66	55.8	32	36	30	32.7	24.6	
GR00T-N1 [4]	3.4B	74	86	50	82	36	66	56	76	65.8	42	46	40	42.7	32.8	
$\pi_{0.5}$ [21]	3.3B	80	90	56	86	42	72	62	82	71.3	48	52	44	48.0	38.2	
Cosmos-Policy [26]	6B	76	88	52	84	38	68	58	78	67.8	44	48	42	44.7	34.6	
Fast-WAM [66]	6B	82	92	58	88	44	74	64	84	73.3	50	54	46	50.0	40.8	
PILOT (Ours)	5.4B	90	100	70	94	56	88	72	94	83.1	72	68	64	68.3	62.4	

Table 4. **Real-world evaluation on Agibot-G1.** Success rates (%) across 8 standard tasks, 3 generalization transfer tasks, and few-shot fine-tuning (10% data). Best results are in **bold**.

Fut. Pred	MotionCoT	RD	CDE	Decoup.	LIBERO Avg.	RoboCasa Avg.
×	×	—	—	—	91.3	51.2
✓	×	—	—	—	93.6	54.8
×	✓	×	×	×	94.5	56.7
×	✓	✓	✓	×	96.1	59.8
×	✓	✓	✓	✓	96.9	61.3
✓	✓	✓	✓	✓	97.9	62.6

Table 5. **Cumulative component ablation on LIBERO and RoboCasa.** Incrementally adding components to baseline.

both benchmarks (+1.6% / +3.1%), validating that explicit dynamics prediction regularizes motion-semantic tokens toward physically meaningful transitions. Enabling causally-decoupled attention adds another +0.8% / +1.5%, confirming that unconstrained cross-attention introduces noise from irrelevant visual regions. Finally, combining future-frame prediction with the full Motion-CoT pipeline yields the best results, demonstrating that the two complementary objectives-pixel-level visual grounding and semantic-level transition prediction-jointly achieve optimal performance.

4.5. Representational Analysis

We project mean-pooled Motion-CoT embeddings via t-SNE[54] (Figure 6) and map 1408-dimensional Causal Dynamics Engine predictions to RGB via PCA (Figure 7). The t-SNE clustering reveals that Motion-CoT tokens naturally group by action type *without any explicit clustering supervision*, indicating that the Causal Dynamics Engine future-prediction objective implicitly distills task-relevant transition semantics from the entangled visual stream. Notably, embeddings from the same action type but different sub-tasks (picking from different locations) remain closely clustered, while different action types are well separated, suggesting that PILOT decouples motion semantics from background and object appearance variations. PCA visualization confirms this: CDE predictions yield spatially coherent structures matching ground-truth future states, capturing region-aware transitions. Importantly, only the foreground regions undergoing manipulation exhibit significant color changes, while static background regions remain largely unchanged, this provides direct evidence that the Representational Deduction branch successfully decouples action-conditioned dynamics from irrelevant background information, enabling model to focus on task-relevant vision.

5. Conclusion

We present PILOT, a framework that decouples high-level motion semantics from low-level trajectory generation via Representational Deduction. By compressing state transitions into Motion-CoT, PILOT mitigates representational entanglement in World Action Models, achieving real-time efficiency and strong few-shot transferability. Future work will extend RD to long-horizon planning by incorporating hierarchical Motion-CoT for multi-stage embodied tasks.

References

- [1] Ali Abouzeid, Malak Mansour, Qinbo Sun, Zezhou Sun, and Dezhen Song. Geoaware-vla: Implicit geometry aware vision-language-action model. *arXiv:2509.14117*, 2025. [12](#)
- [2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv:2506.09985*, 2025. [2](#), [3](#), [12](#)
- [3] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *CVPR*, pages 35101–35113, 2026. [1](#), [2](#), [6](#), [12](#), [20](#)
- [4] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv:2503.14734*, 2025. [7](#), [8](#), [12](#), [20](#), [24](#)
- [5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi-0: A vision-language-action flow model for general robot control. *arXiv:2410.24164*, 2024. [1](#), [8](#), [12](#), [13](#), [20](#)
- [6] 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:2212.06817*, 2022. [1](#), [12](#)
- [7] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *ICML*, 2024. [1](#), [3](#), [12](#)
- [8] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv:2505.06111*, 2025. [2](#), [3](#), [6](#), [12](#), [13](#), [20](#)
- [9] Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv:2511.17502*, 2025. [1](#), [2](#)
- [10] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv:2506.21539*, 2025. [2](#), [6](#), [12](#), [20](#)
- [11] Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, et al. Lawam: Latent world action models for efficient dynamics-aware robot policies. *arXiv:2606.15768*, 2026. [3](#), [13](#)
- [12] Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, Chenfeng Xu, Quan Vuong, and Ken Goldberg. Mirage: Cross-embodiment zero-shot policy transfer with cross-painting. *arXiv:2402.19249*, 2024. [2](#)
- [13] Xi Chen, Ali Ghadirzadeh, Tianhe Yu, Jianhao Wang, Alex Yuan Gao, Wenzhe Li, Liang Bin, Chelsea Finn, and Chongjie Zhang. Lapo: Latent-variable advantage-weighted policy optimization for offline reinforcement learning. *NeurIPS*, 35:36902–36913, 2022. [2](#), [12](#)
- [14] Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, et al. Villa-x: enhancing latent action modeling in vision-language-action models. *arXiv:2507.23682*, 2025. [13](#)
- [15] Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19752–19763, 2025. [1](#), [2](#), [3](#), [13](#)
- [16] Shichao Fan, Kun Wu, Zhengping Che, Xinhua Wang, Di Wu, Fei Liao, Ning Liu, Yixue Zhang, Zhen Zhao, Zhiyuan Xu, et al. Xr-1: Towards versatile vision-language-action models via learning unified vision-motion representations. *arXiv:2511.02776*, 2025. [13](#)
- [17] 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:2510.13626*, 2025. [6](#)
- [18] Yao Feng, Hengkai Tan, Xinyi Mao, Chendong Xiang, Guodong Liu, Shuhe Huang, Hang Su, and Jun Zhu. Vidar: Embodied video diffusion model for generalist manipulation. *arXiv:2507.12898*, 2025. [2](#)
- [19] Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. *arXiv:2602.06949*, 2026. [2](#)
- [20] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv:2301.04104*, 2023. [1](#)
- [21] 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:2504.16054*, 2025. [1](#), [6](#), [8](#), [12](#), [13](#), [20](#)
- [22] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *ICML*, pages 4651–4664, 2021. [3](#)
- [23] Saurav Jha, Artem Zholus, Sarath Chandar, et al. Reconstruction or semantics? what makes a latent space useful for robotic world models. *arXiv:2605.06388*, 2026. [2](#), [12](#)
- [24] 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:2406.09246*, 2024. [1](#), [6](#), [8](#), [12](#), [20](#)
- [25] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv:2502.19645*, 2025. [6](#), [12](#), [13](#), [20](#)
- [26] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-

- Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv:2601.16163*, 2026. 2, 8, 12, 20
- [27] Fuhao Li, Wenxuan Song, Han Zhao, et al. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv:2510.12276*, 2025. 6, 20
- [28] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023. 3, 13
- [29] Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv:2601.21998*, 2026. 1, 2, 12
- [30] Shalfun Li, Victor Yao, Charles Yang, Truth Qu, Regis Cheng, Ryan Yu, Howard Lu, Newton Von, Vincent Chen, Yohann Tang, et al. Wall-wm: Carving world action modeling at the event joints. *arXiv:2606.01955*, 2026. 1, 2
- [31] Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action models via instruction-driven routing & sparsification. *Advances in neural information processing systems*, 38:137646–137675, 2026. 12
- [32] Shijie Lian, Bin Yu, Laurence Tianruo Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, Kai Chen, et al. Langforce: Bayesian decomposition of vision language action models via latent action queries. In *ICML*, 2026. 20
- [33] Xiaopeng Lin, Shijie Lian, Bin Yu, Ruoqi Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Yurun Jin, Yukun Shi, Jiyan He, et al. Physbrain: Human egocentric data as a bridge from vision language models to physical intelligence. *arXiv:2512.16793*, 2025. 20
- [34] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *NeurIPS*, 36: 44776–44791, 2023. 6, 20
- [35] Mengya Liu, Baoxiong Jia, Jiangyong Huang, Jingze Zhang, and Siyuan Huang. Lara: Latent action representation alignment for vision-language-action models. *arXiv:2606.07100*, 2026. 1, 3, 12
- [36] Songming Liu, Lingxuan Wu, Bangguo Li, et al. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *ICLR*, pages 29982–30009, 2025. 12
- [37] Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-h0. 7: A latent world-action model from egocentric videos. *arXiv:2605.00078*, 2026. 3, 13
- [38] Qi Lv, Weijie Kong, Hao Li, et al. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv:2509.06951*, 2025. 6, 20
- [39] 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:2602.12215*, 2026. 1, 2, 12, 20
- [40] Teli Ma, Jia Zheng, Zifan Wang, et al. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv:2603.10448*, 2026. 20
- [41] Xiangkai Ma, Lekai Xing, Han Zhang, Wenzhong Li, and Sanglu Lu. Unifying perception and action: A hybrid-modality pipeline with implicit visual chain-of-thought for robotic action generation. In *CVPR*, pages 22380–22390, 2026. 3, 13
- [42] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv:2603.19312*, 2026. 2, 12
- [43] Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-jepa 2.1: Unlocking dense features in video self-supervised learning. *arXiv:2603.14482*, 2026. 2, 12
- [44] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Chnriadev, et al. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. 6, 20
- [45] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv:2512.15692*, 2025. 1, 2, 12
- [46] Karl Pertsch, Kyle Stachowicz, Brian Ichter, et al. Fast: Efficient action tokenization for vision-language-action models. *arXiv:2501.09747*, 2025. 6, 20
- [47] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv:2501.15830*, 2025. 12
- [48] Sandeep Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. Vipra: Video prediction for robot actions. *arXiv:2511.07732*, 2025. 2, 3, 12
- [49] Yue Su, Sijin Chen, Haixin Shi, Mingyu Liu, Zhengshen Zhang, Ningyuan Huang, Weiheng Zhong, Zhengbang Zhu, Yuxiao Liu, and Xihui Liu. World guidance: World modeling in condition space for action generation. *arXiv:2602.22010*, 2026. 2, 3, 12, 13
- [50] Guoheng Sun, Tingting Du, Kaixi Feng, Chenxiang Luo, Xingguo Ding, Zheyu Shen, Ziyao Wang, Yexiao He, and Ang Li. Rocket: Residual-oriented multi-layer alignment for spatially-aware vision-language-action models. *arXiv:2602.17951*, 2026. 12
- [51] 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. *arXiv:2602.10098*, 2026. 1, 3, 6, 12, 13, 20
- [52] MotuBrain Team, Chendong Xiang, Fan Bao, Haitian Liu, Hengkai Tan, Hongzhe Bi, James Li, Jiabao Liu, Jingrui Pang, Kiro Jing, et al. Motubrain: An advanced world action model for robot control. *arXiv:2604.27792*, 2026. 1
- [53] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv:2302.00482*, 2023. 13, 18

