# IS TEST-TIME VIDEO GENERATION UNNECESSARY? ADAPTIVE VIDEO REASONING FOR WORLD ACTION MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Recent World Action Models (WAMs) remove the future video prediction at test time, which leads to lower inference cost. This raises a question: is test-time video generation unnecessary? We empirically find that removing it can easily degrade control performance, while its benefit varies substantially across decision states and most of the performance gains can be achieved with limited predictive computation. Therefore, we propose ThinkWAM, a world action model with adaptive predictive reasoning, equipped with a router that dynamically determines whether and how much future prediction to perform for each decision state. To support adaptive predictive reasoning, we formulate a multi-intensity reasoning training scheme that enables one WAM to support both direct action prediction and predictive reasoning at different intensities. We then train the router in two stages. First, it learns the benefit of predictive reasoning; second, it learns a cost-aware allocation over reasoning intensities by balancing predicted control quality and computation cost. At test time, the router determines whether predictive reasoning is needed and, when necessary, selects an appropriate reasoning intensity. ThinkWAM achieves success rates of 98.8%, 80.5%, and 77.7% on LIBERO, LIBERO-Plus, and RoboCasa GR1, respectively and 73.8% on real-world tasks. With predictive reasoning applied to only 28% of decision states, ThinkWAM recovers up to 68% of the success-rate gain achieved by reasoning at every state.

## 1 INTRODUCTION

World Action Models (WAMs) (Yuan et al., 2026; Bi et al., 2026; Li et al., 2026a; Team et al., 2026; Su et al., 2026) extend robot policies by jointly modeling future videos and robot actions. Unlike Vision-Language-Action models (VLAs) (Intelligence et al., 2025; Kim et al., 2025), WAMs leverage predicted future dynamics to improve action generation in manipulation tasks. However, future prediction at test time introduces additional denoising computation, creating a trade-off between predictive modeling and inference efficiency. Recent work (Yuan et al., 2026) improves efficiency by retaining future prediction as a training objective while removing it at inference. This raises an important question: **can future prediction be removed uniformly at test time, or does its value depend on the current decision state?**

Rather than treating future prediction as either always on or always off, we consider whether its computation can be allocated adaptively across decision states. This requires answering two questions: whether test-time video is beneficial for the current decision and, if so, how much computation should be used. Since the action can attend to future-video latent at test time, we view test-time future prediction as a form of reasoning in WAMs, turning it from a fixed component into a controllable process along two dimensions: whether to reason and at what intensity.

However, recent adaptive reasoning methods mainly focus on routing between discrete modalities, such as textual and visual reasoning (Tang et al., 2026) with an additional VLM. In contrast, We study a complementary form of adaptivity within the WAM backbone itself: whether reasoning is needed or how much reasoning computation should be allocated to each state.

To this end, we propose ThinkWAM, a world action model with adaptive predictive reasoning. We first train the WAM backbone with a multi-intensity reasoning training scheme, enabling a single



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




Figure 1: **ThinkWAM adaptively allocates predictive reasoning across decision states.** Unlike fixed strategies, our router determines whether reasoning is needed and selects the appropriate reasoning intensity based on state-specific quality-cost trade-offs. Most of the reasoning benefit is obtained early: using reasoning for only 28% of states recovers $\sim$ 68% of the gains.

model to support both direct action prediction and predictive reasoning at different intensities. To determine whether predictive reasoning is needed, we then train a router to estimate the benefit of predictive reasoning for each decision state. The supervision is obtained by comparing generated action with and without predictive reasoning, allowing the router to identify states where predictive reasoning is useful. To further determine how much reasoning should be performed, we model how control quality varies across reasoning intensities and learn a cost-aware allocation policy with GRPO (Shao et al., 2024; Guo et al., 2025) that selects suitable reasoning computation according to the quality-cost trade-off. At inference time, the router first decides whether predictive reasoning is necessary and, if so, selects an appropriate reasoning intensity for the current state; otherwise, the model directly predicts the action.

Our experiments reveal two findings that characterize the role of predictive reasoning at test time. Completely removing future prediction leads to a clear drop in task success, whereas a relatively small amount of predictive computation recovers most of the performance gain and additional computation brings smaller returns. Moreover, the benefit of future prediction varies substantially across decision states: some states require little or no predictive reasoning, whereas others benefit from more predictive computation. On RoboCasa GR1, for example, reasoning on only 28% of decision states recovers approximately 68% of the success-rate gain obtained by reasoning at every state. Across LIBERO, LIBERO-Plus, and RoboCasa GR1, ThinkWAM achieves strong performance while substantially reducing unnecessary predictive computation, and the same adaptive reasoning strategy consistently improves real-world manipulation performance.

Our contributions are summarized as follows:

- We revisit whether test-time video should be removed, and identify two key properties: its benefit varies across decision states, while most gains can be achieved with limited predictive computation. These findings motivate adaptive predictive reasoning.
- We propose ThinkWAM, a world action model with adaptive predictive reasoning, which dynamically determines whether future predictive reasoning is needed and, if so, how much computation should be allocated for each decision state.
- We conduct extensive experiments across LIBERO, LIBERO-Plus, RoboCasa GR1, and real-world manipulation. ThinkWAM achieves strong performance across benchmarks while substantially reducing unnecessary predictive computation.

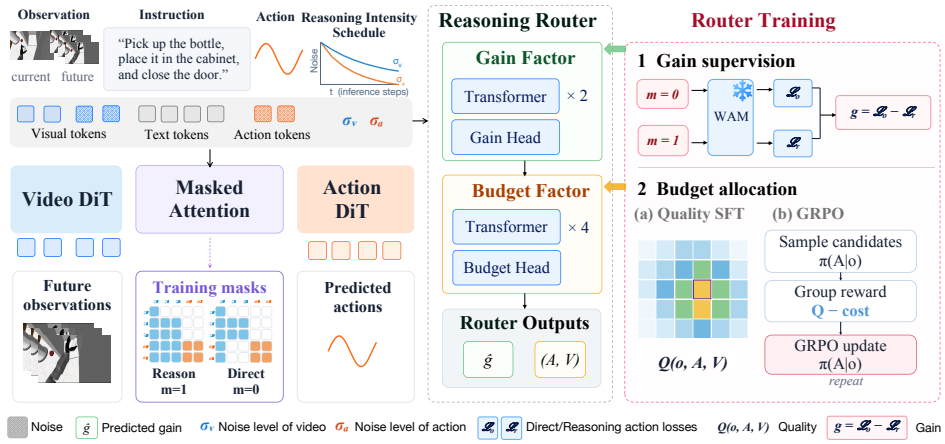


Figure 2: Overview of ThinkWAM. A multi-intensity reasoning WAM supports both direct action prediction and predictive reasoning. The router first decides whether to reason and then selects its reasoning intensity through a learned quality-cost trade-off.

## 2 Related Work

### 2.1 World Action Models

World Action Models (WAMs) (Yuan et al., 2026; Bi et al., 2026; Li et al., 2026a; Team et al., 2026; Su et al., 2026; Kim et al., 2026) extend robot policies by modeling future visual states and actions, allowing action generation to leverage predicted future information. Existing WAMs differ in how future representations are integrated, either through joint video-action denoising (Guo et al., 2026; Ye et al., 2026) or future-conditioned action prediction (Bi et al., 2026; Li et al., 2026a; Team et al., 2026; Su et al., 2026). However, they perform future prediction at inference, incurring video denoising cost regardless of the current state. More recently, Fast-WAM (Yuan et al., 2026) removes inference-time future generation, but whether future prediction should always be removed remains unclear. In contrast, we introduce adaptive predictive reasoning, which learns when predictive reasoning is needed and how much computation to allocate.

### 2.2 Test-Time Reasoning and Adaptive Computation

Recent studies (Snell et al., 2024; Geiping et al., 2025; Guo et al., 2025) on large language models show that test-time computation can improve performance on reasoning tasks, demonstrating that computation can be allocated as a flexible resource. In robotics, recent works (Zhao et al., 2025; Chen et al., 2026a; Zhong et al., 2026) explore reasoning in vision-language-action policies through additional intermediate steps. More recently, AdaWAM (Tang et al., 2026) dynamically chooses textual or visual modality to reason. While it focuses on which reasoning modality to select, we study a complementary question for the reasoning process intrinsic to the WAM: whether predictive reasoning is needed and at what reasoning intensity should be allocated to each decision state. In WAMs, such predictive reasoning is coupled with the denoising process. We exploit this structure to train a single backbone across multiple future-latent refinement levels, and learn a state-dependent quality-cost trade-off to allocate predictive computation adaptively.

