NoiseGate: Learning Per-Latent Timestep Schedules as Information Gating in World Action Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

World Action Models (WAMs) are an emerging family of policies that tie robot action generation to future-observation modeling. In this work, we focus on the joint video–action modeling paradigm, where actions and imagined future observations are co-generated along a shared denoising or flow trajectory, so that perception, prediction, and control are coupled within one generative process. Existing WAMs typically realize this paradigm with a Mixture-of-Transformers (MoT), where video and action tokens interact through shared self-attention. This architecture can in principle assign a separate timestep t_f to each predicted latent frame, yet current systems collapse this degree of freedom onto a single shared scalar t. Under the noise-as-masking view of Diffusion Forcing, this shared schedule imposes the unjustified prior that every predicted latent is equally reliable for action generation. We instead view the per-latent schedule as a *learnable information-gating policy*: by changing a latent frame's noise level, the policy modulates the reliability of its Key/Value contribution to the action tokens. We propose **NoiseGate**, which combines independent per-latent timestep sampling during backbone training, a lightweight Gating Policy Network that emits per-latent time increments during denoising, and task-reward optimization that trains the schedule policy without hand-crafted shape priors. Built on a joint video–action MoT backbone, NoiseGate delivers consistent gains on diverse RoboTwin random-scene manipulation tasks.

1 Introduction

World Action Models (WAMs) [1, 2, 3, 4, 5] generalize classical video-language-action policies by modeling the joint distribution $p(\mathbf{v}, \mathbf{a} \mid v_0, l)$ over a chunk of F future latent frames $\mathbf{v} = (v_1, \ldots, v_F)$ and an action chunk $\mathbf{a}$, given the current observation latent v_0 and language instruction l. Among the various WAM designs [1, 2, 6, 7], one prominent and effective family jointly denoises future latents and actions through a diffusion or flow-matching backbone. This paradigm, exemplified by recent systems such as Motus [1] and Fast-WAM [2], unifies perception, prediction, and control along a common denoising trajectory, offering conceptual simplicity and strong empirical efficiency.

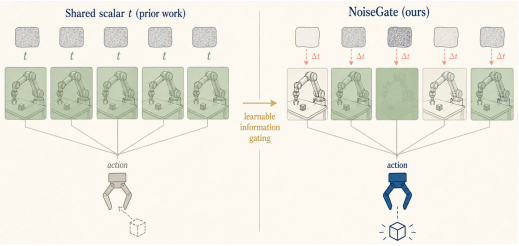

Figure 1: **Method preview.** NoiseGate learns per-latent schedules as task-adaptive information gates in a joint video-action denoising backbone.

The backbones used by common joint video–action WAMs (e.g., chunk-bidirectional Wan 2.2 [8]) are in principle agnostic to how noise levels are distributed across predicted latents: each future latent

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

v_f could in principle carry its own timestep t_f. Yet standard practice [1, 2] forces every predicted latent frame to share a single scalar t, collapsing the per-latent timestep vector $\mathbf{t} \in \mathbb{R}^F$ to a scalar at every denoising step. Diffusion Forcing [9] offers a useful lens: since noise level is a form of partial masking, a shared scalar t encodes the strong prior that every predicted latent frame is equally "visible" to the action generation process. This prior is unmotivated, because the influence of each v_f on predicting the correct action is heterogeneous, task-dependent, and unknown in advance. For example, when the model is uncertain about upcoming critical events, such as grasping or placing, it may be beneficial to keep the corresponding imagined latents blurrier. Any handcrafted shape (e.g., monotonic in the latent-frame f) merely swaps one prior for another, leading to suboptimal action generation.

In this work, we move beyond the fixed shared schedule and reframe the per-latent timestep assignment as a *learnable information-gating policy*. A natural realization of the joint video–action modeling paradigm couples a video DiT with an action-expert DiT through a Mixture-of-Transformers (MoT) [10]: video and action tokens share a single self-attention layer, while each modality retains its own feed-forward expert. This joint self-attention makes the gating interpretation concrete: the noise level on a latent frame directly modulates the reliability of its Key/Value contribution at every denoising step, so $\mathbf{t}$ acts as a bank of continuous gates controlling how much evidence from each future latent propagates into the action tokens (see Fig. 1). The optimal gating pattern is inherently task-dependent and should be learned from data rather than prescribed by a fixed schedule.

We instantiate this view as NoiseGate with three coupled ingredients. First, during training, we sample timesteps independently per latent, borrowing the recipe of Diffusion Forcing [9] but applying it to a chunk-bidirectional (non-causal) backbone, so that the model can handle arbitrary per-latent timestep profiles at inference. Second, a lightweight *Gating Policy Network* (GPN) reads the current latents and their timesteps at each denoising step and emits per-latent increments Δt_f for the future latents $v_{1:F}$. Third, we train the GPN with GRPO [11] against a sparse task-success reward, grounding the gating policy in actual task utility rather than any hand-crafted prior.

Our contributions are summarized as follows:

- We reframe per-latent timestep scheduling in joint video–action WAMs as a learnable information-gating policy over the shared self-attention, and show that the conventional shared-scalar schedule is a strong implicit prior (§3).
- We propose NoiseGate, which realizes this view via independent per-latent timestep sampling, a per-step GPN, and GRPO optimization against task reward, requiring no hand-crafted shape prior (§4).
- On RoboTwin under random-scene conditions, NoiseGate outperforms both shared-scalar and hand-crafted per-latent schedules, confirming that learning the gating policy is essential to exploit the per-latent degree of freedom (§5).

2 Related Work

Video–action generative policies. Diffusion-based policy learning, beginning with Diffusion Policy [12], has been scaled to large visual backbones and language-conditioned VLA systems [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. A recent line of World Action Models (WAMs) [1, 2, 3, 23, 24, 25, 26, 27, 4, 28, 7] goes beyond action-only prediction by co-generating future observations and actions, modeling $p(\mathbf{v}, \mathbf{a} \mid v_0, l)$ rather than only $p(\mathbf{a} \mid v_0, l)$. We focus on the joint video–action denoising/flow-matching variant, exemplified by Motus [1] and Fast-WAM [2], where a video DiT and an action expert interact through shared self-attention in a Mixture-of-Transformers architecture [10]. This design exposes a rarely studied degree of freedom: each predicted latent frame could carry its own diffusion time, yet existing systems usually collapse the whole predicted chunk to one shared scalar timestep.

Per-token noise levels and adaptive diffusion schedules. Diffusion Forcing [9] provides the closest training-time precedent for departing from a shared timestep: it interprets noise level as partial masking and trains causal sequence models with independently sampled per-token noise levels. We borrow this noise-as-masking view and the independent noise sampling recipe, but use them for a different purpose. Our backbone is chunk-bidirectional rather than causal, and independent noise is not used as a rollout mechanism; it is a substrate that makes arbitrary per-latent noise profiles

valid at inference time. Classical samplers such as DDIM [29] and EDM [30] improve the global denoising trajectory, while learned schedule methods such as TPDM [31] predict scalar timesteps for single-modality image generation. In contrast, our scheduler emits a vector $\mathbf{t} = (t_1, \ldots, t_F)$ over future video latents inside a joint video–action backbone, where each t_f controls the reliability of that latent's Key/Value contribution to action tokens.