- [54] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008. 8
- [55] 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:2503.20314*, 2025. 2, 3
- [56] Chenyu Wang, Cai Zhou, Sharut Gupta, Johnson Lin, Stefanie Jegelka, Stephen Bates, and Tommi Jaakkola. Learning diffusion models with flexible representation guidance. *NeurIPS*, 38:131176–131222, 2026. 2, 3, 12
- [57] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Advances in neural information processing systems*, 37:124420–124450, 2024. 2, 12
- [58] Zixuan Wang, Yuxin Chen, Yuqi Liu, et al. Vp-vla: Visual prompting as an interface for vision-language-action models. *arXiv:2603.22003*, 2026. 20
- [59] Sheng Xu, Ruixing Jin, Huayi Zhou, Bo Yue, Guanren Qiao, Yunxin Tai, Yueci Deng, Kui Jia, and Guiliang Liu. From reaction to anticipation: Proactive failure recovery through agentic task graph for robotic manipulation. *arXiv:2605.11951*, 2026. 1
- [60] Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *CVPR*, pages 10632–10643, 2025. 1
- [61] Jiange Yang, Yansong Shi, Haoyi Zhu, Mingyu Liu, Kaijing Ma, Yating Wang, Gangshan Wu, Tong He, and Limin Wang. Como: Learning continuous latent motion from internet videos for scalable robot learning. In *CVPR*, pages 42352–42363, 2026. 3, 13
- [62] Yi Yang, Xueqi Li, Yiyang Chen, Jin Song, Yihan Wang, Zipeng Xiao, Jiadi Su, You Qiaoben, Pengfei Liu, and Zhijie Deng. Mantis: A versatile vision-language-action model with disentangled visual foresight. In *CVPR*, pages 42505–42515, 2026. 1, 6, 20
- [63] Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. Starvla: Reducing complexity in vision-language-action systems. *arXiv preprint arXiv:2604.11757*, 2026. 5, 13, 20
- [64] 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 *ICLR*, pages 28213–28239, 2025. 2, 3, 6, 13, 20
- [65] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv:2602.15922*, 2026. 2, 12
- [66] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv:2603.16666*, 2026. 1, 2, 6, 8, 12, 20
- [67] Michał Zawalski, William Chen, Karl Pertsch, et al. Robotic control via embodied chain-of-thought reasoning. *arXiv:2407.08693*, 2024. 6, 20
- [68] Chubin Zhang, Jianan Wang, Zifeng Gao, Yue Su, Tianru Dai, Cai Zhou, Jiwen Lu, and Yansong Tang. Clap: Contrastive latent action pretraining for learning vision-language-action models from human videos. *arXiv:2601.04061*, 2026. 3, 13
- [69] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *NeurIPS*, 38:24195–24228, 2026. 1, 2
- [70] Wenyao Zhang, Bozhou Zhang, Zekun Qi, Wenjun Zeng, Xin Jin, and Li Zhang. Disentangled robot learning via separate forward and inverse dynamics pretraining. *arXiv preprint arXiv:2604.16391*, 2026. 1, 3, 13
- [71] Yuyang Zhang, Wenyao Zhang, Zekun Qi, et al. Imagewam: Do world action models really need video generation, or just image editing? *arXiv:2606.19531*, 2026. 2, 19
- [72] Yupeng Zheng, Xiang Li, Songen Gu, Yuhang Zheng, Shuai Tian, Weize Li, Linbo Wang, Senyu Fei, Pengfei Li, Yinfeng Gao, et al. Pokevla: Empowering pocket-sized vision-language-action model with comprehensive world knowledge guidance. *arXiv:2604.20834*, 2026. 6, 20
- [73] Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. Robodreamer: Learning compositional world models for robot imagination. *arXiv:2404.12377*, 2024. 2
- [74] Chuning Zhu, Raymond Yu, Siyuan Feng, et al. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv:2504.02792*, 2025. 20
- [75] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023. 1, 12

6. Extended Related Work

We provide a more comprehensive discussion of the research landscape surrounding PILOT, extending the brief treatment in manuscripts.

6.1. Vision-Language-Action Models

Vision-Language-Action (VLA) models [4–6, 21, 24] represent a paradigm that adapts pretrained vision-language models (VLMs) into robotic policies through fine-tuning on robot demonstration data. Early VLA approaches [6, 75] autoregressively decode discrete action tokens from a fine-tuned VLM, leveraging the semantic priors inherited from web-scale pretraining to achieve strong generalization across objects, scenes, and language instructions. This paradigm has been extended through several architectural innovations: parallel decoding for improved efficiency [25], flow-matching action experts for continuous action spaces [5, 21], diffusion-based action heads [36], and compact open-source VLMs for broader accessibility.

Recent efforts have scaled VLAs to larger backbones and more diverse embodiments. GR00T [4] and the π family [5, 21] demonstrate that increasing model capacity yields substantial gains in manipulation success rates. Uni-VLA [8] and CogVLA [31] explore unified architectures that bridge multiple task domains. SpatialVLA [47] and Rocket-VLA [50] attempt to inject geometric understanding by aligning intermediate VLA features with geometric foundation models, while GeoAware-VLA [1] directly replaces the vision encoder with a geometric model.

However, a critical limitation of current VLAs stems from their foundational architecture: they rely on representations learned almost exclusively from static image-text pairs. Consequently, the heavy burden of learning low-level physical interactions and temporal state transitions falls entirely on the downstream robotic fine-tuning phase, which requires thousands of hours of training data. PILOT addresses this limitation by incorporating a pre-trained video diffusion model that naturally internalizes complex, continuous physical dynamics through future-frame prediction, while the RD branch further distills transition dynamics into dedicated semantic tokens.

6.2. World Action Models

World Action Models (WAMs) [10, 29] leverage world modeling, predicting future visual states, to support downstream action prediction. We follow [10] to use this terminology, as these models unify world state evolution modeling and action generation within a single framework.

Existing WAMs can be broadly categorized into two paradigms. The *serial paradigm* (predict-then-act) first generates future visual trajectories and then extracts actions from the predicted observations. Early visual foresight methods pioneered this approach by “imagining” future

states for model-based planning. More recently, Mimic-Video [45] pairs a pre-trained video backbone with a separate flow-matching action decoder, conditioning the policy on partially denoised video latents. Cosmos Policy [26] fine-tunes a pre-trained video diffusion model to directly output robot actions and future expected values as latent frames within the native video diffusion process.

The *parallel paradigm* jointly models future video and actions within a shared generative process. Motus [3] projects both visual dynamics and control signals into a shared latent space using mixture-of-transformers (MoT) attention. LDA [39] extends this with latent dynamics alignment for improved temporal consistency. FastWAM [66] studies whether the gains of WAMs come primarily from video co-training during training or from explicit future imagination during inference, finding that the training-time world-modeling objective is the primary source of benefit. DreamZero [65] eliminates test-time video generation entirely while preserving the world-modeling training signal.

PILOT differs from these approaches in a fundamental way: rather than using pixel-level future prediction as either an intermediate representation or a joint generation target, we introduce the RD branch that operates on the latent space of a frozen VJEP2-AC encoder. This design achieves the representational benefits of future prediction (through CDE supervision during training) without paying the sequential inference penalty of pixel-level generation, while the Motion-CoT mechanism distills transition dynamics into dedicated semantic tokens that serve as an effective information bottleneck.

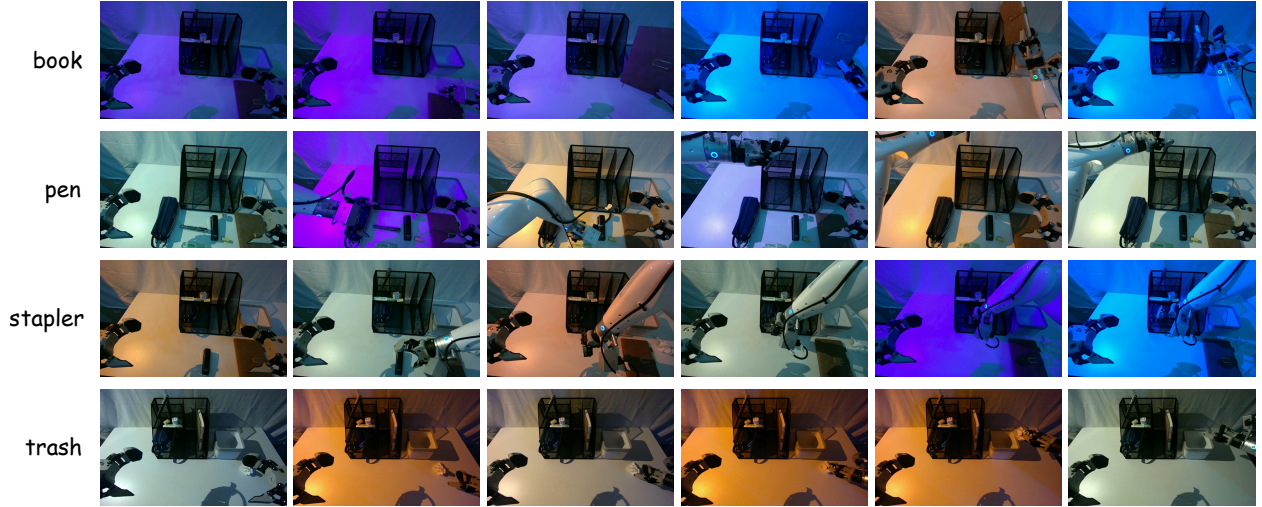
6.3. Joint Embedding Predictive Architectures

Joint Embedding Predictive Architectures (JEPA) [2, 43] represent an alternative to generative pixel reconstruction that mitigates uncertainty in pixel space by actively ignoring entropy-increasing stochastic texture details, thus focusing on predicting dynamic features such as object spatial positions and trajectories.