## 3 Method

### 3.1 Overview

ThinkWAM predicts an action chunk from the current multi-view observation, language instruction, and robot state while adapting inference computation to each input. We first instantiate a WAM that jointly models future visual dynamics and robot actions. We train the backbone to support both direct action prediction and predictive reasoning at multiple intensities, where different intensities

provide future latent representations at different refinement levels. We then freeze the WAM and train an adaptive reasoning router, which first determines whether predictive reasoning is needed and then selects an appropriate reasoning intensity for each state. At inference, ThinkWAM either predicts actions directly or performs latent future reasoning at the selected intensity.

## 3.2 World-Action Model for Adaptive Predictive Reasoning

### 3.2.1 World-Action Model

The World-Action Model (WAM) is built as a mixture-of-transformers with two diffusion transformer experts: a video branch initialized from Wan2.2-5B (Wan et al., 2025) for future-state prediction and an action branch for action generation. With the current observation, language instruction, and robot state, the video branch refines future latent representations that are exposed to the action branch through masked attention. The two experts maintain separate denoising processes, enabling different video-action computation configurations. During training, an attention mask controls whether the action branch can attend to future-video latents, enabling the same backbone to learn both direct action prediction and latent future reasoning. The future latents serve as intermediate reasoning representations without pixel-space decoding.

### 3.2.2 Training Objective

Both video and action branches are trained with rectified flow matching. Given clean data $x$ and noise $\epsilon$, the noisy input is $\tilde{x}_\sigma = (1-\sigma)x + \sigma\epsilon$, with velocity target $\epsilon - x$.

Let $v_v^\star$ and $v_a^\star$ denote the target velocities for video prediction and action generation. The backbone is optimized by

$$\mathcal{L}_{\text{wam}} = \lambda_v \mathbb{E}\left[\|v_v - v_v^\star\|_2^2\right] + \lambda_a \mathbb{E}\left[\|v_a - v_a^\star\|_2^2\right]. \tag{1}$$

During training, we sample $m \sim \text{Bernoulli}(0.5)$: $m = 1$ enables attention to future-video latents for predictive reasoning, while $m = 0$ masks this access for direct action prediction. We then extend the reasoning route to multiple intensities. See App. A.1 for details.

### 3.2.3 Multi-Intensity Predictive Reasoning Training

Beyond switching between direct action prediction and predictive reasoning, adaptive reasoning requires the same backbone to operate at different reasoning intensities. We denote a reasoning intensity by $(A, V)$, where $A$ is the number of joint updates used for action denoising and $V$ controls the refinement rate of the future-video latents through the video denoising schedule. At inference, the reasoning path terminates after $A$ joint updates; the remaining future latents are not further refined or decoded into pixels. Therefore, different $(A, V)$ pairs determine how much refined future information is available to action generation. To train a single backbone across these intensities, we expose it to different video-action refinement relationships using a mixture of synchronous and asynchronous noise schedules. Let $\sigma_a, \sigma_v \in [0, 1]$ denote the action and video noise levels. Both schedules use the shifted rectified-flow clock:

$$\phi_s(u) = \frac{s\,u}{1 + (s-1)u}. \tag{2}$$

For synchronous samples, drawn with probability $1 - p_{\text{async}}$, both streams share the same clock: $\sigma_a = \phi_{s_a}(u)$ and $\sigma_v = \phi_{s_v}(u)$, where $u \sim \mathcal{U}(0, 1)$. For asynchronous samples, inspired by Guo et al. (2026) and drawn with probability $p_{\text{async}}$, we first sample $\sigma_a = \phi_{s_a}(u_a)$ with $u_a \sim \mathcal{U}(0, 1)$, and then sample $\sigma_v \sim \mathcal{U}(\sigma_a, 1)$. Together, these samples expose the action policy to future representations at different refinement stages, enabling the same backbone to execute the family of reasoning intensities selected by the router at inference. See App. A.2 for more details.

## 3.3 Data Generation Pipeline

We construct two supervision signals from the frozen WAM for the two routing decisions: reasoning gain for gating and intensity quality for budget allocation. Let $x = (o, \ell, p)$ denote the routing state, consisting of the observation, language instruction, and proprioceptive state.

For gating supervision, we compare action predictions with and without predictive reasoning under matched conditions, isolating the contribution of predictive reasoning. Given a routing state-action pair $(x, a)$, we define the reasoning gain as

$$g(x) = \mathbb{E}_{\text{seed}}\left[L_{\text{o}}(x, a) - L_{\text{r}}(x, a)\right], \tag{3}$$

where $L_{\text{o}}$ and $L_{\text{r}}$ are the action prediction errors without and with reasoning, respectively. Positive $g(x)$ indicates that predictive reasoning improves action prediction.

For intensity supervision, we evaluate the frozen WAM at each reasoning intensity $(A, V)$ using

$$L(x, A, V) = \text{SmoothL1}\big(\hat{a}(x; A, V), a\big). \tag{4}$$

The resulting errors are normalized into quality targets $Q(x, A, V)$, which characterize the state-dependent quality landscape over reasoning intensities. See App. A.3 for more details.

### 3.4 Adaptive Reasoning Router

The adaptive reasoning router makes two decisions: whether predictive reasoning is needed and what reasoning intensity to use. It reuses the frozen WAM representation with lightweight modules, where the gating factor estimates reasoning benefit and the budget factor selects the reasoning intensity through a learned quality-cost trade-off.

#### 3.4.1 Gating Factor

The gating factor estimates the benefit of enabling latent future reasoning for the current observation. It takes the embedding of routing state $x$ as input, and predicts the reasoning gain $\hat{g}(x)$ through a Transformer encoder and MLP.

The gating factor is trained with a combination of regression and ranking objectives:

$$\mathcal{L}_{\text{gate}} = \text{SmoothL1}\left(\frac{\hat{g}(x)}{\sigma_g}, \frac{g(x)}{\sigma_g}\right) + w\,\mathcal{L}_{\text{rank}}, \tag{5}$$

where $\sigma_g$ is a fixed robust scale computed from the training set. The ranking loss preserves the relative ordering of reasoning gains:

$$\mathcal{L}_{\text{rank}} = \frac{\sum_{(i,j)\in\mathcal{P}} c_{ij}\ \text{softplus}\left(-\text{sign}(g_j - g_i)\frac{\hat{g}_j - \hat{g}_i}{\sigma_g}\right)}{\sum_{(i,j)\in\mathcal{P}} c_{ij}}, \qquad c_{ij} = \min\left(\frac{|g_j - g_i|}{\sigma_g}, C_{\max}\right), \tag{6}$$

where $\mathcal{P}$ denotes sample pairs with sufficiently different reasoning gains. The regression term estimates the reasoning benefit, while the weighted ranking term emphasizes pairs with larger gain differences and improves state ordering for routing.

#### 3.4.2 Budget Factor: Quality Training

When predictive reasoning is activated, the budget factor selects the reasoning intensity by modeling its state-dependent quality landscape. We predict the quality at each $(A, V)$ and center both the target and prediction within the intensity space:

$$\begin{aligned} &\hat{Q}(x, A, V) = f_Q\big(e(x), e_A(A), e_V(V)\big), \\ &\widetilde{Z}(x, A, V) = Z(x, A, V) - \frac{1}{|\mathcal{C}|}\sum_{(A', V')\in\mathcal{C}} Z(x, A', V'), \qquad Z \in \{Q, \hat{Q}\}. \end{aligned} \tag{7}$$

Here, $e(x)$ denotes the routing-state embedding and $e_A, e_V$ are intensity embeddings. The centering focuses the predictor on the relative quality of different reasoning intensities for the same state. The predictor is trained with

$$\mathcal{L}_Q = \frac{1}{|\mathcal{C}|}\sum_{(A,V)\in\mathcal{C}} \text{Huber}\big(\widetilde{\hat{Q}}(x, A, V), \widetilde{Q}(x, A, V)\big) + \lambda_V \mathcal{L}_V, \tag{8}$$