Reinforcement learning for VLA policies. Recent work [32, 33, 34, 35, 36, 37, 38, 39] has also shown that reinforcement learning can substantially improve VLA policies after supervised imitation. Frameworks such as RLinf-vla [32] and SimpleVLA-RL [35] scale outcome-driven RL, including PPO- or GRPO-style optimization, to large VLA models across simulated manipulation benchmarks and real-robot settings. These methods optimize the action-producing policy itself, typically improving the model's ability to map observations and language instructions to successful action trajectories. Our use of RL is complementary: the pretrained video–action backbone is frozen, and reward optimization is applied only to a lightweight scheduler that controls how future latent frames are revealed to the action tokens during denoising. Thus the learned object is not the VLA action policy directly, but a per-latent information-gating policy inside the WAM's generative process.

Taken together, prior WAMs establish joint video–action generation, Diffusion Forcing motivates independent noise levels as masking, learned diffusion schedulers show that timestep selection can be optimized, and VLA-RL methods demonstrate the value of task-reward fine-tuning for embodied policies. NoiseGate combines these threads in a different setting: it learns a task-reward-optimized, per-latent timestep schedule for a joint video–action denoising backbone.

3 Problem Formulation

This section establishes the notation for video–action generation (§3.1), identifies how the conventional shared-scalar schedule limits the model's ability to handle intra-chunk information density (§3.2), and formulates the per-frame latent scheduling problem (§3.3).

3.1 Preliminaries: Video–Action Generation in WAMs

Notation and setup. Let $\mathbf{v} = \{v_1, \ldots, v_F\} \in \mathcal{V}^F$ denote a chunk of F future latent frames. We take v_0 as the clean current observation, l as the language instruction, and $\mathbf{a} \in \mathcal{A}^H$ as the action chunk co-generated with the video. The model captures the conditional distribution $p_\theta(\mathbf{v}, \mathbf{a} \mid v_0, l)$.

Joint video–action denoising backbone. Current WAMs typically use a diffusion or flow-matching Mixture-of-Transformers [10] to jointly denoise future latents and action tokens. While the architecture nominally admits an independent timestep for every token, standard practice collapses the per-element times into two global scalars:

$$g_\theta\big(\mathbf{v}^{\mathbf{t}_v},\, \mathbf{a}^{t_a},\, \mathbf{t}_v, t_a, v_0, l\big)\,, \quad \text{where} \quad \mathbf{t}_v = [t_1, \ldots, t_F] \in [0, T]^F, \quad t_a \in [0, T]. \tag{1}$$

Crucially, in existing WAMs, even the video timesteps are collapsed such that $t_1 = \cdots = t_F = t$. We re-examine this choice, specifically focusing on the flexibility of $\mathbf{t}_v$ while maintaining a synchronous schedule for action $\mathbf{a}$ to ensure a stable execution trajectory.

3.2 Per-Latent Timesteps as Information Gates

The Hidden Prior of a Shared Scalar t. Setting all $t_f = t$ imposes a strong prior: every future frame and every action token is assumed to be equally "visible" or "reliable" at any given step k. However, due to temporal causal dependencies and varying task complexity, certain future frames are inherently harder to predict or more critical for action grounding than others.

Video as a Gated Memory for Action. In the MoT backbone, video and action tokens interact through shared self-attention. Following the "noise-as-masking" interpretation [9], a higher noise level t_f renders the f-th latent frame less reliable, effectively "gating" its contribution to the action tokens' hidden representations. By allowing $\mathbf{t}_v$ to be heterogeneous, we allow the model to *prioritize* the denoising of specific frames that are most informative for the current action, without needing to alter the action's own denoising pace t_a.

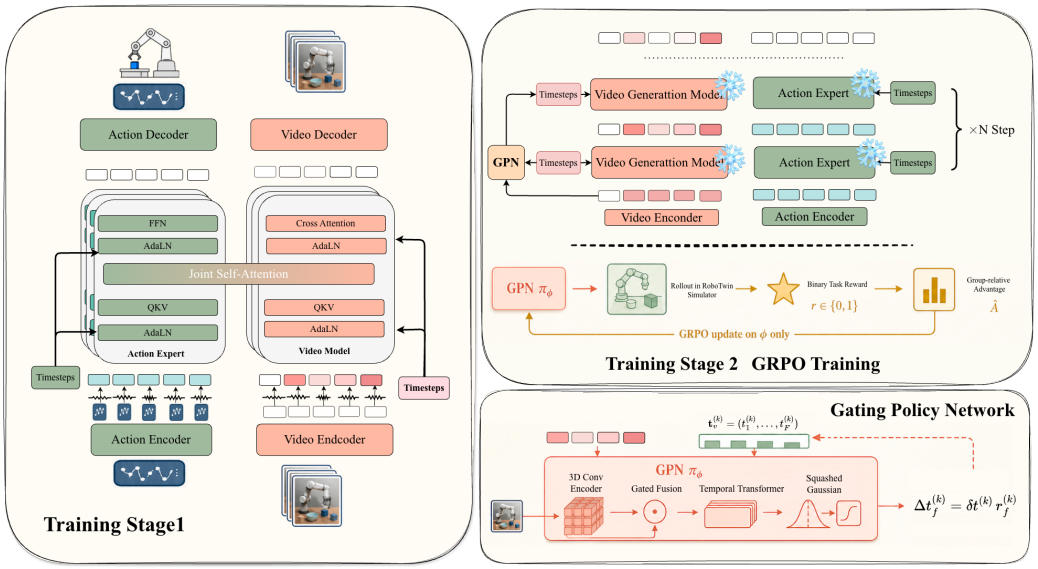

Figure 2: **Overview of NoiseGate.** The unified framework figure summarizes both the joint-sequence MoT backbone and the Gating Policy Network (GPN). At every denoising step, the GPN reads the current predicted-chunk latents and per-latent times, and emits increments Δt_f for $\mathbf{v}$. The observation v_0 is pinned at $t_0 = 0$, and the action follows its own global schedule.

3.3 Learning the Per-Frame Schedule as a Policy

We cast the video scheduling as a policy π_ϕ. At each step k, the policy observes the current state and determines the individual time increments for latent frames only:

$$\Delta \mathbf{t}_v^{(k)} = \pi_\phi\left(\mathbf{v}^{\mathbf{t}_v^{(k)}}, \mathbf{t}_v^{(k)};\ v_0\right). \tag{2}$$

The action schedule t_a follows a fixed linear or cosine decay, acting as a global "clock," while $\mathbf{t}_v$ adapts its trajectory to maximize the task-success reward R.

4 NoiseGate: Per-Latent Schedule as a Learned Gating Policy

NoiseGate turns the per-latent timestep schedule into a first-class learned object. It has three coupled parts: a joint-sequence MoT backbone (§4.1) on which the gating interpretation lives; independent per-latent timestep sampling (§4.2) that makes arbitrary per-latent timestep profiles feasible at inference; and the Gating Policy Network (GPN, §4.3) trained with GRPO (§4.4) to emit per-latent time increments Δt_f at every denoising step. Operationally, these components are trained in two stages: first the WAM backbone is demonstration-finetuned under independent per-latent timesteps, and then the backbone is frozen while the GPN is optimized from task reward. Figure 2 gives an overview.