VJEP2-AC [2] produces patch-level representations conditioned on actions, making it particularly suitable for robotic manipulation where understanding action-conditioned state transitions is essential. LeWorld [42] applies JEPA-style prediction to world modeling for planning. Some studies [23, 49] have incorporated JEPA as an encoder to obtain rich representations without pixel reconstruction. Most relevant to our work is VLA-JEPA [51], which introduces JEPA to extract latent states, addressing the challenges of pixel reconstruction bias and future information leakage.

6.4. Latent Action Models

Latent Action Models (LAMs) [7, 13, 35, 48, 56, 57] aim to pre-train backbones on large-scale human videos lack-



Verify the generalization ability of PILOT
under various background lighting conditions and head camera rotations

Figure 8. Generalization evaluation under the *Flash + Camera Offset* setting. PILOT is deployed on the Agibot-G1 robot under strobe-light background interference and a rotated head-mounted camera, testing robustness to lighting distributional shifts and viewpoint changes.

ing action labels, thereby establishing prior knowledge of latent motion. Existing approaches [8, 15, 61, 64, 70] typically follow a two-stage pipeline: (1) encoding visual observations from adjacent frames into discrete latent tokens via a Q-former [28], and (2) aligning vision and manipulation representations using a unified codebook [11, 14–16, 41, 68] to integrate robotic manipulation into the Next-Token-Prediction framework.

Despite the promise of LAMs, their performance is hindered by noise from camera-induced background variations and future frame leakage, which encourages trivial encoding of future frames rather than learning state transitions [51]. Recent works mitigate this via latent constraints [37] or multi-encoder queries [49]. However, LAMs inherently align latent actions with visual space rather than actionable control signals.

PILOT fundamentally avoids these defects by ensuring that future visual information is used exclusively as a supervision target (through the CDE future-representation prediction objective), not as a source for capturing latent motion representations. The Motion-CoT tokens are distilled from the visual stream through the action perceiver branch, which operates on the current observation context rather than future frames, thereby guaranteeing that the model focuses on dynamic representations independent of camera motion and background changes.

6.5. Flow-Matching for Action Generation

Flow-matching [53] has emerged as a powerful framework for continuous-time generative modeling, offering simpler training objectives and more stable optimization compared

to traditional diffusion models. In the context of robotic action generation, flow-matching has been adopted by several recent works. The π_0 family [5, 21] employs flow-matching with a VAE-based action tokenizer. $\pi_{0.5}$ [21] extends this to multi-embodiment settings with internet-scale pre-training. OpenVLA-OFT [25] introduces orthogonal fine-tuning for flow-matching action experts.

PILOT adopts optimal-transport conditional flow matching (OT-CFM) [53] for its action decoder, which constructs straight-line probability paths between the action prior and target distribution. This choice offers two advantages: (1) the OT formulation minimizes the transport cost between noise and action distributions, leading to more efficient training convergence; and (2) with only 20 denoising steps, the action decoder achieves fast inference while maintaining high prediction quality, contributing to PILOT’s 145 Hz raw action frequency.

7. Implementation Details

7.1. Training Algorithm

Algorithm 1 details the full training procedure of WAM-VJEPa. The three branches are jointly optimized in a single forward-backward pass. Our codebase is built upon the StarVLA project [63]. We highlight several implementation choices that are important for reproduction but too specific for the main paper.

Action repeats. To exploit the inherent noise robustness of the flow-matching objective, each training example is

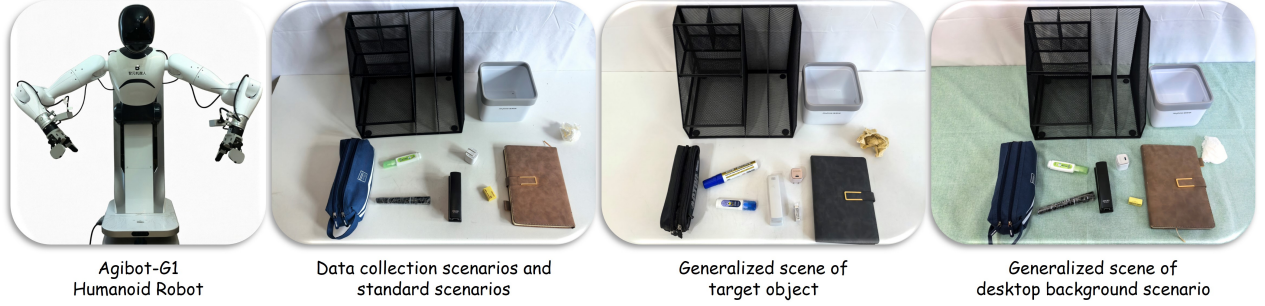


Figure 9. Overview of the real-world experimental setup. **Left:** the Agibot-G1 dual-arm humanoid robot used for data collection and evaluation. **Middle:** the teleoperation setup and standard evaluation scene on a clean desk. **Right:** two generalization transfer scenarios—*Flash + Camera Offset* (strobe lighting with shifted head camera) and *Desk* (replaced table surface).

replicated R times along the batch dimension with independent noise samples ϵ_r . This yields R denoising velocity predictions $\hat{v}_r$ that share the same context c and semantics m_r . Because the decoupled separation attention (Sec. 3.2.4) prevents action noise from leaking into the query slots, all m_r are numerically identical up to floating-point precision; we verify this with an `allclose` assertion (`atol = 10-5`) and average them to obtain a single deterministic motion-semantic context m per example.

Mixed-precision schedule. The comprehension pipeline (VideoDiT^{enc}) and the VJEP2-AC encoder E_ψ operate in `bfloat16` under `torch.autocast`; the Action DiT, the evolver G_ξ , and the two projectors W_m, W_s run in `float32`. This mirrors the original VJEP2-AC codebases and is critical for training stability, and is critical for training stability.

7.2. Inference Algorithm

At test time only the action pathway is required. Algorithm 2 describes Euler integration of the learned velocity field.

The RS branch and the future-frame generation pipeline incur *no* additional inference cost. Optionally, the Video DiT generation pipeline may be invoked to decode the predicted future frame for interpretability by running its full denoising loop separately.

7.3. Understanding vs. Generation Pipeline

A key architectural insight of PILOT is the decoupling of the Wan2.2 backbone into an *understanding pipeline* and a *generation pipeline* that share the same transformer weights but operate in fundamentally different modes at inference time. Algorithm 3 and Algorithm 4 contrast the two inference procedures step by step.

Structural comparison. Both pipelines share the TextEncoder (Qwen2.5-VL) and the VAE encoder as a common

Table 6. **Pipeline strategy comparison on RoboCasa.** Decoupled generation–understanding vs. predict-then-act. Latency in ms; frequency in Hz.

Strategy	Size	RoboCasa Avg.	Latency (ms)	Freq. (Hz)
Predict-then-Act	5.4B	58.4	1329.95	0.75
Decoupled (Ours)	5.4B	62.6	111.60	8.26

preprocessing stage (Steps 1–2). The critical divergence occurs afterward:

- **Understanding pipeline** (Algorithm 3): the Transformer performs a *single* forward pass (Step 4), directly extracting the last-layer hidden states as vl-tokens c . No iterative denoising loop, no scheduler, and no VAE decoding are required. The entire inference cost is dominated by this one Transformer forward pass.
- **Generation pipeline** (Algorithm 4): the Transformer is called N_{gen} times inside a denoising loop (Steps 5–7), each iteration predicting a velocity field that the scheduler uses to refine the noisy latent. After N_{gen} iterations, the VAE decoder maps the denoised latent back to pixel space (Step 8). The inference cost is therefore $N_{\text{gen}} \times$ Transformer forwards + 1 VAE decode.

Inference cost breakdown. With $N_{\text{gen}} = 50$ denoising steps in our implementation, the generation pipeline incurs $\sim 50 \times$ the Transformer compute of the understanding pipeline, plus an additional VAE decoding step. This explains the $11.9 \times$ wall-clock difference (1329.95 ms vs. 111.60 ms on RoboCasa: the per-iteration overhead of scheduler state management, attention mask recomputation, and the final VAE decode further amplify the gap beyond the raw $50 \times$ Transformer-call ratio. In PILOT, the generation pipeline is only activated during training (to provide future-frame supervision via $\mathcal{L}_{\text{wm}}$) and is entirely bypassed at inference, so action generation pays only the cost of the understanding pipeline.



Verify the generalization ability of PILOT
under the condition of changing the desktop background

Figure 10. Generalization evaluation under the *Desk* setting. The desk surface and background are replaced with unseen textures and colors, testing whether the policy relies on the original desk appearance or encodes transferable manipulation dynamics.

7.4. ActionEncoder Details

The ActionEncoder maps a noised action trajectory and its discretized time step into tokens of the Action DiT internal dimension d . Given the noised trajectory $\mathbf{a}^\tau \in \mathbb{R}^{B \times (H+1) \times d_a}$ and the scalar time step τ_{disc} :

1. **Project actions.** $\mathbf{e}_a = \text{Linear}(d_a \rightarrow d)(\mathbf{a}^\tau) \in \mathbb{R}^{B \times (H+1) \times d}$.
2. **Sinusoidal time encoding.** The scalar τ_{disc} is first broadcast to all $H+1$ steps, then encoded via a sinusoidal positional encoding: $\mathbf{e}_\tau = \text{SinPosEnc}(\tau_{\text{disc}}) \in \mathbb{R}^{B \times (H+1) \times d}$.
3. **Fuse.** Concatenate and process through a two-layer MLP:

$$\begin{aligned} \mathbf{A}^\tau &= \text{Linear}(d \rightarrow d) \left(\text{swish}(\text{Linear}(2d \rightarrow d)([\mathbf{e}_a; \mathbf{e}_\tau])) \right) \\ &\in \mathbb{R}^{B \times (H+1) \times d}. \end{aligned} \quad (14)$$

A learned positional embedding (distinct from the sinusoidal one above) is optionally added to $\mathbf{A}^\tau$ to distinguish the temporal position within the action horizon.