**Algorithm 1** One Control Step of Adaptive Inference

**Input:** routing state $x_t = (o_t, \ell, p_t)$, router, quality predictor, threshold $\tau$, direct steps $N_a$

$e_t \leftarrow \mathrm{Encode}(x_t)$; $\hat{g} \leftarrow \mathrm{Gate}(e_t)$
**if** $\hat{g} \le \tau$ **then** ▷ direct prediction
  **for** $i = 0, \dots, N_a - 1$ **do**
    Update action stream at $\sigma_a^{(i)} = \phi_{s_a}(1 - i/N_a)$
  **end for**
**else** ▷ predictive reasoning
  $A_t^* \leftarrow \arg\max_A \pi(A \mid e_t)$; $V_t^* \leftarrow V_{A_t^*}^*$
  **for** $i = 0, \dots, A_t^* - 1$ **do**
    $\sigma_a^{(i)} \leftarrow \phi_{s_a}(1 - i/A_t^*)$; $\sigma_v^{(i)} \leftarrow \phi_{s_v}(1 - i/V_t^*)$
    Jointly update video and action streams at $(\sigma_v^{(i)}, \sigma_a^{(i)})$; enforce the first-frame condition
  **end for**
**end if**
**return** denoised action chunk

where $\mathcal{L}_V$ further preserves the relative quality profile along $V$ for each fixed $A$:

$$\mathcal{L}_V = \frac{1}{k-1} \sum_{A=2}^{k} \frac{1}{|\mathcal{S}_A|} \sum_{x \in \mathcal{S}_A} \mathrm{Huber}_\beta \left( \frac{\hat{q}^c_{x,A}}{\rho_x(A)}, \frac{q^c_{x,A}}{\rho_x(A)} \right), \tag{9}$$

where $q^c_{x,A}$ and $\hat{q}^c_{x,A}$ denote the target and predicted quality profiles, respectively, after centering over $V$ for a fixed $A$. $\mathcal{S}_A$ selects states with non-trivial $V$-wise variation, and $\rho_x(A)$ normalizes the profile range. See App. A.4.3 for details. After training, the quality predictor is frozen and serves as the quality landscape for policy optimization.

### 3.4.3 Budget Factor: GRPO for Cost Allocation

The quality landscape captures reasoning benefits but not computation cost. We therefore learn an state-dependent cost over the action reasoning depth $A$. The cumulative cost is defined as

$$c_1(x) = 0, \qquad c_A(x) = \frac{1}{k-1} \sum_{i=2}^{A} \lambda_i(x). \tag{10}$$

where $k$ is the maximum reasoning depth. For each $A$, we select the reasoning intensity with the highest quality along the $V$ dimension:

$$V_A^* = \arg \max_{V:(A,V) \in \mathcal{C}} \hat{Q}(x, A, V). \tag{11}$$

The reasoning intensity $(A, V_A^*)$ is scored by balancing its performance and cost:

$$z_A(x) = \frac{\hat{Q}(x, A, V_A^*) - c_A(x)}{\tau_q}, \qquad \pi(A|x) = \frac{\exp(z_A(x))}{\sum_{A'} \exp(z_{A'}(x))}. \tag{12}$$

We optimize policy $\pi(A|x)$ with GRPO (Shao et al., 2024; Guo et al., 2025). For each $x$, candidate intensities are sampled from an $\epsilon$-mixed behavior policy, and the reward is

$$r(x, A) = Q(x, A, V_A^*) - c_r \frac{A-1}{k-1}. \tag{13}$$

where $c_r$ controls the fixed normalized computation penalty. Group-relative advantages compare different reasoning intensities for the same $x$, making the optimization depend on relative quality-cost trade-off rather than the absolute scale of the quality values. See App. A.4 for details.

## 3.5 Adaptive Inference

As detailed in Algorithm 1, at each step, the router determines whether reasoning is needed and selects the reasoning intensity $(A, V)$ when activated. The routing states $x$ are encoded once, and the shared representation is used by both the frozen WAM and the router. If reasoning is not selected, the action stream performs direct denoising from the current observation. Otherwise, the WAM follows the selected reasoning intensity to generate the action chunk. See App. A.5 for details.

# 4 EXPERIMENTS

We conduct experiments to study the role of test-time video reasoning in WAMs and to evaluate ThinkWAM. Our experiments are organized around the following questions: **Q1:** Can ThinkWAM achieve strong performance across diverse benchmarks and embodiments? **Q2:** Should predictive reasoning be removed at test time, or adaptively invoked based on the state? **Q3:** Are the designs of adaptive reasoning, including multi-intensity reasoning training and budget factor, necessary? **Q4:** Does adaptive video reasoning remain effective in real-world robotic manipulation?

## 4.1 EXPERIMENTAL SETUP

**Simulation benchmarks.** We evaluate on LIBERO (Liu et al., 2023) (2,000 trials), LIBERO-Plus (Fei et al., 2025) under seven distribution shifts without fine-tuning (10,030 trials), and RoboCasa GR1 Tabletop (Bjorck et al., 2025) across 24 tasks (600 rollouts).

**Baselines.** We compare with strong manipulation policies from two categories: VLA-based methods, including OpenVLA-OFT (Kim et al., 2025) and $\pi_0$/$\pi_{0.5}$ (Black et al., 2024; Intelligence et al., 2025), ABot-M0 (Yang et al., 2026), $\pi_0$-FAST (Pertsch et al., 2025), DIAL (Chen et al., 2026b), and WAM-based methods, including Motus (Bi et al., 2026), LingBot-VA (Li et al., 2026a),WALA (Liu et al., 2026), AdaWAM (Tang et al., 2026), Fast-WAM (Yuan et al., 2026), Joint-WAM following the Fast-WAM, and LAWA (Li et al., 2026b).

**Implementation.** The WAM uses 30-layer video and action DiTs. We train the LIBERO and RoboCasa backbones for 50k and 330k steps, respectively, and use a maximum reasoning intensity of $(A, V) = (5, 25)$. More details are provided in Apps. B.1 and B.2, and App. A.6.

## 4.2 MAIN RESULTS

We first evaluate performance of ThinkWAM across the three benchmarks with two inference settings: **ThinkWAM-Adaptive** gates reasoning and selects its intensity per state, whereas **ThinkWAM-All-State** reasons at every state while retaining adaptive intensity selection.

**LIBERO.** Table 1 shows that ThinkWAM achieves 98.8% average success, outperforming all baselines. Compared with Fast-WAM (Yuan et al., 2026), which removes predictive reasoning at inference, this improvement demonstrates the value of test-time reasoning beyond its training effect. **LIBERO-Plus.** Table 2 evaluates robustness under distribution shift. ThinkWAM achieves the best overall performance and remains strong across diverse perturbations, demonstrating its generalization beyond the in-distribution LIBERO setting. **RoboCasa GR1 Tabletop.** Table 3 shows that ThinkWAM achieves 77.7% success on RoboCasa GR1 Tabletop, outperforming the compared methods. This result demonstrates the effectiveness of ThinkWAM on a different embodiment and task distribution. Details are provided in App. B.6.

Taken together, these results answer **Q1**, demonstrating that ThinkWAM achieves strong manipulation performance across benchmarks and embodiments. See App. B.3 and App. B.5 for more details.

## 4.3 SHOULD PREDICTIVE REASONING BE USED FOR EVERY DECISION?

We investigate whether future prediction should be removed at test time or selectively applied. We vary the gating threshold $\tau$ with the WAM fixed. The *reasoning ratio* is the percentage of decisions for which $\hat{g}(o_t) > \tau$($0\%$ denotes direct prediction). See App. B.4 for more details.

**Future prediction remains useful at test time.** Across all three benchmarks, completely removing predictive reasoning degrades control performance, while moving beyond the $0\%$ reasoning point improves task success. This shows that future latent reasoning remains useful at inference.

**Most of the benefit appears early and the value of reasoning is state-dependent.** Predictive reasoning yields diminishing returns as more states are routed to reasoning: across all three benchmarks, a small selectively chosen subset recovers most of the full-reasoning gain. At matched reasoning ratios, the learned router (black) consistently outperforms random routing (gray), especially

under limited budgets, indicating that it prioritizes states where reasoning is most valuable. On RoboCasa GR1, reasoning on only 28% of states improves success from 72.0% to 75.8%, recovering approximately 68% of the full-reasoning gain at 77.7%.

**Inference efficiency.** The direct route matches Fast-WAM at 1.298 TFLOPs per forward, while predictive reasoning costs 4.791 TFLOPs per forward. Thus, adaptive routing reduces inference computation by activating the higher-cost reasoning path only on selected states.