4.1 World Action Model and MoT Backbone

Following the notation in §3.1, we instantiate NoiseGate in the joint video–action modeling paradigm of World Action Models (WAMs), as shown in Fig. 2. Rather than modeling an action-only policy $p_\theta(\mathbf{a} \mid v_0, l)$, a WAM co-generates future latents and actions under the current observation and language instruction:

$$p_\theta(\mathbf{v}, \mathbf{a} \mid v_0, l), \qquad g_\theta\left(\mathbf{v}^{\mathbf{t}_v}, \mathbf{a}^{t_a}, \mathbf{t}_v, t_a, v_0, l\right). \tag{3}$$

Here $\mathbf{t}_v = (t_1, \ldots, t_F)$ collects per-frame video timesteps, t_a is the action timestep, and superscripts indicate the current noise level. The backbone head g_θ denotes the per-step prediction target of the generative backbone, e.g., diffusion noise prediction or flow-matching velocity prediction. This formulation ties prediction and control to the same denoising/flow trajectory: the future latents provide an explicit imagined context, while the action tokens are refined in the same process.

The backbone realizes this WAM with a Mixture-of-Transformers (MoT) [10] architecture, integrating a chunk-bidirectional video DiT (Wan 2.2-TI2V-5B [8]) with an Action-Expert DiT [40]. The clean observation latent v_0 is pinned at $t_0 = 0$, while predicted latent frames $\{v_f^{t_f}\}_{f=1}^{F}$ and action tokens $\mathbf{a}^{t_a}$ are processed within a joint sequence where all modalities share self-attention blocks but utilize modality-specific feed-forward layers.

As established in §3.2, this shared attention serves as the physical layer for information gating: the individual noise level t_f modulates the reliability of the f-th predicted latent frame's contribution to the action tokens' representations. Notably, while each predicted latent frame v_f is assigned a unique t_f, the action tokens $\mathbf{a}$ share a single, global t_a at any given step, ensuring the backbone learns to extract features from a heterogeneously-noised video context to predict a consistent action chunk.

4.2 Training Substrate: Independent Per-Latent Timestep Sampling

To enable the backbone to generalize to arbitrary per-frame schedules at inference, Stage 1 trains the joint video–action backbone with a per-latent timestep sampling regime inspired by Diffusion Forcing [9].

For each training sample, we draw independent timesteps $\mathbf{t}_v = (t_1, \ldots, t_F)$ and a separate action timestep t_a, construct the heterogeneously noised input $(\mathbf{v}^{\mathbf{t}_v}, \mathbf{a}^{t_a})$, and optimize the backbone with the corresponding diffusion noise target or flow-matching velocity target. This objective ensures the model can effectively "read" through varying levels of uncertainty across the video chunk, making the per-frame schedule a controllable degree of freedom for the policy.

4.3 Gating Policy Network (GPN)

Policy interface. The GPN π_ϕ makes the per-latent schedule a learned control variable. At denoising step k, let $\mathbf{t}_v^{(k)} = (t_1^{(k)}, \ldots, t_F^{(k)})$ denote the current timesteps of the predicted video latents. The policy observes the heterogeneously noised video chunk $\mathbf{v}^{\mathbf{t}_v^{(k)}}$, the clean observation latent v_0, and $\mathbf{t}_v^{(k)}$, and outputs timestep decrements only for the predicted video latents:

$$\Delta \mathbf{t}_v^{(k)} = \pi_\phi\Big(\mathbf{v}^{\mathbf{t}_v^{(k)}},\, \mathbf{t}_v^{(k)}\, v_0\Big), \qquad \Delta \mathbf{t}_v^{(k)} \in \mathbb{R}_+^F. \tag{4}$$

The observation remains pinned at $t_0 \equiv 0$, while the action tokens follow the original global schedule $t_a^{(k)}$ and are not directly controlled by the GPN. This keeps action denoising synchronized for execution while allowing the video context to be selectively unmasked.

Relative-scale action. Rather than predicting unconstrained absolute decrements, the policy is parameterized through bounded relative scales $\mathbf{r}^{(k)} = (r_1^{(k)}, \ldots, r_F^{(k)}) \in (0, 2)^F$. A lightweight spatiotemporal encoder and squashed-Gaussian actor head parameterize this distribution; the layer-wise architecture and log-density are given in Appendix C. Let $\delta t^{(k)}$ denote the nominal time decrement that the original scalar denoising schedule would take at step k. Each component of the decrement vector is

$$\Delta t_f^{(k)} = \delta t^{(k)}\, r_f^{(k)}, \qquad f \in \{1, \ldots, F\}. \tag{5}$$

This parameterization preserves the global pace of the sampler while letting the policy decide which predicted latents should be denoised faster or slower at each step. It also avoids asking the policy to learn both the absolute magnitude and the relative priority of each frame from sparse rollout reward.

Per-latent denoising trajectory. The predicted-video timestep vector is then updated component-wise:

$$t_f^{(k+1)} = \max\{0,\, t_f^{(k)} - \Delta t_f^{(k)}\}, \qquad f \in \{1, \ldots, F\}. \tag{6}$$

We impose no monotonicity or hand-crafted shape prior across frames; the schedule is learned end-to-end from task reward via GRPO (§4.4). Figure 2 shows the GPN in the full framework, and Appendix C gives the inference algorithm.

4.4 Training with GRPO

In Stage 2, we freeze the trained WAM backbone and optimize only the scheduler policy π_ϕ with GRPO [11], which replaces a learned value network by a group-relative baseline. The simulator

provides a binary episodic reward $r \in \{0, 1\}$. For a group of G trajectories sampled under a frozen snapshot $\pi_{\phi_{\text{old}}}$, the advantage is

$$\hat{A}_i \;=\; \frac{r_i - \bar{r}}{\text{std}(\{r_j\}_{j=1}^{G}) + \epsilon}, \qquad \bar{r} \;=\; \tfrac{1}{G}\sum_j r_j. \tag{7}$$

We then perform E epochs of updates on π_ϕ over the same batch, using a per-latent importance ratio

$$\rho_f \;=\; \frac{\pi_\phi(\Delta t_f \mid \mathbf{v}^{\mathbf{t}}, \mathbf{t})}{\pi_{\phi_{\text{old}}}(\Delta t_f \mid \mathbf{v}^{\mathbf{t}}, \mathbf{t})}, \tag{8}$$

and optimize the clipped surrogate with an entropy bonus

$$\mathcal{L}_{\text{GRPO}}(\phi) \;=\; -\,\mathbb{E}\left[\sum_f \min\Big(\rho_f\,\hat{A},\; \text{clip}(\rho_f, 1-\varepsilon, 1+\varepsilon)\,\hat{A}\Big) \;+\; \beta\,\mathcal{H}[\pi_\phi]\right], \tag{9}$$

where $\mathcal{H}[\pi_\phi]$ is the per-step entropy of the Squashed Gaussian and β is decayed exponentially. We omit a KL term since π_ϕ is trained from scratch with no meaningful reference policy; ratio clipping and the entropy bonus suffice for stability. The backbones are frozen; only ϕ is optimized.

5 Experiments

5.1 Setup

Benchmark and evaluation protocol. We evaluate on **RoboTwin** [41] under its random-scene condition, where object poses, colors, and backgrounds are randomized at every episode. We report two complementary studies: a scaled comparison against strong baselines, and a controlled ablation that isolates the effect of per-latent scheduling. All methods within each study use the same task list, backbone configuration, rollout budget, and 100-episode-per-task success metric; Appendix E gives the task-selection details and full 50-task context.