7.5. Full Perceiver Attention Block

We provide the complete forward computation of a single Perceiver-style attention block with the decoupled separation attention mechanism (Sec. 3.2.4). Let $\mathbf{Z} = [\mathbf{S}; \mathbf{Q}] \in \mathbb{R}^{B \times (K+1) \times d}$ denote the non-action latents, $\mathbf{A}^\tau \in$

$\mathbb{R}^{B \times (H+1) \times d}$ the action latents, $\mathbf{c} \in \mathbb{R}^{B \times N_c \times d}$ the projected vision-language context, and t the flow time embedding.

$$\bar{\mathbf{Z}} = \text{LN}(\mathbf{Z}), \quad (15)$$

$$\bar{\mathbf{A}} = \text{AdaLN}(\mathbf{A}^\tau, t), \quad (16)$$

$$\bar{\mathbf{c}} = \text{LN}(\mathbf{c}), \quad (17)$$

$$\mathbf{Z} \leftarrow \mathbf{Z} + \text{MHA}(\bar{\mathbf{Z}}, [\bar{\mathbf{Z}}; \bar{\mathbf{c}}]), \quad (18)$$

$$\mathbf{A}^\tau \leftarrow \mathbf{A}^\tau + \text{MHA}(\bar{\mathbf{A}}, [\bar{\mathbf{Z}}; \bar{\mathbf{A}}; \bar{\mathbf{c}}]), \quad (19)$$

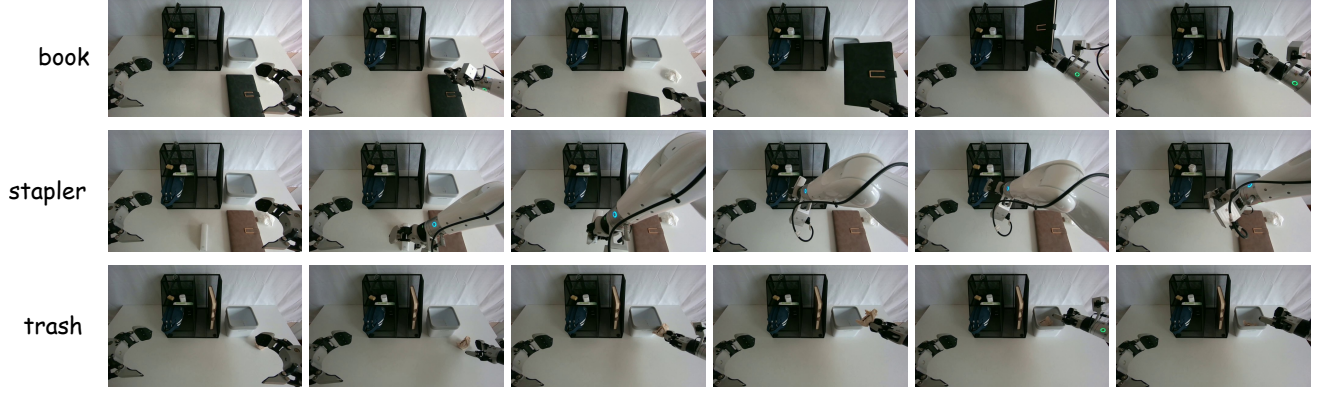
$$\mathbf{Z} \leftarrow \mathbf{Z} + \text{FFN}(\text{LN}(\mathbf{Z})), \quad (20)$$

$$\mathbf{A}^\tau \leftarrow \mathbf{A}^\tau + \text{FFN}(\text{AdaLN}(\mathbf{A}^\tau, t)), \quad (21)$$

where $\text{MHA}(\mathbf{q}, \mathbf{kv})$ denotes multi-head attention with $\mathbf{q}$ as queries and $\mathbf{kv}$ as both keys and values; FFN denotes $\text{Linear}(d \rightarrow 4d) \rightarrow \text{SquaredReLU} \rightarrow \text{Linear}(4d \rightarrow d)$; and $\text{AdaLN}(\mathbf{x}, t) = \gamma(t) \odot \text{LN}(\mathbf{x}) + \beta(t)$, with γ, β produced by a zero-initialized linear layer from $\text{SiLU}(t)$, ensuring that the block behaves as a standard LayerNorm at initialization. The key-value token counts are: $|\text{kv}_Z| = K+1+N_c = 261$ and $|\text{kv}_A| = K+1+(H+1)+N_c = 277$.

7.6. Evolver Internal Architecture

The evolver G_ξ is a causal Vision Transformer predictor initialized from the pre-trained VJEP2-AC predictor weights.



Verify the generalization ability of PILOT
under various shapes and colors of target objects

Figure 11. Generalization evaluation under the *Color* setting. Target objects are replaced with variants of different shapes and colors, testing the model’s ability to generalize beyond the appearance of training objects.

Its forward computation for WAM-VJEPa proceeds as follows.

Input assembly. The current-frame patch tokens $\mathbf{r}_t \in \mathbb{R}^{B \times P \times d_r}$ ($P = 256$, $d_r = 1408$) are first projected into the predictor embedding dimension $d_{\text{out}} = 1024$:

$$\mathbf{x}_0 = \text{Linear}(d_r \rightarrow d_{\text{out}})(\mathbf{r}_t) \in \mathbb{R}^{B \times P \times d_{\text{out}}}. \quad (22)$$

The projected motion-semantic context $\tilde{\mathbf{m}} \in \mathbb{R}^{B \times K \times d_{\text{out}}}$ and state embedding $\tilde{\mathbf{s}} \in \mathbb{R}^{B \times 1 \times d_{\text{out}}}$ are reshaped with a leading frame dimension $T = 1$:

$$\begin{aligned} \mathbf{x}_0 &\leftarrow \mathbf{x}_0.\text{view}(B, T, P, d_{\text{out}}), \\ \tilde{\mathbf{m}} &\leftarrow \tilde{\mathbf{m}}.\text{view}(B, T, K, d_{\text{out}}), \\ \tilde{\mathbf{s}} &\leftarrow \tilde{\mathbf{s}}.\text{view}(B, T, 1, d_{\text{out}}). \end{aligned} \quad (23)$$

Token concatenation. Conditioning tokens are prepended to the patch tokens:

$$\begin{aligned} \mathbf{x} &= [\tilde{\mathbf{m}}; \tilde{\mathbf{s}}; \mathbf{x}_0] \\ &\in \mathbb{R}^{B \times T(K+1+P) \times d_{\text{out}}} \\ &= \mathbb{R}^{B \times 321 \times 1024}, \end{aligned} \quad (24)$$

where $K + 1 + P = 64 + 1 + 256 = 321$.

Causal transformer blocks. The concatenated sequence is processed by $L_p = 24$ causal transformer blocks (ACBlock), each with a block-causal attention mask that respects both the frame ordering and the prefix position of the conditioning tokens. Within each block, conditioning tokens (indices 0 to K) can attend to all positions, while patch tokens (index $K+1$ onwards) may only attend to preceding positions within the same frame and all conditioning tokens.

Output projection. After the final block, the conditioning tokens are discarded and only the patch-token outputs are retained:

$$\begin{aligned} \mathbf{x}_{\text{out}} &= \mathbf{x}[:, K+1 :, :] \\ &\in \mathbb{R}^{B \times TP \times d_{\text{out}}} \\ &= \mathbb{R}^{B \times 256 \times 1024}. \end{aligned} \quad (25)$$

A LayerNorm followed by a linear projection maps back to the encoder dimension:

$$\begin{aligned} \hat{\mathbf{r}}_{t+\Delta} &= \text{Linear}(d_{\text{out}} \rightarrow d_r)(\text{LN}(\mathbf{x}_{\text{out}})) \\ &\in \mathbb{R}^{B \times 256 \times 1408}. \end{aligned} \quad (26)$$

Frozen action encoders. The original VJEPa2-AC predictor contains action encoders that map discrete actions to conditioning tokens. In WAM-VJEPa these encoders are bypassed (replaced by the projectors $\mathbf{W}_m, \mathbf{W}_s$) and explicitly frozen to prevent unused-parameter drift during optimization.

7.7. Change-Weighted Future-Frame Loss

The future-frame loss (Eq. 8 of the main paper) optionally applies a *change-weighted mask* that up-weights spatial regions where the future frame differs from the current observation.

Change binary mask. For each spatial location (i, j) in the latent space, we compute:

$$C_{ij} = \begin{cases} 1 & \text{if } \|\mathbf{z}_{ij}^{\text{fut}} - \mathbf{z}_{ij}^{\text{obs}}\| > \delta, \\ 0 & \text{otherwise,} \end{cases} \quad (27)$$

where δ is a threshold determined from the ρ -th percentile of the per-pixel latent distances within each batch.