Taken together, these results answer **Q2**: future prediction remains useful at inference, but should be adaptively utilized based on the current state rather than uniformly removed.

Table 1: Comparison on LIBERO. We report success rate (%) on the four task suites and average.

| Method | Paradigm | Spatial | Object | Goal | Long | Avg. |
|---|---|---|---|---|---|---|
| $\pi_{0.5}$ | VLA | 98.8 | 98.2 | 98.0 | 92.4 | 96.8 |
| OpenVLA-OFT | VLA | 97.6 | 98.4 | 97.9 | 94.5 | 97.1 |
| Fast-WAM | WAM | 98.2 | **100.0** | 97.0 | 95.2 | 97.6 |
| Motus | WAM | 96.8 | 99.8 | 96.6 | 97.6 | 97.7 |
| LingBot-VA | WAM | 98.5 | 99.6 | 97.2 | 98.5 | 98.5 |
| AdaWAM | WAM | 98.0 | 99.6 | 97.1 | **99.1** | 98.5 |
| **ThinkWAM-Adaptive** | **WAM** | **99.4** | 99.8 | 97.6 | 97.8 | **98.7** |
| **ThinkWAM-All-State** | **WAM** | **99.4** | 99.8 | **98.4** | 97.4 | **98.8** |

Table 2: Comparison on LIBERO-Plus under seven distribution shifts. Cam., Lang., and Bg. denote camera, language, and background, respectively. Robot, Light, Noise, and Layout denote the corresponding distribution shifts, and Total reports the overall success rate (%).

| Method | Paradigm | Cam. | Robot | Lang. | Light | Bg. | Noise | Layout | Total |
|---|---|---|---|---|---|---|---|---|---|
| $\pi_0$ | VLA | 13.8 | 6.0 | 58.8 | 85.0 | 81.4 | 79.0 | 68.9 | 53.6 |
| $\pi_0$-FAST | VLA | 65.1 | 21.6 | 61.0 | 73.2 | 73.2 | 74.4 | 68.8 | 61.6 |
| OpenVLA-OFT | VLA | 56.4 | 31.9 | 79.5 | 88.7 | 93.3 | 75.8 | 74.2 | 69.6 |
| Fast-WAM | WAM | 38.5 | 71.0 | 74.2 | 92.1 | 64.0 | 74.3 | 74.7 | 69.0 |
| Joint-WAM | WAM | 53.2 | 72.8 | 83.9 | **96.6** | 58.5 | 79.8 | 79.0 | 74.6 |
| LAWA | WAM | **69.2** | 66.7 | 62.8 | 96.2 | **64.5** | 85.5 | 78.6 | 74.4 |
| **ThinkWAM-Adaptive** | **WAM** | 58.2 | 76.5 | 85.4 | 96.3 | 60.9 | 81.0 | 81.4 | **77.0** |
| **ThinkWAM-All-State** | **WAM** | 63.0 | **81.2** | **88.0** | 95.5 | 64.2 | **87.8** | **83.3** | **80.5** |

Table 3: Comparison on RoboCasa GR1 Tabletop. We report average task success rate (%).

| Method | Paradigm | Success Rate (%) |
|---|---|---|
| ABot-M0 | VLA | 58.3 |
| DIAL | VLA | 70.2 |
| WALA | VLA | 75.2 |
| Fast-WAM | WAM | 72.0 |
| **ThinkWAM-Adaptive** | **WAM** | **75.8** |
| **ThinkWAM-All-State** | **WAM** | **77.7** |

### 4.4 Ablation Studies

We ablate two components that enable adaptive video reasoning: adaptive budget factor and multi-intensity training. The effect of gain factor is analyzed in Sec. 4.3. See App. B.8 for additional ablation details. **w/o budget factor** removes adaptive reasoning-intensity selection and uses a fixed intensity $(A, V) = (5, 25)$ for all reasoning states. The success rate remains close (80.5% $\rightarrow$ 79.9%),

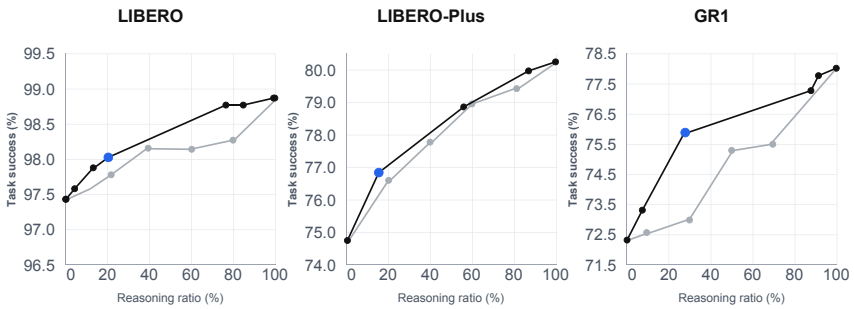


Figure 3: Task success versus reasoning ratio. The gating threshold $\tau$ controls the reasoning ratio. Black curves use adaptive reasoning, while gray curves use random routing. Most gains are achieved at low reasoning ratios, highlighted by the blue markers.

Table 4: Ablations on LIBERO-Plus under seven distribution shifts. Cam., Lang., and Bg. denote camera, language, and background, respectively.

| Variant | Cam. | Robot | Lang. | Light | Bg. | Noise | Layout | Total |
|---|---|---|---|---|---|---|---|---|
| ThinkWAM | 63.0 | 81.2 | 88.0 | 95.5 | 64.2 | 87.8 | 83.3 | 80.5 |
| w/o budget factor | 61.5 | 80.7 | 86.9 | 97.6 | 63.0 | 87.0 | 82.4 | 79.9 |
| w/o multi-intensity reasoning training | 62.2 | 79.7 | 86.9 | 94.6 | 62.2 | 83.8 | 82.0 | 78.8 |

showing that adaptive intensity selection maintains comparable performance while avoiding unnecessary predictive computation. **w/o multi-intensity reasoning training** uses only the shared synchronous video-action schedule, reducing success to 78.8%, showing that multi-intensity reasoning training improves adaptive inference robustness. Overall, these results address Q3, confirming that our designs are well-motivated and may provide useful insights for future research.

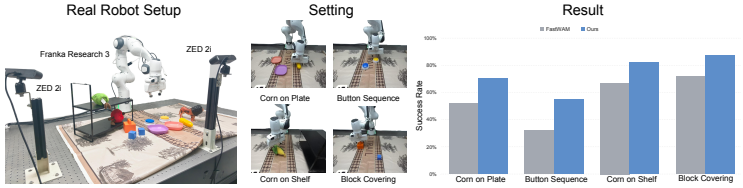


Figure 4: Real-world evaluation on the Franka Research 3 robot. ThinkWAM consistently improves success rates, demonstrating the effectiveness of adaptive predictive reasoning in real-world.

### 4.5 Real-World Evaluation

We further evaluate ThinkWAM on a Franka Research 3 with two ZED 2i cameras across four tasks covering sequential interaction, spatial reasoning, deformable objects, and constrained placement. For each task, we collect 50 teleoperated demonstrations and evaluate 40 trials, with the backbone fine-tuned per task. As shown in Figure 4, ThinkWAM achieves 73.8% average success, improving over FastWAM by 17.5 percentage points.

## 5 Conclusion

We show that test-time future prediction in WAMs is neither universally necessary nor uniformly removable: its value varies across decision states, and most of its benefit can be recovered by reasoning selectively. ThinkWAM combines a WAM backbone trained for multiple predictive-reasoning intensities with a state-adaptive router that determines whether and how deeply to reason. It achieves strong performance across three simulation benchmarks, while adaptive routing recovers up to 68% of the full-reasoning gain using reasoning for only 28% of decision states.

### AI USE STATEMENT

In this work, we used generative AI tools only to aid and polish the writing of the manuscript, including improving clarity, grammar, and presentation. We did not use generative AI tools for other research tasks requiring disclosure, and the remaining disclosure categories are not applicable to this work. All AI-assisted edits were reviewed by the authors. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

The main paper provides the formulation, training objectives, and experimental setup of ThinkWAM. Appendix A details the model architecture, multi-intensity predictive reasoning training, router supervision and optimization, adaptive inference, and hyperparameters. Appendix B provides additional benchmark settings, baseline details, reasoning-ratio analyses, qualitative results, per-task results, and ablations. Upon acceptance, we will release our code, data, and model weights to facilitate reproducibility.