Backbone configurations. All experiments use the same joint video–action WAM/MoT substrate: a Wan 2.2 [8] video DiT coupled with an Action-Expert DiT through shared self-attention. We instantiate it in two training-scale configurations: (i) a Motus-derived configuration with the VLM removed and a batch size of 128, used for controlled ablations and diagnostics, and (ii) a scaled Fast-WAM-style configuration with a batch size of 1024, used for the main performance comparison. This treats Motus-derived and Fast-WAM-style runs as two configurations of the same WAM/MoT family rather than different architectural claims.

Training protocol. Following the two-stage procedure introduced in §4, Stage 1 produces a demonstration-trained WAM backbone, denoted **Stage-1 WAM**, using the official RoboTwin training data. Stage 2 freezes this backbone and trains only the GPN with GRPO. This protocol lets the main comparison, Stage-1 WAM vs. NoiseGate, isolate the effect of learning the per-latent denoising schedule on top of the same video–action backbone.

Baselines. The main comparison evaluates NoiseGate against Stage-1 WAM and representative RoboTwin baselines: Fast-WAM, LingBot-VA, $\pi_{0.5}$, and Motus. The ablation compares against Shared-t, Stage-1 WAM, and Hand-crafted schedule variants that progressively remove the learned schedule policy.

Metric and implementation. We report task success rate averaged over 100 episodes per task under the random-scene condition of RoboTwin. Full GRPO hyperparameters are given in Appendix D, and detailed RoboTwin tables are given in Appendix E.

5.2 Main Results

Table 1 gives the primary RoboTwin random-scene comparison, with the omitted rows expanded in Appendix E. The rightmost block separates the demonstration-tuned Stage-1 WAM from the GRPO-trained NoiseGate policy and reports the absolute gain Δ for each task. Across the displayed tasks, NoiseGate usually preserves already-saturated behavior while improving tasks with remaining headroom: Hanging Mug, Blocks Ranking Size, Put Object Cabinet, and Place Bread Skillet gain

Table 1: **Primary RoboTwin random-scene comparison.** Success rates are percentages over 100 episodes per task. Stage-1 WAM is the scaled WAM backbone after demonstration fine-tuning; NoiseGate is the Stage-2 model after GRPO training of the per-latent gating policy. Best per row in bold; Δ denotes the absolute improvement of NoiseGate over Stage-1 WAM.

Task	Fast-WAM	LingBot-VA	$\pi_{0.5}$	Motus	Stage-1 WAM	NoiseGate (ours)	Δ
Adjust Bottle	**100**	94	99	93	**100**	**100**	0
Beat Block Hammer	97	**98**	93	88	**98**	**98**	0
Blocks Ranking RGB	**100**	98	85	97	97	99	+2
Blocks Ranking Size	**98**	96	26	63	80	87	+7
Dump Bin Bigbin	96	96	97	91	94	**99**	+5
Handover Block	81	78	57	73	81	**84**	+3
Hanging Mug	62	28	17	38	60	**69**	+9
Move Can Pot	88	97	55	74	**100**	**100**	0
Pick Diverse Bottles	85	82	71	**91**	87	89	+2
Pick Dual Bottles	96	99	63	90	99	**100**	+1
Place A2B Left	93	93	82	79	97	**100**	+3
Place A2B Right	**99**	95	84	87	94	98	+4
Place Bread Basket	93	95	64	94	97	**98**	+1
Place Bread Skillet	93	90	66	83	90	**96**	+6
Place Burger Fries	99	95	87	98	96	**100**	+4
			...				
Place Cans Plasticbox	96	99	84	94	97	**100**	+3
Place Container Plate	**100**	97	95	99	99	**100**	+1
Place Dual Shoes	88	89	75	87	**94**	**94**	0
Place Fan	96	93	85	87	95	**98**	+3
Place Mouse Pad	89	**96**	39	68	94	94	0
Place Object Basket	**88**	**88**	76	87	76	79	+3
Place Object Scale	97	95	80	85	**99**	98	-1
Place Object Stand	94	96	85	97	96	**100**	+4
Press Stapler	97	82	83	**98**	94	96	+2
Put Bottles Dustbin	90	91	79	79	90	**93**	+3
Put Object Cabinet	89	87	79	71	87	**94**	+7
Rotate QRcode	89	**91**	87	73	86	86	0
Stack Blocks Three	97	**98**	76	95	96	**98**	+2
Stack Bowls Three	81	83	71	87	**89**	**89**	0
Turn Switch	59	45	54	**78**	76	**78**	+2
Average	91.78	91.50	76.76	87.02	92.58	**94.28**	1.70

Table 2: **Schedule ablation summary** on the controlled RoboTwin random-scene study (overall average success rate). Each row removes one component of NoiseGate.

Configuration	Overall Success (%)
NoiseGate (full)	**67.5**
w/o GPN, hand-crafted per-latent timestep schedule	63.4
w/o GPN (Stage-1 WAM)	61.3
w/o GPN, w/o independent per-latent noise training (Shared-t)	57.5

6–9 points, and many medium-difficulty placement and manipulation tasks gain another 2–5 points. The few zero-gain rows are mostly tasks where Stage-1 WAM is already near the ceiling, while the only small regressions shown are on Place Object Scale and Open Microwave. Relative to the prior RoboTwin baselines, this pattern indicates that the learned per-latent schedule is not merely increasing average performance uniformly; it selectively helps tasks where action generation benefits from task-adaptive control over which predicted video latents are made reliable during denoising.

5.3 Ablation Studies

Table 2 isolates the role of the schedule design; the per-task breakdown is deferred to Appendix E, Table 4. Removing both the GPN and independent-noise training (Shared-t) causes the largest drop (67.5 $\rightarrow$ 57.5), confirming that the per-latent timestep schedule—training substrate plus learned gating—is the primary driver. Keeping the Stage-1 WAM setting (independent per-latent noise training with shared-scalar inference) recovers most of the substrate gain (61.3) but leaves the learned schedule unexploited. Replacing the GPN with a hand-crafted monotone timestep schedule adds a further 2.1 points (63.4), showing that *some* per-latent structure is beneficial—but an arbitrary fixed shape is not sufficient; the schedule must be *task-adaptive* to realize the full gain. Only the full

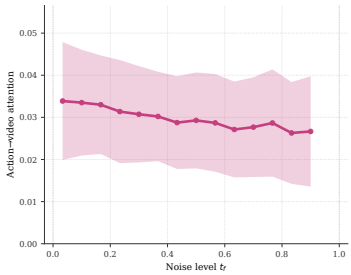

Figure 3: **Noise as masking in the joint self-attention.** Mean action→video attention versus each predicted frame's current noise level t_f. The monotone decay confirms that t_f empirically attenuates each frame's K/V contribution to the action tokens.