Algorithm 1 WAM-VJEPa Training

Require: Observation o_t , future frame $o_{t+\Delta}$, instruction ℓ , state s_t , action chunk $\mathbf{a}_t$, action mask $\mathbf{M}$

Hyper-parameters: action repeats R , loss weights $\lambda_{\text{wm}}, \lambda_{\text{rs}}$

▷ **Branch 1 — Action & Motion Semantics**

- 1: $\mathbf{c} \leftarrow \text{VideoDiT}_{\theta}^{\text{enc}}(o_t, \ell) \in \mathbb{R}^{B \times 196 \times 2048}$ // bf16
- 2: Repeat $\mathbf{c}, \mathbf{a}_t, s_t$ along batch dim $\rightarrow \mathbf{c}', \mathbf{a}'_t, s'_t \in \mathbb{R}^{RB \times \dots}$
- 3: Sample $\tau \sim \text{Beta}(\alpha, \beta); \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 4: $\mathbf{a}^\tau \leftarrow (1-\tau)\epsilon + \tau \mathbf{a}'_t; \mathbf{v}^* \leftarrow \mathbf{a}'_t - \epsilon$
- 5: $\mathbf{S} \leftarrow \text{MLP}_{\text{state}}(s'_t); \mathbf{Q} \leftarrow \text{learnable queries}; \mathbf{A}^\tau \leftarrow \text{ActionEnc}(\mathbf{a}^\tau, \tau_{\text{disc}})$
- 6: $\mathbf{Z}_{\text{sa}} \leftarrow [\mathbf{S}; \mathbf{Q}; \mathbf{A}^\tau] \in \mathbb{R}^{RB \times 81 \times 768}$
- 7: $\mathbf{O} \leftarrow \text{ActionDiT}(\mathbf{Z}_{\text{sa}}, \mathbf{c}'; \tau)$ // fp32
- 8: $\hat{\mathbf{v}} \leftarrow \text{MLP}_{\text{dec}}(\mathbf{O}[:, -16 :, :]); \mathcal{L}_{\text{act}} \leftarrow \|\hat{\mathbf{v}} - \mathbf{v}^*\|_M^2$
- 9: $\mathbf{m}_r \leftarrow \mathbf{O}[:, 1:65, :]$ for each replica $r \in \{1, \dots, R\}$
- 10: $\mathbf{m} \leftarrow \frac{1}{R} \sum_r \mathbf{m}_r \in \mathbb{R}^{B \times 64 \times 1024}$ // average over replicas

▷ **Branch 2 — Future-Frame Generation**

- 11: $\mathbf{z}^* \leftarrow \text{VAE}_{\text{enc}}([o_t; o_{t+\Delta} \times 4]) \in \mathbb{R}^{B \times 2 \times 16 \times 28 \times 28}$
- 12: Sample $\tau' \sim \text{LogitNormal}(\mu, \sigma^2); \epsilon' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- 13: Noised: $\mathbf{z}^{\tau'} \leftarrow (1-\tau')\mathbf{z}^* + \tau'\epsilon'$ on future-latents only
- 14: $\hat{\mathbf{v}}_\theta \leftarrow \text{VideoDiT}_{\theta}^{\text{gen}}(\mathbf{z}^{\tau'}; \tau', \mathbf{p})$
- 15: $\mathcal{L}_{\text{wm}} \leftarrow \|\hat{\mathbf{v}}_\theta - (\epsilon' - \mathbf{z}^*)\|_{\odot}^2 M_{\text{tgt}}$

▷ **Branch 3 — Representation Simulation**

- 16: $\mathbf{r}_t, \mathbf{r}_{t+\Delta} \leftarrow E_\psi([o_t; o_{t+\Delta}])$ // bf16, no grad
- 17: $\tilde{\mathbf{m}} \leftarrow \mathbf{W}_m \mathbf{m}; \tilde{\mathbf{s}} \leftarrow \mathbf{W}_s s_t$ // project to $d_{\text{out}} = 1024$
- 18: $\hat{\mathbf{r}}_{t+\Delta} \leftarrow G_\xi(\mathbf{r}_t; \tilde{\mathbf{m}}, \tilde{\mathbf{s}})$ // fp32
- 19: $\mathcal{L}_{\text{rs}} \leftarrow \text{SmoothL1}(\hat{\mathbf{r}}_{t+\Delta}, \mathbf{r}_{t+\Delta})$

▷ **Total Loss**

- 20: $\mathcal{L} \leftarrow \mathcal{L}_{\text{act}} + \lambda_{\text{wm}} \mathcal{L}_{\text{wm}} + \lambda_{\text{rs}} \mathcal{L}_{\text{rs}}$
 - 21: Update $\theta, \xi, \mathbf{W}_m, \mathbf{W}_s$ via $\nabla_{\theta, \xi, \mathbf{W}_m, \mathbf{W}_s} \mathcal{L}$
-

Loss weight. The binary mask is softened with a linear ramp:

$$\mathbf{W}_{ij}^{\text{change}} = \begin{cases} w_{\text{max}} & \text{if } C_{ij} = 1, \\ 1 & \text{if } C_{ij} = 0, \end{cases} \quad (28)$$

and the final masked MSE loss becomes:

$$\mathcal{L}_{\text{wm}} = \frac{\sum_{ij} M_{\text{tgt}, ij} \mathbf{W}_{ij}^{\text{change}} \|\hat{\mathbf{v}}_{ij} - \mathbf{v}_{ij}^*\|^2}{\sum_{ij} M_{\text{tgt}, ij} \mathbf{W}_{ij}^{\text{change}}}. \quad (29)$$

When `masked_denoising` is enabled, the noised input is further replaced by the clean latent at statically-unchanged positions, so the model only denoises regions that actually change.

Algorithm 2 WAM-VJEPa Inference (Action Only)

Require: Observation o_t , instruction ℓ , state s_t ; inference steps N

- 1: $\mathbf{c} \leftarrow \text{VideoDiT}_{\theta}^{\text{enc}}(o_t, \ell) \in \mathbb{R}^{B \times 196 \times 2048}$
 - 2: $\mathbf{S} \leftarrow \text{MLP}_{\text{state}}(s_t) \in \mathbb{R}^{B \times 1 \times 768}$
 - 3: $\mathbf{Q} \leftarrow \text{learnable queries} \in \mathbb{R}^{B \times 64 \times 768}$
 - 4: $\mathbf{a}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^{B \times 16 \times 32}$
 - 5: **for** $n = 0, 1, \dots, N-1$ **do**
 - 6: $t \leftarrow n/N; t_{\text{disc}} \leftarrow \lfloor t \cdot 1000 \rfloor$
 - 7: $\mathbf{A}^t \leftarrow \text{ActionEnc}(\mathbf{a}^{(n)}, t_{\text{disc}})$
 - 8: $\mathbf{Z} \leftarrow [\mathbf{S}; \mathbf{Q}; \mathbf{A}^t] \in \mathbb{R}^{B \times 81 \times 768}$
 - 9: $\mathbf{O} \leftarrow \text{ActionDiT}(\mathbf{Z}, \mathbf{c}; t_{\text{disc}})$
 - 10: $\hat{\mathbf{v}} \leftarrow \text{MLP}_{\text{dec}}(\mathbf{O}[:, -16 :, :]) \in \mathbb{R}^{B \times 16 \times 32}$
 - 11: $\mathbf{a}^{(n+1)} \leftarrow \mathbf{a}^{(n)} + \frac{1}{N} \hat{\mathbf{v}}$
 - 12: **end for**
 - 13: **return** $\mathbf{a}^{(N)}$
-

Algorithm 3 Understanding Pipeline Inference

Require: Observation o_t , instruction ℓ

Shared encoders (run once):

- 1: $\mathbf{p} \leftarrow \text{TextEncoder}(\ell) \in \mathbb{R}^{B \times 512 \times 100352}$ // Qwen2.5-VL, frozen
- 2: $\mathbf{z}_{\text{obs}} \leftarrow \text{VAE}_{\text{enc}}(o_t) \in \mathbb{R}^{B \times 16 \times 1 \times 28 \times 28}$ // Wan2.2 VAE, frozen

▷ **Single forward pass**(no iterative denoising):

- 3: $\mathbf{h} \leftarrow \text{Transformer}(\mathbf{z}_{\text{obs}}, \mathbf{p}; t=0.1)$ // 1 forward, bf16
 - 4: $\mathbf{c} \leftarrow \mathbf{h}.\text{last_layer} \in \mathbb{R}^{B \times 196 \times 2048}$ // extract via hook
 - 5: **return** $\mathbf{c}$ // vl-tokens for Action Head
-

7.8. Flow Time Sampling

Action branch. The flow time τ for the action OT-CFM objective is sampled from a shifted Beta distribution:

$$\tau = \frac{s-x}{s}, \quad x \sim \text{Beta}(\alpha, \beta), \quad s = 0.999, \quad (30)$$

with default $\alpha = 1.5, \beta = 1.0$, which concentrates mass near $\tau = 1$ (clean actions) and thus emphasizes the final denoising stages.

World-model branch. The generation pipeline samples τ' from a Logit-Normal distribution with configurable (μ, σ^2) , followed by a time-shift:

$$\begin{aligned} \tau' &= \frac{\sigma u + \mu}{1 + |\sigma u + \mu|}, \quad u \sim \mathcal{N}(0, 1), \\ t_{\text{shift}} &= \frac{s' \tau'}{1 + (s' - 1) \tau'}, \end{aligned} \quad (31)$$

with shift factor $s' = 5.0$ and t_{shift} discretized via $\lfloor t_{\text{shift}} \times 1000 \rfloor$ for the timestep embedding.

Algorithm 4 Generation Pipeline Inference (Predict-then-Act)

Require: Observation o_t , instruction ℓ ; denoising steps N_{gen}

Shared encoders (run once):

- 1: $\mathbf{p} \leftarrow \text{TextEncoder}(\ell) \in \mathbb{R}^{B \times 512 \times 100352}$ // Qwen2.5-VL, frozen
- 2: $\mathbf{z}_{\text{obs}} \leftarrow \text{VAE}_{\text{enc}}(o_t) \in \mathbb{R}^{B \times 16 \times 1 \times 28 \times 28}$ // Wan2.2 VAE, frozen
- 3: $\mathbf{z}^{(0)} \leftarrow \epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ // initialize noise for future latents
- 4: **for** $i = 0, 1, \dots, N_{\text{gen}} - 1$ **do**
- 5: $\hat{\mathbf{v}}_i \leftarrow \text{Transformer}([\mathbf{z}_{\text{obs}}; \mathbf{z}^{(i)}], \mathbf{p}; t_i)$ // forward pass # $i+1$
- 6: $\mathbf{z}^{(i+1)} \leftarrow \text{SchedulerStep}(\mathbf{z}^{(i)}, \hat{\mathbf{v}}_i, t_i)$ // UniPC scheduler update
- 7: **end for**
- 8: $\hat{o}_{t+\Delta} \leftarrow \text{VAE}_{\text{dec}}(\mathbf{z}^{(N_{\text{gen}})})$ // decode future frame
- 9: **return** $\hat{o}_{t+\Delta}$ // predicted future image

7.9. Architecture Hyper-parameters

Table 7 summarizes the key architectural hyper-parameters of WAM-VJEPa.