## REFERENCES

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. $\pi_0$: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

William Chen, Jagdeep Singh Bhatia, Catherine Glossop, Nikhil Mathihalli, Ria Doshi, Andy Tang, Danny Driess, Karl Pertsch, and Sergey Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control. *arXiv preprint arXiv:2602.13193*, 2026a.

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

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

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL https://arxiv.org/abs/2502.05171.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jun Guo, Qiwei Li, Peiyan Li, Zilong Chen, Nan Sun, Yifei Su, Heyun Wang, Yuan Zhang, Xinghang Li, and Huaping Liu. Unified 4d world action modeling from video priors with asynchronous denoising. *arXiv preprint arXiv:2604.26694*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

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.

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 preprint arXiv:2601.21998*, 2026a.

Xiang Li, Yupeng Zheng, Songen Gu, Huailiang Ma, Feng Yu, Xian Nie, Shanshuai Yuan, Yujie Zang, Weize Li, Shuai Tian, et al. Latent action as intention enables efficient future imagination for world action models. *arXiv preprint arXiv:2608.24882*, 2026b.

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

Jiahao Liu, Zhongpu Xia, Shuai Tian, Huangrui Li, Yuhang Zheng, Ning Ma, Xin Fu, Xiaotian Liu, Jing Li, Yixian Li, et al. Wala learning executable latent actions from action-labeled demonstrations and action-free videos. *arXiv preprint arXiv:2607.11397*, 2026.

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

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.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

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 preprint arXiv:2602.22010*, 2026.

Yinzhou Tang, Jingbo Xu, Yu Shang, Zihao Song, Chen Gao, Wei Wu, and Yong Li. Dreaming when necessary: Advancing world action models with adaptive multi-modal reasoning, 2026. URL https://arxiv.org/abs/2606.07089.

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 preprint arXiv:2604.27792*, 2026.

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

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

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.

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.

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1702–1713. IEEE, 2025.

Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8152–8162, 2026.

# A Method Details

This appendix provides additional details for the method described in Sec. 3.1: the world-action model (WAM) architecture and training objective (App. A.1), the multi-intensity reasoning training (App. A.2), the data generation pipeline (App. A.3), the adaptive reasoning router (App. A.4), the adaptive inference procedure (App. A.5), and a complete summary of hyperparameters (App. A.6).

## A.1 World-Action Model: Architecture and Training Details

### A.1.1 Architecture

ThinkWAM instantiates the WAM as a mixture-of-transformers (MoT) with two diffusion transformer experts: a video expert for future-state prediction and an action expert for action generation. The video expert is initialized from Wan2.2-TI2V-5B (Wan et al., 2025) and uses 30 layers, and the action expert uses 30 layers. Both experts use 24 attention heads with head dimension 128 and maintain separate noise-level conditioning.

The video expert operates on latent representations produced by the Wan2.2 causal TI2V VAE with $16\times$ spatial compression, where multi-view observations are concatenated horizontally before encoding. The action expert operates on action chunks $\mathbf{a} \in \mathbb{R}^{T_a \times d_a}$.

The two experts share the same depth (30 layers), head count (24), and head dimension (128), while maintaining separate parameters for attention and feed-forward layers. At each layer, video and action tokens are jointly attended through mixed self-attention under the mask in Fig. 2, allowing information exchange between the two streams while keeping the expert parameters separate. Language and proprioceptive conditioning are provided through cross-attention.

**Context conditioning.** The language instruction is encoded by T5 into context tokens. The proprioceptive state $\mathbf{p}$ is linearly projected to the text embedding dimension and appended to the context as one extra token. Both experts cross-attend to the same context at every layer.

**Attention masks and the route variable $m$.** The WAM is trained to support both direct action prediction and latent future reasoning. During training, we sample a binary route variable $m \sim \mathrm{Bernoulli}(0.5)$ for each example, which controls whether the action branch can access future-video latents through masked attention.

The video branch always conditions on the observed first frame and refines the remaining future-video latents under its causal attention mask. The action branch uses a route-dependent mask:

- **Direct action prediction** ($m = 0$): the action tokens can attend to the observed first-frame video tokens, while access to all future-video latents is masked out. The action is therefore predicted directly from the current observation without using predicted future information.
- **Latent future reasoning** ($m = 1$): the action tokens can additionally attend to the future-video latents produced by the video branch at their current refinement level. These latents

act as intermediate predictive representations for action generation and are not decoded into pixel space.

Sampling the two routes uniformly enables a single backbone to learn both inference modes with shared parameters. At test time, the router determines whether the direct route or the predictive-reasoning route is executed for the current decision state.

#### A.1.2 TRAINING OBJECTIVE

In WAM, both the video and action streams are trained with rectified flow matching. Given clean data $\mathbf{x}$ and Gaussian noise $\boldsymbol{\epsilon}$, we construct

$$\tilde{\mathbf{x}}_\sigma = (1-\sigma)\mathbf{x} + \sigma\boldsymbol{\epsilon},$$

with velocity target $\boldsymbol{\epsilon} - \mathbf{x}$. The two streams are jointly optimized with the backbone objective in Eq. equation 1, using $\lambda_v = \lambda_a = 1$.

In implementation, the two streams follow the same training objective but maintain separate noise conditioning and valid regions for the loss:

- *Multi-intensity reasoning training.* The video and action experts are conditioned on their own noise levels, $\sigma_v$ and $\sigma_a$, respectively, through adaLN. This supports the multi-intensity predictive reasoning training in Sec. 3.2.3, where synchronous and asynchronous video-action noise schedules expose the action policy to future-video latents at different refinement levels.
- *First-frame conditioning.* The first video latent frame corresponds to the current observed image and is kept as a clean conditioning frame. During training, it is re-inserted before each forward pass and excluded from the video prediction loss, so the video objective is applied only to future frames. During inference, the same observed first-frame condition is re-imposed after every denoising update, as shown in Algorithm 1.
- *Padding masks.* Episodes may contain padded video or action timesteps. These padded positions are removed from the corresponding loss using `is_pad` masks, and both video and action losses are averaged only over valid timesteps.

### A.2 MULTI-INTENSITY PREDICTIVE REASONING TRAINING: ADDITIONAL DETAILS

#### A.2.1 TRAINING SCHEDULE SAMPLING

As described in Sec. 3.2.3, multi-intensity predictive reasoning training is applied to the predictive-reasoning route so that the action policy can use future-video latents at different refinement levels. For each training example, we first sample the route variable $m \sim \mathrm{Bernoulli}(0.5)$. When $m = 0$, the model is trained for direct action prediction using the synchronous video–action schedule. When $m = 1$, the action branch can attend to future-video latents, and we further sample either a synchronous or an asynchronous noise schedule.

Both streams use the shifted rectified-flow clock in Eq. equation 2, with $s_a = s_v = 5$ and 1,000 training timesteps. The sampling procedure is summarized in Algorithm 2.

**Training for multiple reasoning intensities.** The multi-intensity predictive reasoning training enables a single backbone to operate across different reasoning intensities $(A, V)$. Different intensities induce different relative refinement levels between the video and action streams, and therefore expose the action branch to future-video latents at different stages of denoising. To make these inference-time configurations accessible during training, we construct the training noise-pair distribution to cover the corresponding video-action refinement relationships.

On the predictive-reasoning route, this distribution includes both the shared-clock case and asynchronous training in which the video stream remains less refined than the action stream. On the direct route, future-video latents are masked from the action branch, so only the shared-clock configuration is required.

For a reasoning intensity $(A, V)$, we use $V \geq A$. During joint denoising, the video stream is therefore refined more slowly than the action stream, so the corresponding noise levels satisfy $\sigma_v^{(i)} \geq \sigma_a^{(i)}$.