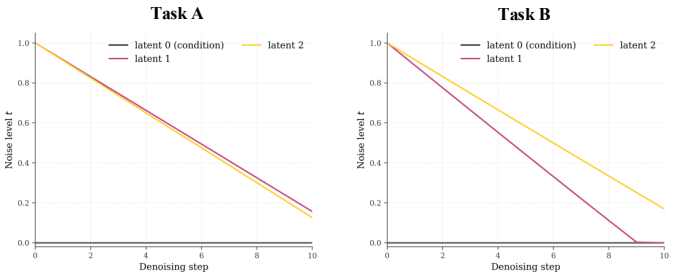

Figure 4: **Task-specific timestep schedules.** Per-frame timestep trajectories for one denoising chunk from two representative tasks. The observation latent is fixed at t_0=0, while the predicted latents follow different task-dependent trajectories. Under the noise-as-masking interpretation, these curves show how the GPN decides which future latents to expose early to the action tokens and which to keep partially masked.

NoiseGate closes the remaining gap to 67.5%, consistent with our framing of the per-latent timestep schedule as a learnable information-gating policy rather than a fixed prior.

5.4 Qualitative Analysis: Gating, Schedules, and Noise-as-Masking

The primary comparison and schedule ablations establish that learning the per-latent timestep schedule helps; we now use two diagnostic views to explain *why* it helps. First, we verify the mechanism predicted by the noise-as-masking interpretation: a noisier latent frame contributes less reliable evidence to the action tokens. Second, we visualize the actual timestep trajectories selected by the GPN for individual chunks from two different tasks, showing that the learned schedule is non-uniform and task-dependent rather than a fixed shared-scalar trajectory. Both probes are taken on the trained NoiseGate policy under the random-scene evaluation protocol.

Noise level acts as a reliability gate. The noise-as-masking view of Diffusion Forcing [9] predicts that, inside the shared self-attention, the noise level t_f on a video token should monotonically attenuate the contribution of its Key/Value projections to the action representations. We test this directly by recording the mean attention from action tokens to each latent frame at every probe step and binning it against that frame's current t_f. As shown in Fig. 3, attention scores exhibit a clear monotone decay, so the per-latent vector $\mathbf{t}$ functions empirically—not just nominally—as a bank of continuous reliability gates.

The GPN uses those gates through task-specific schedules. The attention probe only shows that noise *can* serve as a gate; the scheduler is useful only if it learns when to open or close those gates. Fig. 4 therefore visualizes the full timestep trajectory of a single denoising chunk from two different tasks. This view exposes the policy decision made at every denoising step: the predicted latents separate into distinct trajectories instead of following one shared curve. Interpreted through Fig. 3, a faster drop in t_f means the corresponding future frame is made visible to the action tokens earlier, whereas a slower drop keeps that frame partially masked and prevents unreliable K/V features from dominating the action update. The important point is not merely that the curves are non-identical, but that their ordering and separation change across tasks. This rules out both the shared-scalar sampler, which has no per-frame degree of freedom, and a fixed monotone hand-crafted schedule, which would impose the same ordering on every chunk. The learned GPN instead allocates the denoising budget according to the current task context, selectively trusting the future latents that are useful for the action while suppressing those that remain ambiguous.

Together, these probes connect the controlled ablations in Table 2 to the mechanism of NoiseGate: independent per-latent noise training makes heterogeneous timestep profiles meaningful, and the

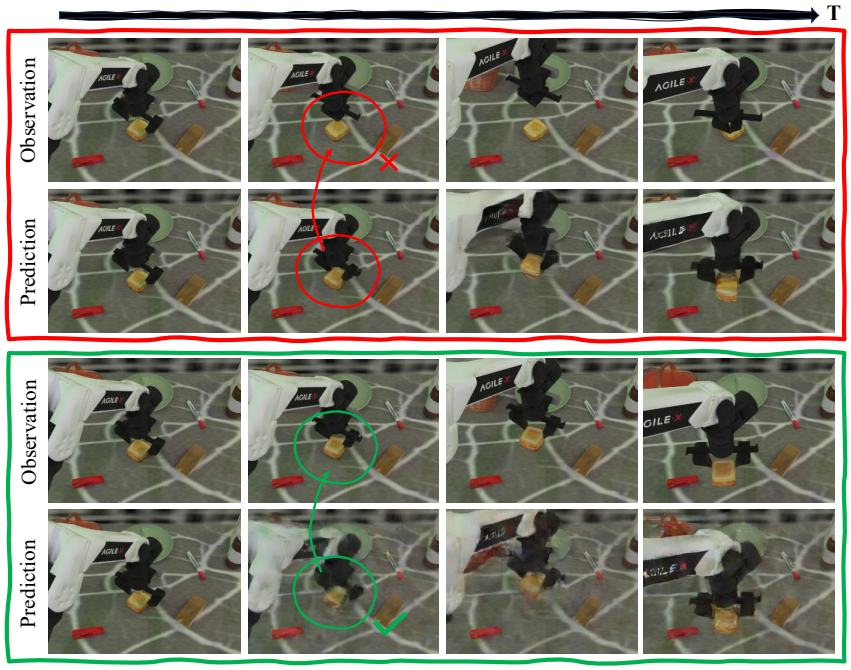

Figure 5: **Visualization of a real test case comparing Stage-1 WAM and NoiseGate**. For each method, the top row shows the true frames after executing the predicted actions, and the bottom row shows the predicted frames. Stage-1 WAM's standard denoising leads to overconfident grasp prediction (second frame) and premature failure, whereas NoiseGate maintains higher uncertainty in the grasp frame through the learnable scheduler, leveraging other predicted frames to achieve successful grasping.

GPN learns to choose those profiles so that predicted latent frames are unmasked only to the extent that they are useful for action generation.

5.5 Case Study

Finally, Fig. 5 presents a representative test case illustrating how the learned scheduler improves action generation over Stage-1 WAM. Each method is shown with two rows: executed observations on top and predicted future frames on the bottom. Stage-1 WAM produces visually clean future predictions, but appears overconfident about the grasp outcome in the second predicted frame, which leads to a premature grasp and failure. In contrast, NoiseGate keeps the critical grasp frame less fully denoised, reflecting higher uncertainty about the contact event. This uncertainty prevents the action tokens from over-relying on an unreliable future latent; instead, the model integrates evidence from the remaining predicted frames and the current observation, enabling a successful grasp at the critical moment.

6 Conclusion

We presented NoiseGate, which reframes the per-latent timestep schedule of a World Action Model as a learnable information-gating policy and optimizes it against task reward. The approach combines independent per-latent noise sampling during training (from Diffusion Forcing, applied to a chunk-bidirectional backbone), a lightweight GPN emitting per-latent time increments at every step, and GRPO grounded in task success, without hand-crafted shape priors. On RoboTwin random-scene evaluation, NoiseGate improves a strong Stage-1 WAM on the primary comparison and yields a +10.0 point gain over the shared-t baseline in controlled schedule ablations, supporting the view that the per-latent timestep schedule in a WAM is a first-class design object rather than an implicit hyperparameter.

References

[1] 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. *arXiv preprint arXiv:2512.13030*, 2025.

[2] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026.

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

[4] Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *arXiv preprint arXiv:2512.06963*, 2025.

[5] 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 preprint arXiv:2601.16163*, 2026.

[6] 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 preprint arXiv:2511.17502*, 2025.

[7] Shuang Li, Yihuai Gao, Dorsa Sadigh, and Shuran Song. Unified video action model. *arXiv preprint arXiv:2503.00200*, 2025.