7.10. Model Architecture Details

World Model Backbone. We adopt Wan2.2-TI2V-5B as the world model backbone, a 5-billion-parameter video DiT that serves a dual role: (i) as a comprehension encoder that maps the current observation o_t and language instruction ℓ into context tokens $\mathbf{c} \in \mathbb{R}^{B \times 196 \times 2048}$, and (ii) as a generation decoder that denoises future-frame latents via a flow-matching velocity objective. The encoder extracts hidden states from the last layer (`extract_layers=[-1]`) of the DiT backbone. Input images are resized to 224×224 for both LIBERO and RoboCasa benchmarks, with a conditional-frame timestep of 0.1. All the components in the backbone are trainable.

The future-frame generation branch operates in the latent space of the accompanying VAE encoder. Given a current observation o_t and a future frame $o_{t+\Delta}$, the VAE encodes the concatenated frame pair into $\mathbf{z}^* \in \mathbb{R}^{B \times 2 \times 16 \times 28 \times 28}$. A *MaskFutureImage* strategy then adds noise only to the future latent channels, preserving the current-frame channels as conditioning. The Video DiT generation pipeline predicts the velocity field $\hat{\mathbf{v}}_\theta$ and is trained with a change-weighted MSE loss (Sec. 7.7). Flow time τ' is sampled from a Logit-Normal distribution with shift factor $s' = 5.0$ (Sec. 7.8). At inference, the generation pipeline uses 20 denoising steps with classifier-free guidance scale 5.0, pro-

ducing a 224×224 future image for both benchmarks.

Representation Simulation. The VJEPa2-AC encoder E_ψ (ViT-G, $\sim 1\text{B}$ parameters) is initialized from `vjepa2-ac-vitg.pt` and remains fully frozen. It produces patch-level representations $\mathbf{r}_t, \mathbf{r}_{t+\Delta} \in \mathbb{R}^{B \times 256 \times 1408}$ for the current and future frames. The evolver G_ξ (24-layer causal transformer, $d_{\text{out}} = 1024$) takes the projected motion-semantic context $\tilde{\mathbf{m}}$ and state embedding $\tilde{\mathbf{s}}$ as conditioning tokens, concatenated with $\mathbf{r}_t$ to form a 321-token sequence (Eq. 24). The two projectors $\mathbf{W}_m$ and $\mathbf{W}_s$ are linear layers that map from $d = 768$ and $d_s = 64$ respectively into $d_{\text{out}} = 1024$. The representation-simulation loss uses Smooth- ℓ_1 with $\lambda_{\text{rs}} = 1.0$.

Action Head. The Action Head is a 16-layer Perceiver-style DiT (denoted DiT-B) with internal dimension $d = 768$ and 16 attention heads. It consumes 81 latent tokens: 1 state token (MLP : $64 \rightarrow 768$), 64 future-query tokens (`nn.Embedding(64, 768)`, initialized from $\mathcal{N}(0, 0.02)$), and 16 action tokens produced by the ActionEncoder (Sec. 7.4). Cross-attention is performed against the 196 vision-language context tokens $\mathbf{c}$ with cross-attention dimension 2048. The decoupled separation attention mechanism (Sec. 7.5) ensures that action noise does not contaminate the query slots. Flow time is injected exclusively into the action-token group via AdaLN with zero-initialized modulation parameters. The action decoder is a two-layer MLP ($1024 \rightarrow 1024 \rightarrow 32$) that maps the final action-token outputs to the predicted velocity in $\mathbb{R}^{32}$. During training, each example is replicated $R = 4$ times with independent noise samples; the 64 query-token outputs are averaged across replicas to form the deterministic motion-semantic context $\mathbf{m}$. At inference, the action trajectory is recovered via Euler integration of the learned velocity field (Algorithm 2).

7.11. Action DiT Flow-Matching Formulation

We provide a detailed derivation of the Action DiT flow-matching objective used in PILOT. Let $\mathbf{a}_{1:H}$ denote an action trajectory of horizon H and let $\mathbf{a}^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ be the noise source. We adopt the Optimal-Transport Conditional Flow Matching (OT-CFM) [53] formulation, which constructs a conditional probability path and vector field that transport $\mathbf{a}^0$ to $\mathbf{a}_{1:H}$.

Probability path and vector field. Given a data sample $\mathbf{a}_{1:H}$ and noise $\mathbf{a}^0$, the conditional flow is the linear interpolation

$$\mathbf{a}^\tau = (1 - \tau) \mathbf{a}^0 + \tau \mathbf{a}_{1:H}, \quad \tau \in [0, 1]. \quad (32)$$

Component	Parameter	Value
Video DiT	Context tokens N_c	196
	Context dimension d_c	2048
	Num. layers (enc / gen)	28 / 28
Action DiT	Internal dimension d	768
	Output dimension d_{out}	1024
	Num. Perceiver layers L	16
	Num. attention heads	16
	Num. query tokens K	64
	FFN multiplier	$4 \times$ (SquaredReLU)
State / Action	State dimension d_s	64
	Action dimension d_a	32
	Action horizon $H + 1$	16
VJEP2-AC	Patch tokens P	256
	Representation dim d_r	1408
	Predictor layers L_p	24
Flow Matching	Timestep buckets	1000
	WM inference steps	20
	Action inference steps	20
	Action repeats R	4

Table 7. Architecture hyper-parameters.

The corresponding conditional vector field is

$$\mathbf{u}(\mathbf{a}^\tau \mid \mathbf{a}^0, \mathbf{a}_{1:H}) = \frac{d}{d\tau} \mathbf{a}^\tau = \mathbf{a}_{1:H} - \mathbf{a}^0. \quad (33)$$

Under the OT-CFM framework, the marginal vector field that generates the marginal probability path $p_\tau(\mathbf{a}^\tau)$ is

$$\mathbf{v}_\tau(\mathbf{a}^\tau) = \mathbb{E}_{\mathbf{a}^0, \mathbf{a}_{1:H} \mid \mathbf{a}^\tau} [\mathbf{a}_{1:H} - \mathbf{a}^0]. \quad (34)$$

Training objective. The Action DiT $\mathbf{f}_\phi$ is trained to regress the conditional vector field via

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{\tau, \mathbf{a}^0, \mathbf{a}_{1:H}} \left[\left\| \mathbf{f}_\phi(\mathbf{a}^\tau, \tau, \mathbf{c}, \tilde{\mathbf{m}}, \tilde{\mathbf{s}}) - (\mathbf{a}_{1:H} - \mathbf{a}^0) \right\|^2 \right], \quad (35)$$

where $\mathbf{c}$ are the vision–language context tokens, $\tilde{\mathbf{m}}$ the projected motion-semantic context, and $\tilde{\mathbf{s}}$ the projected robot-state embedding.

Flow time sampling. The flow time τ is sampled from a Logit-Normal distribution $\text{LogitNormal}(\mu=0, \sigma=1)$ with a learnable shift $s=3.0$ for the action branch (different from the generation branch’s $s'=5.0$), ensuring more samples near $\tau=1$ for accurate endpoint prediction (Sec. 7.8).

Classifier-free guidance. During inference, we apply classifier-free guidance with scale w :

$$\hat{\mathbf{v}}_\tau = (1+w) \mathbf{f}_\phi(\mathbf{a}^\tau, \tau, \mathbf{c}, \tilde{\mathbf{m}}, \tilde{\mathbf{s}}) - w \mathbf{f}_\phi(\mathbf{a}^\tau, \tau, \emptyset, \tilde{\mathbf{m}}, \tilde{\mathbf{s}}), \quad (36)$$

where $\emptyset$ denotes a null language embedding. We use $w=1.5$ by default and 20 Euler integration steps.

Action chunking and horizon. PILOT predicts an action chunk of horizon $H=50$ for LIBERO and $H=100$ for RoboCasa, but executes only the first H_a actions before re-planning. Following prior work [71], we use $H_a=8$ for LIBERO and $H_a=50$ for RoboCasa. The temporal ensemble weighting is applied over overlapping predictions.

Separation attention in action DiT. A key architectural detail is the *decoupled separation attention* (Sec. 7.5): the 81 tokens inside the action DiT are partitioned into *state* + *query* (65 tokens) and *action* (16 tokens) groups. Self-attention is performed *within* each group independently; cross-attention against $\mathbf{c}$ is shared across both groups. This ensures that the noisy action tokens at flow time τ cannot leak information into the query slots that must remain deterministic, which is critical for the quality of the motion-semantic context $\mathbf{m}$.

Loss weighting. The total training loss combines three objectives:

$$\mathcal{L} = \lambda_{\text{act}} \mathcal{L}_{\text{act}} + \lambda_{\text{gen}} \mathcal{L}_{\text{gen}} + \lambda_{\text{rs}} \mathcal{L}_{\text{rs}}, \quad (37)$$

where $\lambda_{\text{act}}=1.0$, $\lambda_{\text{gen}}=1.0$, $\lambda_{\text{rs}}=1.0$. All three losses are averaged over the batch dimension; $\mathcal{L}_{\text{gen}}$ further includes the change-weighted mask (Sec. 7.7).

7.12. Training Details

Data preprocessing. For both LIBERO and RoboCasa benchmarks, observations are resized to 224×224 pix-

els and normalized using the ImageNet mean and standard deviation. We adopt `view_mode=primary` for both benchmarks, extracting only the primary camera view as input. The video backends differ: LIBERO uses `torchvision_av` while RoboCasa uses `decord`, reflecting the different episode storage formats of each benchmark. The future-frame index is set to $\Delta = 16$ timesteps ahead. The robot proprioceptive state vector has dimension 8 for LIBERO and 64 for RoboCasa, and the action dimension is 7 and 32 respectively.

Optimizer and learning rate. We use the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1, and an initial learning rate schedule with linear warmup followed by cosine decay to a minimum learning rate. For **LIBERO**, the peak learning rates are $\text{lr}_{\text{base}} = 1 \times 10^{-5}$, $\text{lr}_{\text{action}} = 1 \times 10^{-4}$, $\text{lr}_{\text{predictor}} = 1 \times 10^{-4}$, $\text{lr}_{\text{projectors}} = 1 \times 10^{-4}$, with warmup for 1,000 steps and $\text{min_lr} = 1 \times 10^{-6}$. For **RoboCasa**, all learnable components share a unified learning rate $\text{lr} = 1 \times 10^{-5}$ (base, action model, predictor, projectors), with warmup for 500 steps and $\text{min_lr} = 1 \times 10^{-7}$.