**Algorithm 2** Training schedule sampling for multi-intensity predictive reasoning.

**Require:** clean video/action data $(x_v, x_a)$, clock shifts $s_a, s_v$, asynchronous probability $p_{\text{async}}$

Sample $m \sim \text{Bernoulli}(0.5)$ and construct the corresponding attention mask
**if** $m = 0$ **then** ▷ direct prediction
  Sample $u \sim \mathcal{U}(0, 1)$
  $\sigma_a \leftarrow \phi_{s_a}(u)$, $\sigma_v \leftarrow \phi_{s_v}(u)$
**else** ▷ predictive reasoning
  Sample $b \sim \text{Bernoulli}(p_{\text{async}})$
  **if** $b = 1$ **then** ▷ asynchronous schedule
    Sample $u_a \sim \mathcal{U}(0, 1)$
    $\sigma_a \leftarrow \phi_{s_a}(u_a)$
    Sample $\sigma_v \sim \mathcal{U}(\sigma_a, 1)$
  **else** ▷ synchronous schedule
    Sample $u \sim \mathcal{U}(0, 1)$
    $\sigma_a \leftarrow \phi_{s_a}(u)$, $\sigma_v \leftarrow \phi_{s_v}(u)$
  **end if**
**end if**
Add noise to the video and action streams using $(\sigma_v, \sigma_a)$
Re-impose the observed first-frame condition
Update the backbone with $\mathcal{L}_{\text{wam}}$

Sampling asynchronous training pairs from the same region matches the video-action refinement relationships encountered by the reasoning route at inference.

For a fixed action noise level $\sigma_a$, different values of $\sigma_v$ correspond to future-video latents at different refinement levels. Sampling $\sigma_v \sim \mathcal{U}(\sigma_a, 1)$ therefore exposes the action policy to future representations ranging from equally refined to less refined than the action stream. Therefore, the multi-intensity reasoning training allows the same backbone to support different reasoning intensities $(A, V)$ at inference.

### A.2.2 Relation between reasoning intensities and denoising schedules

As described in Sec. 3.2.3, a reasoning intensity $(A, V)$ specifies how the video and action streams are refined during predictive reasoning. Here, $A$ determines the number of joint video-action updates used to predict the action, while $V$ controls the refinement rate of the future-video latents.

Starting from $(\sigma_a^{(0)}, \sigma_v^{(0)}) = (1, 1)$, the $i$-th joint update, $i \in \{0, \dots, A-1\}$, uses

$$\sigma_a^{(i)} = \phi_{s_a}\left(1 - \frac{i}{A}\right), \qquad \sigma_v^{(i)} = \phi_{s_v}\left(1 - \frac{i}{V}\right),$$

following the inference procedure in Algorithm 1. The action stream therefore completes its denoising process after exactly $A$ joint updates. The video stream follows the schedule determined by $V$ and may remain only partially refined when action generation finishes.

Because the candidate space satisfies $V \geq A$, the future-video latents are refined no faster than the action stream. Consequently, $\sigma_v^{(i)} \geq \sigma_a^{(i)}$ throughout the joint updates. Different choices of $(A, V)$ therefore expose the action branch to future-video representations at different refinement levels, which is the basis of the reasoning-intensity space used by ThinkWAM.

Importantly, the future-video latents are used only as intermediate predictive representations: the action branch accesses them through masked attention during the joint updates, and they are not decoded into pixels.

For direct action prediction, predictive reasoning is bypassed. The action branch does not access future-video latents and instead performs $N_a$ action-only denoising steps conditioned on the current observation.

### A.3 Data Generation Pipeline: Additional Details

As described in Sec. 3.3, the router is trained from supervision collected offline using the frozen WAM. We construct two types of supervision corresponding to the two routing decisions: reasoning gain for the gating factor and intensity quality for the budget factor. Below, we provide additional details on the routing-state representation and the collection of these two supervision signals.

**Routing-state representation.** The routing state is $x = (o, \ell, p)$, consisting of the current observation, language instruction, and proprioceptive state. Its embedding $e(x)$ is constructed without running the diffusion process. Specifically, we use the mean-pooled VAE latent of the current observation together with the masked mean-pooled context representation of the language instruction and proprioceptive token. These features provide the router with the current visual and task context before any predictive reasoning is performed.

**Reasoning-gain supervision.** Following Eq. equation 3, the gating factor is supervised by measuring how much predictive reasoning improves action prediction for the same routing state. We evaluate the frozen WAM under the direct and predictive-reasoning routes and measure their action prediction errors using mean squared error:

$$L_{\mathrm{o}}(x, a) = \mathrm{MSE}(\hat{a}_{\mathrm{direct}}(x), a)\,, \qquad L_{\mathrm{r}}(x, a) = \mathrm{MSE}(\hat{a}_{\mathrm{reason}}(x), a)\,. \tag{14}$$

The reasoning gain is then

$$g(x) = \mathbb{E}_{\mathrm{seed}}\left[L_{\mathrm{o}}(x, a) - L_{\mathrm{r}}(x, a)\right]. \tag{15}$$

We average the paired difference over five initial-noise seeds to reduce sampling variation. A positive $g(x)$ indicates that predictive reasoning reduces the action prediction error for the current state.

**Intensity-quality supervision.** When predictive reasoning is used, the budget factor must determine which reasoning intensity $(A, V)$ to execute. We therefore evaluate the frozen WAM over the candidate intensity space

$$\mathcal{C} = \{(1, 1)\} \cup \{(A, V) : 2 \le A \le k,\ A \le V \le V_{\max}\}, \tag{16}$$

where $k = 5$ and $V_{\max} = 25$.

For each routing state $x$ and each candidate $(A, V) \in \mathcal{C}$, we execute the corresponding reasoning-intensity schedule and compute the SmoothL1 action prediction error

$$L(x, A, V) = \mathrm{SmoothL1}(\hat{a}(x; A, V), a)\,. \tag{17}$$

We convert the prediction error into a state-normalized quality target:

$$Q(x, A, V) = -\frac{L(x, A, V)}{\sigma(x)}, \tag{18}$$

where $\sigma(x)$ measures the error gap between the lowest and the highest reasoning intensity for the same state. Since the prediction error enters with a negative sign, a larger $Q(x, A, V)$ indicates better action prediction quality.

Finally, we center the quality targets within the intensity space:

$$\widetilde{Q}(x, A, V) = Q(x, A, V) - \frac{1}{|\mathcal{C}|} \sum_{(A', V') \in \mathcal{C}} Q(x, A', V'). \tag{19}$$

This removes state-specific offsets and makes the supervision focus on the relative quality of different reasoning intensities for the same state.

### A.4 Adaptive Reasoning Router: Architecture and Training

#### A.4.1 Architecture

As described in Sec. 3.4, the adaptive reasoning router operates on the frozen routing-state representation $e(x)$ and contains two lightweight components corresponding to the two routing decisions: the gating factor and the budget factor.

The gating factor uses a 2-layer Transformer encoder with hidden size 256, 8 attention heads, and an FFN ratio of 4, followed by an MLP head that predicts the reasoning gain $\hat{g}(x)$.

The budget factor uses a 4-layer Transformer encoder with hidden size 512, 8 attention heads, and an FFN ratio of 4. Given the routing-state representation together with learned embeddings $e_A(A)$ and $e_V(V)$ for the reasoning intensity, it predicts the quality $\hat{Q}(x, A, V)$ following Eq. equation 7. The learned intensity embeddings have dimension 128. At inference, the quality values for all candidate intensities in $\mathcal{C}$ are evaluated in parallel to construct the state-dependent quality landscape used by the budget factor.

### A.4.2 Gating Factor

The gating factor estimates the state-dependent benefit of predictive reasoning and is trained with the regression and ranking objective in Eqs. equation 5 and equation 6.

The regression term directly fits the measured reasoning gain $g(x)$. Both the predicted and target gains are normalized by a fixed robust scale $\sigma_g$ computed once from the training set, which stabilizes the magnitude of the regression objective.

The ranking term complements gain regression by preserving the relative ordering of reasoning benefits within each mini-batch. For a pair of states $(i, j)$ with $|g_j - g_i| > \delta$, the pairwise loss is weighted by

$$c_{ij} = \min\left(\frac{|g_j - g_i|}{\sigma_g}, C_{\max}\right).$$

Pairs with larger differences in reasoning gain therefore receive greater weight, while pairs with very similar gains contribute less to the ranking objective. We use $\delta = 0$, $C_{\max} = 10$, and ranking weight $w = 0.1$. The gating factor is optimized with AdamW using a learning rate of $10^{-3}$, weight decay $10^{-4}$, and batch size 8 for 5,000 steps.

### A.4.3 Budget factor: quality training

As described in Sec. 3.4.2, the budget factor models the state-dependent quality landscape over reasoning intensities through

$$\hat{Q}(x, A, V) = f_Q(e(x), e_A(A), e_V(V)).$$

The predictor is trained on the centered quality targets $\widetilde{Q}(x, A, V)$.