[8] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[9] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

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

[11] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

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

[14] Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. *arXiv preprint arXiv:2508.21046*, 2025.

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

[16] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[17] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

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

[19] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[20] 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.

[21] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

[22] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[23] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[24] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, et al. Dreamgen: Unlocking generalization in robot learning through video world models. *arXiv preprint arXiv:2505.12705*, 2025.

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

[26] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

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

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

[29] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[30] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[31] Zilyu Ye, Zhiyang Chen, Tiancheng Li, Zemin Huang, Weijian Luo, and Guo-Jun Qi. Schedule on the fly: Diffusion time prediction for faster and better image generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23412–23422, 2025.

[32] Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, et al. Rlinf-vla: A unified and efficient framework for vla+ rl training. *arXiv preprint arXiv:2510.06710*, 2025.

[33] Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *arXiv preprint arXiv:2509.15965*, 2025.

[34] Zhong Guan, Haoran Sun, Yongjian Guo, Shuai Di, Xiaodong Bai, Jing Long, Tianyun Zhao, Mingxi Luo, Chen Zhou, Yucheng Guo, et al. Rl-vla3: Reinforcement learning vla accelerating via full asynchronism. *arXiv preprint arXiv:2602.05765*, 2026.

[35] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025.

[36] Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*, 2025.

[37] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025.

[38] Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

[39] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025.

[40] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

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

A Limitations

The main limitation of the current schedule-policy training is data-collection efficiency in simulation. GRPO uses sparse task-success rewards, so each update depends on simulator rollouts to obtain learning signal; rollout collection remains the dominant cost and can slow convergence when successful trajectories are rare. Learning a general schedule policy that scales across large-scale multi-task suites is therefore left to future work.

B Broader applicability

The view of per-latent noise as a learnable information-gating policy is not specific to robotics: any generative setting in which a downstream consumer reads, via shared attention, from jointly-denoised tokens with heterogeneous utility—video prediction, autonomous driving, embodied navigation—can in principle be framed the same way.

C GPN Architecture and Inference Details

This appendix specifies the Gating Policy Network (GPN) used in §4.3. The goal is to make explicit how the policy maps the current heterogeneously noised video chunk and timestep vector to per-latent scheduling actions.

Interface and notation. Let $F'{=}F{+}1$ denote the observation latent together with the F predicted latents. At denoising step k, the GPN consumes the current latent stack $\mathbf{v}^{\mathbf{t}^{(k)}} \in \mathbb{R}^{B\times C\times F'\times H\times W}$ and per-latent video times $\mathbf{t}^{(k)} \in \mathbb{R}^{B\times F'}$, with $t_0^{(k)} = 0$. It returns a sampled relative scale $\mathbf{r}^{(k)} \in (0,1)^F$, the corresponding updated video times $\mathbf{t}^{(k+1)}$, and the sampled-action log-probability used by GRPO. The action timestep $t_a^{(k)}$ is not an input to the GPN policy head; it is advanced by the fixed global schedule.

Network architecture. The network first converts the latent stack into one token per video latent. Two strided 3-D convolutions with GroupNorm and SiLU reduce the spatial resolution, producing $\mathbf{z} \in \mathbb{R}^{B\times F'\times C'\times H'\times W'}$. For each latent, attention pooling, average pooling, and max pooling over (H', W') are concatenated and projected to a $D{=}256$ dimensional token, yielding $\mathbf{x} \in \mathbb{R}^{B\times F'\times D}$.

The observation token $\mathbf{x}_0$ is used only as conditioning. Predicted-frame tokens $\mathbf{x}_{1:F}$ are first reweighted by a channel-attention MLP, then fused with $\mathbf{x}_0$ through a gated residual update:

$$\mathbf{f}_f \;\leftarrow\; \mathbf{f}_f \;+\; \sigma\big(W_g\,[\mathbf{f}_f;\,\mathbf{x}_0]\big) \;\odot\; W_v\,\mathbf{x}_0. \tag{10}$$

Here $W_g \in \mathbb{R}^{D\times 2D}$ produces the sigmoid gate from $[\mathbf{f}_f; \mathbf{x}_0]$, and $W_v \in \mathbb{R}^{D\times D}$ projects the observation token before the residual add.

The resulting predicted-frame tokens receive a learnable frame-position embedding and pass through a three-layer pre-norm Transformer encoder with eight attention heads and GELU feed-forward blocks of width $4D$. In parallel, the current times $\mathbf{t}_{1:F}^{(k)}$ are embedded with sinusoidal features and a two-layer MLP. The latent token and time embedding for each frame are concatenated, normalized, and projected to produce $\mathbf{h}_f \in \mathbb{R}^D$. An attention-pooling head forms a global summary $\mathbf{g} = \sum_f w_f \mathbf{h}_f$, where $w_f = \mathrm{softmax}_f(\mathrm{MLP}_{\mathrm{score}}(\mathbf{h}_f))$.

Squashed-Gaussian policy. The actor is a three-layer MLP applied to $[\mathbf{g}; \mathbf{h}_1; \ldots; \mathbf{h}_F]$ and outputs $\boldsymbol{\mu} \in \mathbb{R}^F$. The log standard deviation $\log \boldsymbol{\sigma} \in \mathbb{R}^F$ is a learned parameter clamped to $[-5, 2]$. For each predicted latent, the policy samples a pre-squash variable

$$u_f \sim \mathcal{N}(\mu_f, \sigma_f^2), \qquad r_f = 2\sigma(u_f).$$

Equivalently, for an observed relative-scale action $r_f \in (0, 2)$ with $u_f = \mathrm{logit}(r_f)$, the corresponding per-latent log-density is

$$\log \pi_\phi(r_f \mid s^{(k)}) \;=\; \log \mathcal{N}(u_f;\, \mu_f, \sigma_f^2) \;-\; \big(u_f - 2\,\mathrm{softplus}(u_f)\big), \tag{11}$$

where $s^{(k)}$ denotes the GPN input state $(\mathbf{v}^{\mathbf{t}^{(k)}}, \mathbf{t}^{(k)})$. GRPO stores the log-probability of the sampled pre-clipping relative-scale action, $\sum_{f=1}^{F} \log \pi_\phi(r_f \mid s^{(k)})$. Since $\Delta t_f^{(k)} = \delta t^{(k)} r_f$ differs from r_f only by a fixed step-dependent scale, the additional log-Jacobian term cancels in the new/old policy ratio for a fixed denoising step.

Inference loop. Algorithm 1 summarizes how the GPN is invoked inside one denoising step. The algorithm is intentionally written at the scheduling level; the architectural mapping from latents and times to $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ is the network described above.

Algorithm 1 Per-step GPN scheduling inside the denoising loop

Require: Current latents $\mathbf{v}^{\mathbf{t}^{(k)}}$; video times $\mathbf{t}^{(k)}$ with $t_0^{(k)} = 0$; scalar-schedule decrement $\delta t^{(k)}$; action time $t_a^{(k)}$.

Ensure: Updated video times $\mathbf{t}^{(k+1)}$; next action time $t_a^{(k+1)}$; sampled-action log-probability.