Batch size and hardware. All models are trained on 32 NVIDIA H20 GPUs using DeepSpeed ZeRO Stage 2. For LIBERO, the per-device batch size is 16, yielding an effective batch size of 512. For RoboCasa, the per-device batch size is 8, yielding an effective batch size of 256. For LIBERO, we trained for a total of 100,000 steps (taking approximately 90 hours), while for Robocasa, we trained for a total of 300,000 steps (taking approximately 200 hours).

Training hyper-parameters. Table 8 summarizes the key training hyper-parameters for both benchmarks.

Checkpointing and evaluation. We save training checkpoints every 10,000 steps and evaluate the policy on the benchmark tasks at each checkpoint. The final reported results correspond to the checkpoint with the highest average success rate across all tasks within each benchmark.

7.13. Additional Experimental Details

LIBERO benchmark. LIBERO [34] provides 130 manipulation tasks across 5 task suites built on the Franka Panda robot in the MuJoCo simulator. We evaluate on LIBERO-Spatial (10 tasks), LIBERO-Object (10 tasks), LIBERO-Goal (10 tasks), and LIBERO-Long (10 tasks) for a total of 40 tasks, following the standard protocol. Each task is evaluated over 20 episodes, and the success rate is averaged across all tasks. The action space is 7-dimensional (EE pose + gripper), and the proprioceptive state is 8-dimensional. Observations consist of a primary wrist-camera RGB image at 224×224 resolution.

RoboCasa-GR1 benchmark. RoboCasa [4] is a large-scale simulation platform for household manipulation tasks featuring the GR1 humanoid robot with 29 degrees of freedom. We use the RoboCasa-GR1 Tabletop Task suite [4] which provides 24 manipulation tasks across 5 categories (Pick, Stack, Pour, Navigate, and Composite). Each task is evaluated over 20 episodes. The action space is 32-dimensional (joint positions + gripper), and the proprioceptive state is 64-dimensional. Observations consist of a primary third-person RGB image at 224×224 resolution.

Baselines. We compare PILOT against the following baselines, organized by model category:

- *Vision-Language-Action Models.* π_0 [5] and $\pi_{0.5}$ [21] are VLM-based flow-matching policies built on PaliGemma with internet-scale pre-training. **GR00T-N1** [4] is a VLA baseline built on the SmolVLM backbone, and **GR00T-N1.6** [44] is its scaled successor. **OpenVLA** [24] and **OpenVLA-OFT** [25] are open-source VLAs fine-tuned with optimal transport. **ECot** [67] incorporates explicit chain-of-thought reasoning for manipulation. **Spatial Forcing** [27] augments VLA training with spatial-reasoning regularization. π_0 -**FAST** [46] is a fast-distilled variant of π_0 .
- *Latent Action Models.* **LAPA** [64] pre-trains latent action tokens from large-scale human videos. **UniVLA** [8] unifies visual-action representations via a shared codebook. **Mantis** [62] aligns latent actions with multi-modal visual features. **VLA-JEPA** [51] introduces a JEPA component to address pixel reconstruction bias and future information leakage. **PokeVLA** [72] leverages poke-style interaction priors for manipulation.
- *World-Action Models.* **Fast-WAM** [66] accelerates world-action inference via a parallel prediction head. **Motus** [3] jointly models visual and action tokens through shared attention (MoT). **LDA** [39] learns latent dynamics alignment for world-action prediction. **F1** [38] is a world-action model with a factorized prediction architecture. **WorldVLA** [10] integrates world-model predictions into VLA generation. **Cosmos-Policy** [26] is a Cosmos-based video prediction policy. **DiT4DiT** [40] employs a DiT-for-DiT architecture for world-action modeling. **UWM-XL** [74] is a universal world model at XL scale. **StarVLA** [63] is a star-topology VLA with distributed attention. **VP-VLA** [58] incorporates video prediction into VLA training. **PhysBrain** [33] embeds physical reasoning into world-action generation. **Lang-Force** [32] leverages language-guided force estimation for manipulation.

Evaluation protocol. For both benchmarks, we evaluate each method by running 20 rollouts per task from the initial state distribution. Success is determined by the task-specific

Category	Hyper-parameter	LIBERO	RoboCasa	Real-World
Data	Resolution	224×224	224×224	224×224
	View mode	primary	primary	3-view
	Video backend	torchvision_av	decord	decord
	Future-frame Δ	16	16	16
Dimensions	Action dim	7	32	32
	State dim	8	64	64
	Horizon H	50	100	30
Optimization	Optimizer	AdamW ($\beta_1 = 0.9, \beta_2 = 0.95$)		AdamW ($\beta_1 = 0.9, \beta_2 = 0.999$)
	Weight decay		0.1	
	Training steps	100,000	300,000	100,000
	Warmup steps	1,000	3,000	1,000
	Min. learning rate	1×10^{-6}	1×10^{-7}	1×10^{-7}
	Gradient clipping		1.0	
Learning Rate	lr_{base}	1×10^{-5}	1×10^{-5}	1×10^{-5}
	lr_{action}	1×10^{-4}	5×10^{-5}	5×10^{-5}
	$lr_{predictor}$	1×10^{-4}	5×10^{-5}	5×10^{-5}
	$lr_{projectors}$	1×10^{-4}	5×10^{-5}	5×10^{-5}
Hardware & Batch	GPUs		32 $\times$ NVIDIA H20	
	Per-device BS	16	8	16
	Effective BS	512	256	512
	Precision		BF16 mixed	
Flow Matching	WM inference steps		20	
	Action inference steps		20	
	CFG scale	1.5		5.0
	Action repeats R		4	
Checkpointing	Save interval	Every 10,000 steps		

Table 8. Training hyper-parameters for PILOT.

completion criteria. A task is considered successful if the goal condition is met within the maximum episode length (500 steps for LIBERO, 1000 steps for RoboCasa). We report the average success rate across all tasks within each suite.

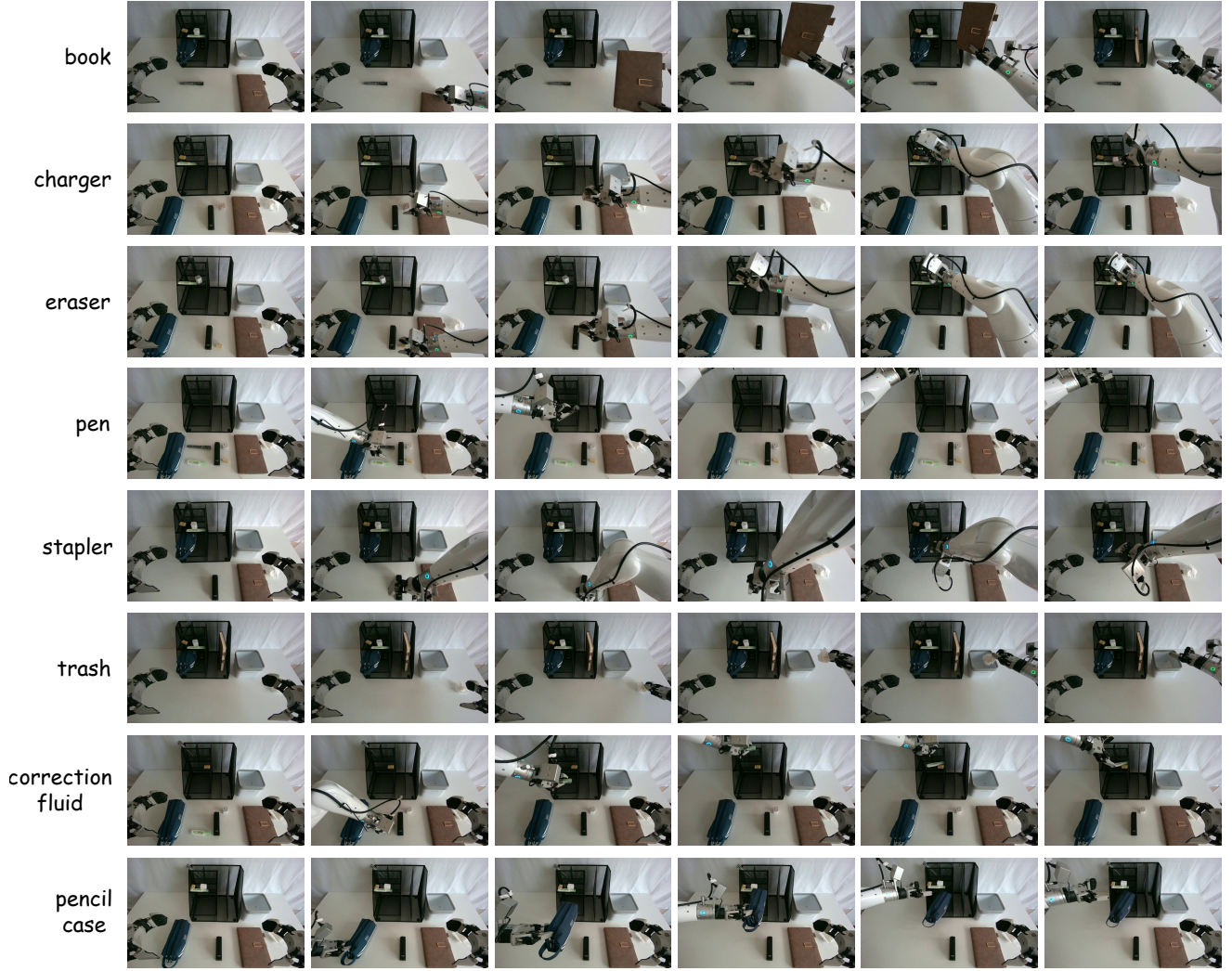
Real-World benchmark. We deploy PILOT on an Agibot-G1 dual-arm humanoid robot to evaluate its performance on real-world desk manipulation tasks. The robot features 7-DOF arms on each side, a 1-DOF gripper per hand, a 2-DOF head (pan/tilt), and a 2-DOF waist, yielding a 32-dimensional action space and a 64-dimensional proprioceptive state. Observations are captured from three RGB cameras at 224×224 resolution: a head-mounted camera (`image_head_color`), a left-hand wrist camera (`image_hand_left`), and a right-hand wrist camera (`image_hand_right`), providing complementary global and egocentric viewpoints. Figure 9 shows the Agibot-G1 robot, the teleoperation-based data collection setup, the standard evaluation scene, and two representative generalization scenarios.