In addition to the global quality regression, we use the auxiliary same-$A$ profile loss $\mathcal{L}_V$ in Eq. equation 9 to preserve the relative quality variation along the video-depth dimension $V$ for a fixed action depth $A$.

For each state $x$ and action depth $A$, let $\mathcal{C}_A = \{(A, V') \in \mathcal{C}\}$ denote the candidate intensities sharing the same $A$. We center the target quality values within $\mathcal{C}_A$ to obtain the profile $q^c_{x,A}$, and apply the same operation to the predicted values to obtain $\hat{q}^c_{x,A}$. The profile range is denoted by $\rho^\circ_x(A)$ and normalized as

$$\rho_x(A) = \max(\rho^\circ_x(A), \rho_{\min}).$$

We further define

$$\mathcal{S}_A = \{x : \rho^\circ_x(A) \geq \tau\},$$

which keeps only states with non-trivial variation across $V$. The summation in Eq. equation 9 starts from $A = 2$ because $A = 1$ has only one valid video depth, $V = 1$, and therefore no within-$A$ profile to preserve.

The auxiliary loss is introduced because intensity selection depends not only on the quality differences across action depths, but also on which video refinement depth is most useful for a fixed $A$. Without this constraint, the smaller variation along $V$ can be dominated by the larger quality differences across $A$ during training.

We use $\beta = 0.02$, $\tau = \rho_{\min} = 5 \times 10^{-3}$, and $\lambda_V = 0.2$. After quality training, the predictor is frozen and used as the quality landscape for the subsequent cost-aware policy optimization.

### A.4.4 BUDGET FACTOR: GRPO FOR COST-AWARE ALLOCATION

After quality training, we utilize GRPO to learn the state-dependent computation cost that determines how much predictive reasoning should be allocated to each routing state.

**Cost model.** As defined in Eq. equation 10, the budget factor assigns a cumulative cost $c_A(x)$ to each action depth $A$. We parameterize this cost through bounded marginal increments:

$$\lambda_i(x) = \lambda_{\max} \operatorname{sigmoid}(h_i(x)), \qquad i = 2, \dots, k, \tag{20}$$

where $\lambda_{\max} = 0.1$. The cumulative cost is

$$c_1(x) = 0, \qquad c_A(x) = \frac{1}{k-1} \sum_{i=2}^{A} \lambda_i(x).$$

Because each marginal increment is non-negative, the resulting cost increases monotonically with the action depth $A$. This cost is combined with the quality prediction in Eq. equation 12 to produce the depth policy $\pi(A \mid x)$.

**Quality-cost reward.** For each candidate action depth $A$, we first select the video depth with the highest predicted quality:

$$V_A^* = \arg \max_{V:(A,V)\in\mathcal{C}} \hat{Q}(x, A, V),$$

following Eq. equation 11. GRPO then evaluates the resulting intensity $(A, V_A^*)$ using the reward in Eq. equation 13:

$$r(x, A) = Q(x, A, V_A^*) - c_r \frac{A-1}{k-1}. \tag{21}$$

The first term measures the observed quality of the selected reasoning intensity, while the second term penalizes larger action depths with a fixed normalized computation cost. We use $c_r = 0.01$ on LIBERO and $c_r = 0.03$ on RoboCasa GR1. GRPO uses this reward to optimize the cost head so that the resulting policy learns a better quality-computation trade-off for each state.

**GRPO optimization.** For each routing state $x$, we sample a group of $G = 32$ candidate action depths from the exploration policy

$$\pi_{\text{mix}}(A \mid x) = (1-\epsilon)\pi(A \mid x) + \epsilon \operatorname{Unif}_{\{1,\dots,k\}}(A), \qquad \epsilon = 0.2. \tag{22}$$

The uniform component maintains exploration over both shallow and deep reasoning depths during training.

For the sampled group, we compute group-relative advantages without variance normalization:

$$\operatorname{adv}(A) = r(A) - \overline{r}.$$

The policy is optimized with a PPO-clipped objective using $\epsilon_{\text{PPO}} = 0.2$, together with a behavior-policy importance correction, a KL penalty to the frozen initial policy ($\lambda_{\text{KL}} = 0.02$), and an entropy coefficient of $0.01$. We use 2 epochs per batch and optimize the cost head for $2{,}000$ updates.

**Training summary.** The router is trained on the frozen WAM in two stages (Sec. 3.4).

*(i) Supervised training.* Using the offline measurements collected in App. A.3, the gating factor is trained from reasoning-gain supervision, while the budget factor's quality predictor is trained from intensity-quality supervision. The gating factor is trained for $5{,}000$ steps, and the quality predictor is trained for $5{,}000$ updates with the best validation checkpoint retained.

*(ii) Cost-aware policy optimization.* After the quality predictor is frozen, GRPO optimizes only the state-dependent cost head, and hence the depth policy $\pi(A \mid x)$, for $2{,}000$ updates. The WAM backbone and the quality predictor remain frozen throughout this stage.

## A.5 ADAPTIVE INFERENCE: IMPLEMENTATION DETAILS

This section provides implementation details for the adaptive inference procedure in Sec. 3.5 and Algorithm 1. At each control step, the current observation, language instruction, and proprioceptive state are encoded and reused by the frozen WAM and the router. The gating factor then determines whether to execute direct action prediction or predictive reasoning.

**Direct action prediction ($\hat{g} \leq \tau$).** When predictive reasoning is not activated, the action branch follows the direct route. The observed first-frame video features are computed once and cached. The action stream then performs $N_a$ action-only denoising steps, attending only to the cached representation of the current observation through the masked mixed attention.

Since no future-video denoising is performed, this route has the same per-forward computation as Fast-WAM in our implementation, 1.298 TFLOPs.

**Predictive reasoning ($\hat{g} > \tau$).** When predictive reasoning is activated, the budget factor first selects the action depth

$$A_t^* = \arg\max_A \pi(A \mid e(x_t)),$$

and then selects the corresponding video depth $V^*_{A_t^*}$ using Eq. equation 11. The resulting pair $(A_t^*, V^*_{A_t^*})$ defines the reasoning intensity for the current decision state.

The WAM then executes $A_t^*$ joint video–action updates following the denoising schedule associated with the selected reasoning intensity, as detailed in App. A.2.2. The observed first-frame condition is re-imposed after every update, following Algorithm 1. Each joint forward activates both the video and action streams and requires 4.791 TFLOPs in our implementation.

Therefore, the number of joint WAM forwards performed during predictive reasoning is determined by the selected action depth $A_t^*$, while $V^*_{A_t^*}$ controls the refinement rate of the future-video latents within these updates.

**Router computation.** The routing decision is computed from the frozen routing-state representation before the WAM denoising process. The router requires only one forward pass per control step and does not introduce additional diffusion iterations.

### A.6 Hyperparameters

Unless otherwise specified, the following hyperparameters are shared across benchmarks.

The WAM backbone uses 30 layers for both the video and action DiTs, shifted rectified-flow clocks with $s_v = s_a = 5$, and 1,000 training timesteps. The video and action loss weights are $\lambda_v = \lambda_a = 1$. The backbone is trained for 50k steps on LIBERO and 330k steps on RoboCasa, with batch sizes 8 and 16, respectively.

For multi-intensity predictive reasoning training, the route variable is sampled as $m \sim \mathrm{Bernoulli}(0.5)$. On the predictive-reasoning route, the asynchronous schedule is sampled with $p_{\mathrm{async}} = 0.5$. The reasoning-intensity space uses maximum action depth $k = 5$ and maximum video depth $V_{\max} = 25$.

The gating factor uses ranking weight $w = 0.1$, pair margin $\delta = 0$, and ranking cap $C_{\max} = 10$, and is trained for $5{,}000$ steps. The quality predictor uses Huber parameter $\beta = 0.02$ for the same-$A$ profile loss and weight $\lambda_V = 0.2$, and is trained for $5{,}000$ updates with the best validation checkpoint retained.

For GRPO, we use cost-penalty weight $c_r = 0.01$ on LIBERO and $c_r = 0.03$ on RoboCasa GR1, cost scale $\lambda_{\max} = 0.1$, group size $G = 32$, exploration coefficient $\epsilon = 0.2$, PPO clip $0.2$, KL penalty $\lambda_{\mathrm{KL}} = 0.02$, entropy coefficient $0.01$, and learning rate $10^{-5}$. The cost head is optimized for $2{,}000$ updates with two PPO epochs per batch.