1: Encode $(\mathbf{v}^{\mathbf{t}^{(k)}}, \mathbf{t}^{(k)})$ with the GPN to obtain $(\boldsymbol{\mu}, \boldsymbol{\sigma})$.
2: Sample $u_f \sim \mathcal{N}(\mu_f, \sigma_f^2)$ and set $r_f \leftarrow \sigma(u_f)$ for $f = 1, \ldots, F$.
3: Set $\Delta t_f^{(k)} \leftarrow 2\,\delta t^{(k)} r_f$ for $f = 1, \ldots, F$.
4: Update $t_f^{(k+1)} \leftarrow \max(0, t_f^{(k)} - \Delta t_f^{(k)})$ for $f = 1, \ldots, F$; keep $t_0^{(k+1)} \leftarrow 0$.
5: Advance $t_a^{(k+1)}$ using the fixed global action schedule.
6: Run the frozen MoT backbone with $(\mathbf{t}^{(k+1)}, t_a^{(k+1)})$ to produce the next denoising state.
7: Store $\sum_{f=1}^{F} \log \pi_\phi(r_f \mid s^{(k)})$ for the GRPO update.

During GRPO training, only the GPN parameters ϕ are optimized. The Wan 2.2 video DiT and Action-Expert DiT backbone remain frozen throughout the rollout collection and policy update.

D GRPO Hyperparameters

Table 3 lists the GRPO training hyperparameters used in all experiments.

Table 3: **GRPO hyperparameters.**

Hyperparameter	Value
Group size G	8
Clip range ε	0.2
Inner epochs E	4
Optimizer	AdamW
Learning rate	1×10^{-4}
Gradient-norm clip	1.0
Entropy coefficient β_0	0.01
Entropy decay rate	0.999 per step (floored at 0)
Denoising steps K	10
Training epochs	50
Episodes per epoch	8
Rollout workers (GPUs)	8
Seed	42

E Detailed RoboTwin Results

The controlled ablation uses the Motus-derived batch-128 configuration to support exhaustive component comparisons and diagnostics; the primary comparison uses the scaled Fast-WAM-style batch-1024 configuration. Table 4 gives the per-task breakdown behind the schedule ablation summary in Table 2. Table 5 provides the full 50-task RoboTwin random-scene context. Unavailable full-method entries are marked with "–".

Table 4: **Per-task schedule ablation on RoboTwin random-scene tasks (100 episodes each).** All variants in this table use the same controlled Motus-derived backbone configuration and evaluation protocol. "Shared-t" is the shared-scalar baseline; "Stage-1 WAM" uses independent per-latent noise training but denoises with a shared scalar at inference; "Hand-crafted" uses a fixed monotone per-latent timestep schedule; "NoiseGate" learns the per-latent timestep schedule via GRPO. Best per row in bold.

Task	Shared-t	Stage-1 WAM	Hand-crafted	NoiseGate (Ours)
Adjust Bottle	0.88	0.85	0.85	**0.94**
Open Microwave	0.87	0.86	0.84	**0.91**
Move Playingcard Away	0.86	0.87	0.90	**0.91**
Place Bread Basket	0.75	0.82	0.87	**0.92**
Place A2B Right	0.75	0.68	0.70	**0.78**
Beat Block Hammer	0.76	0.70	0.73	**0.82**
Blocks Ranking RGB	0.72	0.76	0.73	**0.82**
Place Fan	0.56	0.57	0.60	**0.62**
Lift Pot	0.51	0.56	0.51	**0.57**
Place Mouse Pad	0.50	0.49	0.52	**0.54**
Place Dual Shoes	0.36	0.46	0.51	**0.59**
Place Can Basket	0.40	0.44	0.44	**0.49**
Move Pillbottle Pad	0.32	0.47	0.50	**0.51**
Move Can Pot	0.21	0.27	0.30	**0.32**
Pick Diverse Bottles	0.11	0.34	0.39	**0.39**
Average	0.575	0.613	0.634	**0.675**

F Additional Qualitative Analyses

This appendix collects additional diagnostic views that complement the two qualitative probes in §5.4. They provide layer-stratified attention statistics and broader schedule summaries beyond the compact main-text visualization.

F.1 Mechanism: layer-stratified action→video attention vs. noise level

Fig. 3 reports the mechanism aggregated across the recorded probe layers. Fig. 6 stratifies the same scatter into three layer groups (early / middle / late). The monotone decay of action→video attention with respect to t_f is preserved in the early- and late-layer groups, while the middle-layer group is near-flat. The mechanism is therefore not an artifact of any single probe layer, but it is also non-uniform across depth: the gating effect is strongest where the action tokens directly read out video features (early and late layers), and weakest in the intermediate layers, where representations are already largely modality-agnostic.

F.2 Cross-task structure of final-step residual noise

Fig. 7 further examines the timestep schedule learned by the GPN from a task-level perspective. For each task, we collect all evaluated chunks and report the mean final-step noise level $\bar{t}$ of each predicted future frame, with horizontal bars denoting ± 1 standard deviation. The conditioning frame is omitted because it is fixed at $t_0 \equiv 0$. Tasks are sorted by their success rate, so the figure jointly shows task difficulty and the residual masking pattern selected by the learned schedule.

The learned policy does not simply drive all predicted frames to the same clean endpoint. Instead, the two future frames exhibit distinct residual-noise levels, and the gap is task-dependent. In several tasks, frame 1 is almost fully denoised while frame 2 retains a larger residual t, indicating that the policy chooses to keep the farther future latent partially masked at the end of action generation. The effect is especially visible on tasks such as `place_a2b_right` and `place_can_basket`, where the uncertainty over future contact or placement outcomes is higher. This supports the information-gating interpretation: the GPN learns when a predicted latent should contribute as reliable evidence and when it should remain attenuated rather than forcing every latent through a shared scalar schedule.

Table 5: **Full 50-task RoboTwin random-scene context.** All entries are success rates in percent. The Fast-WAM-family and prior-method columns are adapted from the corresponding RoboTwin detail table; the Stage-1 WAM column uses our 50-task random-scene evaluation. "–" denotes full-method entries not reported.