The demonstration data is collected via teleoperation. The `desk_all_3view` data mixture comprises 8 sub-datasets collected under consistent desk-top conditions, each corresponding to a manipulation task with objects

commonly found on office desks. In total, the mixture aggregates 32 source recording sessions, yielding 3,965 demonstration trajectories per sub-task and a combined total of 7.1M frames across all tasks. Visual observations are recorded by two RealSense D405 cameras mounted on the left and right wrists (480×848 native resolution) and one head-mounted camera (800×1280), all operating at 30 fps. The data is stored in the LeRobot format with min-max normalization applied to all state and action keys. We use the following 8 standard tasks for evaluation: *pen*, *eraser*, *correction fluid*, *pencil case*, *stapler*, *book*, *trash*, and *charger*. Figure 12 illustrates PILOT deployed on the Agibot-G1 robot performing the 8 standard tasks, ranging from precise manipulation (e.g., inserting a pen into a grid slot) to grasping specific objects on the desk, with the most challenging task being picking up a book and placing it upright into a storage rack.

Table 9 summarizes the dataset statistics for each task.

Evaluation settings. We evaluate under three settings following the main text: (1) **Standard**: 50 rollouts per task with the default initial state distribution; (2) **Generalization**: three variants—*Flash + Camera Offset*, which introduces sudden lighting changes and a shifted camera position; *Desk*, which replaces the desk surface and background; and *Color*, which replaces object colors with un-



The PILOT was deployed on the Agibot-G1 robot and tested on eight challenging desktop tasks

Figure 12. PILOT deployed on the Agibot-G1 robot performing 8 standard desk manipulation tasks. The tasks include inserting a pen into a grid slot (requiring high precision), grasping and placing various desktop objects, and the most challenging task of picking up a book and placing it upright into a storage rack.

seen variants; Figures 8, 10, and 11 visualize the three generalization transfer scenarios, respectively.

(3) **Few-Shot (10%)**: only 10% of the demonstration data is used for training, controlled by the `fewshot_ratio=0.1` flag in the training script.

Ablation configurations. We conduct four sets of ablation experiments corresponding to different training configurations:

- **PILOT (Full)**: Frozen mode `wm_train_vlm_frozen` with the full training set—the default setting that trains both the World Model (WM) branch and action modules

while keeping the VLM backbone frozen.

- **PILOT (Few-Shot)**: Same frozen mode as Full but with `fewshot_ratio=0.1`, using only 10% of the demonstration data.
- **RD-only**: Frozen mode `wm_frozen_vlm_frozen` with `future_image_training.enabled=false`, which disables the future-image generation branch entirely, retaining only the Representational Deduction (RD) pathway.
- **RD-only (Few-Shot)**: Same as RD-only but with `fewshot_ratio=0.1`.

Task	Directory	Trajectory	Length	Frames	Frequency
Pen	g1_pen	3,965	248	983,400	30
Eraser	g1_eraser	3,965	158	625,110	30
Correction fluid	g1_correction	3,965	220	872,097	30
Pencil case	g1_pencil	3,965	260	1,030,137	30
Stapler	g1_stapler	3,965	210	830,546	30
Book	g1_books	3,965	418	1,657,390	30
Trash	g1_trash	3,965	136	537,195	30
Charger	g1_charger	3,965	151	599,312	30
Total	—	—	—	7,135,187	—

Table 9. Real-world desk manipulation dataset statistics. *Traj.* = number of demonstration trajectories; *Len.* = average timesteps per trajectory; *Frames* = total frames; *Freq.* = control frequency (Hz). All tasks share the same 3,965 trajectories from 32 source sessions.

These configurations allow us to isolate the contribution of the WM generation branch versus the RD pathway, as well as the data efficiency of each component.

7.14. Inference Efficiency

A fundamental architectural question is whether generation and understanding should be interleaved sequentially (predict-then-act) or decoupled into parallel cooperative branches. In the predict-then-act paradigm, the Wan2.2 backbone operates in its *generation mode*: the Transformer is invoked 50 times within an iterative denoising loop (each pass predicting a velocity field that a scheduler uses to refine the noisy latent), followed by a VAE decoding step to produce a future pixel frame, *before* the Action Head can begin inference (see Algorithm 4). Our decoupled design instead uses the Wan2.2 backbone in its *understanding mode*: the Transformer performs a *single forward pass* that directly extracts vision–language context tokens, bypassing the 50-step denoising loop and VAE decoder entirely (Algorithm 3). The understanding pipeline thus eliminates three costly components that the generation pipeline requires: (1) the iterative denoising scheduler loop ($50\times$ Transformer forwards), (2) the VAE decoding step, and (3) the per-iteration attention mask and scheduler state management.

Table 6 compares these two strategies on RoboCasa. The two models achieve nearly identical performance (62.6% vs. 58.4%), confirming that decoupling does *not* sacrifice task accuracy. However, the inference cost differs drastically: the decoupled pipeline runs at 111.60 ms per step (8.26 Hz), while predict-then-act requires 1329.95 ms (0.75 Hz)—an $11.9\times$ slowdown. This gap is primarily attributable to the $50\times$ Transformer forward-pass ratio between the two pipelines; scheduler overhead and VAE decoding further amplify the difference beyond the raw compute ratio. These results demonstrate that the Representational Deduction mechanism achieves the representational benefits of future prediction (provided by the CDE during training) without paying the sequential inference penalty of

pixel-level generation.

PILOT achieves 6.9 ms per action step (145 Hz raw frequency), which is $2.1\times$ faster than π_0 (14.5 ms) and $1.7\times$ faster than CogACT (11.8 ms). The effective control frequency, accounting for the fact that PILOT predicts actions over a horizon (16 steps for RoboCasa), reaches 58 Hz—well above the 20 Hz threshold for real-time closed-loop control. This efficiency stems from two design choices: (1) the Causal Dynamics Engine is a lightweight transformer with only 4 layers, adding minimal overhead; and (2) the flow-matching decoder uses only 20 denoising steps, reducing the action generation cost.

7.15. Full RoboCasa-GR1 Results

Table 10 presents the per-category results on the RoboCasa-GR1 Tabletop Task benchmark. PILOT achieves the highest average success rate across all categories, with particularly large improvements on the Composite and Pour categories that require multi-step reasoning and precise control.

Task	GR00T N1.5 [†]	GR00T N1.6 [†]	DiT4DiT	UWM-XL	StarVLA	LDA	VP-VLA	PhysBrain	LangForce	FastWAM	Motus	PILOT
<i>PnP Close</i>												
PnP Bottle to Cab.	54	51.5	48	41	46	76	54	74	72	58	55	70
PnP Can to Drawer	50	13	74	53	80	71	72	68	78	60	57	54
PnP Cup to Drawer	38	8.5	52	12	54	41	44	42	46	45	42	50
PnP Milk to Micro.	60	14	50	25	48	52	74	54	56	54	50	68
PnP Potato to Micro.	32	41.5	36	29	28	41	34	24	36	43	40	62
PnP Wine to Cab.	38	16.5	42	24	46	57	48	54	46	56	52	54
PnP * to * Close (Avg)	45.3	24.2	50.3	30.7	50.3	56.3	54.3	52.7	55.7	52.7	49.3	59.7
<i>PnP Novel From Cuttingboard</i>												
NvCB to Basket	38	58	52	18	48	65	66	62	66	64	60	84
NvCB to CbBox	46	46.5	48	14	40	69	54	44	40	58	54	58
NvCB to Pan	58	68.5	76	20	68	75	74	56	68	72	68	86
NvCB to Pot	62	65	62	25	52	61	54	58	48	59	55	44
NvCB to TBasket	28	46.5	50	10	56	51	56	40	44	50	46	34
Cuttingboard * (Avg)	46.4	56.9	57.6	17.4	52.8	64.2	60.8	52	53.2	60.6	56.6	61.2
<i>PnP Novel From Placemat</i>												
NvPM to Basket	30	58.5	50	16	42	53	48	42	54	55	51	72
NvPM to Bowl	60	57.5	56	10	44	55	74	56	62	58	54	58
NvPM to Plate	56	63	32	12	48	59	70	80	52	54	50	64
NvPM to TShelf	26	28.5	18	2	18	24	26	14	24	22	20	30
Placemat * (Avg)	45.5	51.9	39	10	38	47.8	54.5	48	48	47.2	43.8	56
<i>PnP Novel From Tray</i>												
NvTR to CbBox	52	51.5	38	25	38	65	44	40	50	62	58	86
NvTR to Plate	48	71	56	18	56	63	66	66	58	64	60	94
NvTR to Pot	60	64.5	54	25	50	55	38	52	62	60	56	92
NvTR to TBasket	52	57	46	16	36	51	58	50	44	52	48	80
NvTR to TShelf	32	31.5	38	2	16	33	24	22	22	36	32	22
Tray * (Avg)	48.8	55.1	46.4	17.2	39.2	53.4	46	46	47.2	54.8	50.8	74.8
<i>PnP Novel From Plate</i>												
NvPL to Bowl	58	57	56	8	60	53	52	54	54	56	52	52
NvPL to CbBox	44	43.5	58	10	50	43	44	50	48	50	46	52
NvPL to Pan	60	51	68	20	54	55	56	68	54	60	56	50
NvPL to Plate	64	78.7	58	27	70	61	62	78	78	68	64	88
Plate * (Avg)	56.5	57.5	60	16.2	58.5	53	53.5	62.5	58.5	58.5	54.5	60.5
Average (All 24 Tasks)	48.2	47.6	50.8	19.2	47.8	55.4	53.8	50	52.6	56.7	53.1	62.6

Abbreviations: PnP = Pick and Place; Nv = Novel; CB = Cuttingboard; CbBox = Cardboardbox; PM = Placemat; TR = Tray; PL = Plate; TBasket = Tieredbasket; TShelf = Tieredshelf; Cab. = Cabinet; Micro. = Microwave.

Table 10. **Full evaluation on RoboCasa-GR1 [4].** Success rates (%) across all 24 manipulation tasks with the GR1 humanoid robot (29-DoF actions, ego-centric view, 1000 demos/task, 50 rollouts). [†] denotes methods using internet-scale pre-training. Dark gray and light gray backgrounds indicate the best and second-best results in each row, respectively.