Task	**Fast-WAM**	**Fast-WAM-Joint**	**Fast-WAM-IDM**	**LingBot-VA**	$\pi_{0.5}$	**Motus**	**Stage-1 WAM**	**NoiseGate**
Adjust Bottle	100	99	99	94	99	93	100	100
Beat Block Hammer	97	98	98	98	93	88	98	98
Blocks Ranking RGB	100	100	99	98	85	97	97	99
Blocks Ranking Size	98	91	90	96	26	63	80	87
Click Alarmclock	100	100	100	100	89	100	100	100
Click Bell	100	98	96	100	66	100	100	100
Dump Bin Bigbin	96	95	98	96	97	91	94	99
Grab Roller	100	100	100	100	100	100	100	100
Handover Block	81	91	94	78	57	73	81	84
Handover Mic	100	100	99	96	97	63	100	100
Hanging Mug	62	56	62	28	17	38	60	69
Lift Pot	100	100	100	99	85	99	99	99
Move Can Pot	88	99	100	97	55	74	100	100
Move Pillbottle Pad	99	100	100	99	61	96	100	100
Move Playingcard Away	100	100	100	99	84	96	100	100
Move Stapler Pad	64	81	85	79	42	85	81	85
Open Laptop	100	92	92	94	96	91	99	99
Open Microwave	45	14	53	86	77	91	77	69
Pick Diverse Bottles	85	87	89	82	71	91	87	89
Pick Dual Bottles	96	99	98	99	63	90	99	100
Place A2B Left	93	96	96	93	82	79	97	100
Place A2B Right	99	95	98	95	84	87	94	98
Place Bread Basket	93	94	97	95	64	94	97	98
Place Bread Skillet	93	93	95	90	66	83	90	96
Place Burger Fries	99	100	99	95	87	98	96	100
Place Can Basket	69	23	28	84	62	76	58	62
Place Cans Plasticbox	96	98	96	99	84	94	97	100
Place Container Plate	100	98	96	97	95	99	99	100
Place Dual Shoes	88	89	87	89	75	87	94	94
Place Empty Cup	100	100	100	100	99	98	100	100
Place Fan	96	96	95	93	85	87	95	98
Place Mouse Pad	89	91	93	96	39	68	94	94
Place Object Basket	88	81	82	88	76	87	76	79
Place Object Scale	97	99	99	95	80	85	99	98
Place Object Stand	94	98	100	96	85	97	96	100
Place Phone Stand	99	100	99	97	81	86	98	98
Place Shoe	99	97	98	98	93	97	97	99
Press Stapler	97	50	57	82	83	98	94	96
Put Bottles Dustbin	90	95	92	91	79	79	90	93
Put Object Cabinet	89	90	90	87	79	71	87	94
Rotate QRcode	89	92	86	91	87	73	86	86
Scan Object	92	92	90	91	65	66	87	92
Shake Bottle	100	100	100	97	97	97	100	100
Shake Bottle Horizontally	100	100	100	99	99	98	100	100
Stack Blocks Three	97	97	95	98	76	95	96	98
Stack Blocks Two	100	100	100	98	100	98	100	100
Stack Bowls Three	81	86	83	83	71	87	89	89
Stack Bowls Two	98	95	96	98	96	98	99	100
Stamp Seal	94	99	94	97	55	92	96	97
Turn Switch	59	72	74	45	54	78	76	78
Average	91.78	90.32	91.34	91.50	76.76	87.02	92.58	94.28

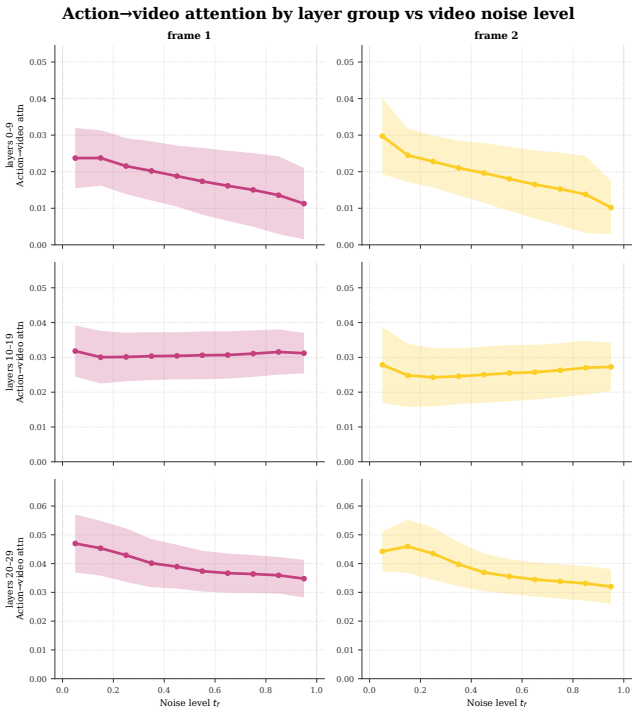

Figure 6: **Layer-stratified version of Fig. 3.** Same binned-mean ± 1 std curve of action→video attention vs. frame noise level t_f, separated into three layer groups (early: layers 0–9; middle: 10–19; late: 20–29). The gating effect is concentrated in the early and late layers, with the middle layers behaving near-uniformly in t_f.

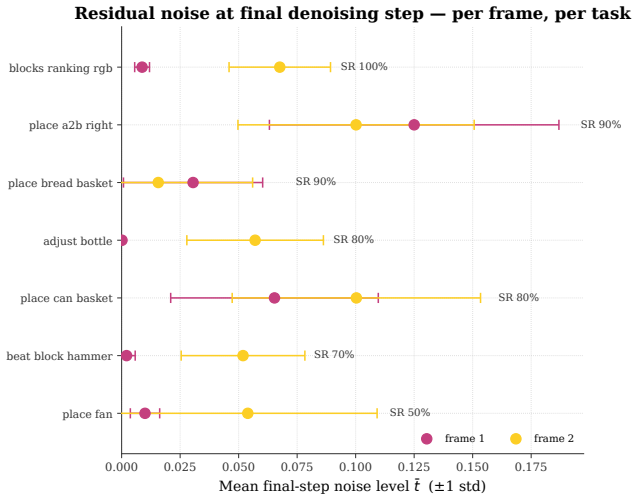

Figure 7: **Final-step residual noise by task and future-frame index.** For each RoboTwin task, dots show the mean final-step noise level $\bar{t}$ of each predicted future frame over all evaluated chunks, and horizontal bars show ± 1 standard deviation. Tasks are sorted by success rate. The non-uniform, task-dependent residuals show that the learned GPN does not collapse to a shared denoising endpoint; instead, it selectively leaves some future latents partially masked when their contribution should be attenuated.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Yes, the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discussed the limitations in Appendix.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: This paper does not present any new theoretical results; all content consists of algorithm design and experimental verification, and therefore does not involve theorems or proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper details all key hyperparameters, model architectures, training procedures, and evaluation protocols. The code will be open-sourced after publication to further support reproducibility.

Guidelines:

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

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

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

Answer: [No]

Justification: The code is not yet publicly available at the time of submission, but it will be open-sourced upon publication. The paper provides detailed algorithmic descriptions and hyperparameters to aid reproducibility.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: The paper specifies training details including data splits, hyperparameters, optimizer settings, and selection criteria in Section 5 and Appendix.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: The paper reports average performance without error bars or statistical significance measures, as repeated runs are computationally prohibitive.

Guidelines:

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

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

8. **Experiments compute resources**

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

Answer: [Yes]

Justification: The paper reports the number of GPUs, providing sufficient details for reproduction.

Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The research uses only public datasets and standard benchmarks, involves no human subjects or sensitive data, and does not pose any foreseeable ethical or societal harms. All models are intended for embodied AI research with no malicious applications.

Guidelines:

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

10. **Broader impacts**

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

Answer: [N/A]

Justification: There is no societal impact of the work performed.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The proposed model is for embodied AI and robotics control, not for general-purpose content generation, and poses no high-risk misuse potential. Therefore no special safeguards are needed.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All datasets, pretrained models, and code libraries used are properly cited with their original papers, and licenses are acknowledged where applicable.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets (code, data, or models) at this time; all documentation is provided within the paper itself. No external assets are distributed alongside the submission.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve any crowdsourcing or research with human subjects; all experiments are conducted on standard robotic benchmarks and synthetic data.

    Guidelines:

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

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

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

    Answer: [N/A]

Justification: The paper does not involve any research with human subjects, so no IRB approval or risk disclosure is applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: The core method of this research does not involve LLMs as an important, original, or non-standard component. LLMs were used only for writing and editing assistance, which does not require declaration.

Guidelines:

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