At inference, direct action prediction uses $N_a = 5$ action denoising steps. The gating threshold $\tau$ controls the reasoning ratio to obtain the performance-computation curves in Fig. 3.

## B Experimental Details

### B.1 Benchmark Details

We evaluate on three simulation benchmarks and one real-world platform, covering different embodiments, task distributions, and distribution shifts.

**LIBERO.** LIBERO (Liu et al., 2023) contains four task suites—Spatial, Object, Goal, and Long (LIBERO-10)—with ten tasks each, instantiated in a MuJoCo-based tabletop environment with a fixed robot arm. Following the standard protocol, we run 50 trials per task, giving 2,000 trials in total. The model receives two camera views (an exterior view and a wrist view, resized to $224 \times 224$ and concatenated horizontally), the language instruction, and the proprioceptive state, and outputs a 32-step delta end-effector action chunk (6-DoF pose plus a non-delta gripper dimension). During evaluation the initial 30 environment steps are used for settling, the action chunk is re-planned every 10 steps, and the gripper command is binarized; these settings are identical for ThinkWAM and the retrained WAM baselines.

**LIBERO-Plus.** LIBERO-Plus (Fei et al., 2025) perturbs the LIBERO suites along seven distribution-shift dimensions: *camera* (viewpoint changes), *robot* (initial robot states), *language* (instruction paraphrases), *light* (lighting conditions), *background* (background textures), *noise* (sensor noise), and *layout* (object layouts). We evaluate the LIBERO-trained model on LIBERO-Plus *without any fine-tuning*, which tests how well adaptive predictive reasoning transfers under distribution shift; the full evaluation comprises 10,030 trials.

**RoboCasa GR1 Tabletop.** RoboCasa GR1 Tabletop (Bjorck et al., 2025) evaluates bimanual humanoid manipulation with the GR-1 robot across 24 tabletop tasks, covering diverse tasks. We follow the training data of DIAL (Chen et al., 2026b) for this benchmark and run 25 rollouts per task, for 600 rollouts. We repeat this full evaluation three times (1800 rollouts in total) and report the average success rate across the three runs.

**Real-world manipulation.** The real-world platform is a Franka Research 3 arm with two ZED 2i cameras (an exterior view and a wrist view). We consider four tasks that cover diverse challenges: *Button Sequence* (sequential interaction: press the yellow button, then press the blue button), *Block Covering* (spatial reasoning: cover the blue block with the box), *Corn on Shelf* (constrained placement: put the corn on the upper rack), and *Corn on Plate* (deformable-object manipulation: put the corn on the pink plate). For each task we collect 50 teleoperated demonstrations with both camera views and evaluate 40 independent trials. The backbone is fine-tuned per task with the same recipe as in simulation, and the router is trained on the frozen backbone with the same two-stage procedure as in Sec. 3.4.

### B.2 Baseline Details

We compare against strong manipulation policies from two categories. Moreover, we follow the standard evaluation protocols of each benchmark and report baseline results under the corresponding benchmark settings.

**VLA-based baselines.**

- **OpenVLA-OFT** (Kim et al., 2025) fine-tunes OpenVLA with parallel decoding, continuous action chunks, and L1 regression, which substantially improves inference speed and manipulation success over the original discrete-token OpenVLA. It maps the current observation and instruction directly to actions without any future modeling. Compared on LIBERO and LIBERO-Plus.
- $\pi_0$ (Black et al., 2024) pairs a vision-language backbone with a flow-matching action expert that denoises continuous action chunks conditioned on the current observation. We compare this method on LIBERO-Plus.
- $\pi_{0.5}$ (Intelligence et al., 2025) extends $\pi_0$ with co-training on heterogeneous data for open-world generalization, and is one of the strongest generalist VLA policies. We compare this method on LIBERO.
- $\pi_0$**-FAST** (Pertsch et al., 2025) combines $\pi_0$ with FAST action tokenization, converting continuous action chunks into compact discrete action tokens for autoregressive prediction. We compare this method on LIBERO-Plus.
- **ABot-M0** (Yang et al., 2026) is a VLA foundation model that learns an action manifold for robotic manipulation, generating actions on a learned manifold conditioned on the current observation and instruction. We compare this method on RoboCasa GR1.

- **DIAL** (Chen et al., 2026b) decouples intent and action in an end-to-end VLA via latent world modeling, using a latent intent representation to mediate between perception and action generation. We compare this method on RoboCasa GR1.

**WAM-based baselines.**

- **Motus** (Bi et al., 2026) is a unified latent-action world model that jointly models future visual observations and robot actions, with the action stream conditioned on the jointly denoised future representation. We compare this method on LIBERO.
- **LingBot-VA** (Li et al., 2026a) couples robot control with a causal world model, generating actions while causally predicting future visual states at inference. We compare this method on LIBERO.
- **WALA** (Liu et al., 2026) learns executable latent actions from action-labeled demonstrations together with action-free videos, and decodes latent plans into actions at test time. We compare this method on RoboCasa GR1.
- **AdaWAM** (Tang et al., 2026) studies multimodal reasoning by selecting between textual and visual reasoning modalities during execution. Its focus is on choosing which modality to use under a reasoning-based policy. ThinkWAM, instead, studies the predictive reasoning process intrinsic to the WAM backbone itself: whether future-latent reasoning is needed for the current decision state and, if so, what reasoning intensity should be used. We compare this method on LIBERO.
- **Fast-WAM** (Yuan et al., 2026) retains future-video prediction only as a training objective and removes it entirely at inference: the action stream denoises directly from the current observation, paying no video denoising cost but also receiving no test-time future information. It shares the training data and backbone recipe with ThinkWAM, which makes it the most direct comparison for the value of test-time reasoning. Compared on all three simulation benchmarks and the real-world tasks.
- **Joint-WAM** implements joint video-action denoising following the Fast-WAM codebase: the action stream attends to synchronously denoised future latents at *every* decision state, with the same training data as ThinkWAM. It represents the "always reason at a fixed schedule" operating point and, together with Fast-WAM ("never reason"), brackets the two fixed strategies that ThinkWAM interpolates between adaptively. We compare this method on LIBERO-Plus.
- **LAWA** (Li et al., 2026b) treats latent actions as intentions and enables efficient future imagination for world action models, generating compact latent intention representations that condition action generation with reduced future-imagination cost. We compare this method on LIBERO-Plus.

**Relation to ThinkWAM.** Existing methods treat test-time video largely as a fixed inference component or as a choice among separate modalities. For example, some WAMs use future prediction at every decision state, while Fast-WAM removes it uniformly at inference; AdaWAM instead focuses on selecting between textual and visual reasoning modalities.

ThinkWAM studies a different and complementary problem: the predictive reasoning process intrinsic to the WAM backbone itself. The action branch accesses future latents produced by the same backbone through masked attention, and the router determines, for each decision state, whether this internal predictive reasoning should be activated and, if so, what reasoning intensity should be used.

### B.3 Evaluation and Operating Points

All evaluations use closed-loop control. At each control step, the policy observes the current environment state, the router makes the routing decision, and the selected inference route produces an action chunk. The first part of the predicted chunk is then executed before the next observation is received and the policy replans.

We evaluate two inference settings. In ThinkWAM-Adaptive, predictive reasoning is activated only when $\hat{g}(x) > \tau$; otherwise, the model performs direct action prediction. When reasoning is acti-

vated, the budget factor selects the reasoning intensity according to Eq. equation 12. In ThinkWAM-All-State, every decision state is routed to predictive reasoning, while the reasoning intensity remains adaptively selected by the same budget factor.

The two settings therefore share the same backbone and intensity-selection policy and differ only in whether the gating factor is used to suppress predictive reasoning. Direct action prediction uses $N_a = 5$ action-only denoising steps, while predictive reasoning uses at most $A = 5$ joint video-action updates per control step.

### B.4 Reasoning-Ratio Analysis

Varying $\tau$ changes only how many states are routed to predictive reasoning; the WAM backbone, the budget factor, and all other inference components remain fixed. We measure this coverage as the *reasoning ratio*: the fraction of control decisions with $\hat{g}(x) > \tau$, computed over the whole evaluation. Lowering $\tau$ moves the policy along this axis, from direct action prediction everywhere ($0\%$) to ThinkWAM-All-State ($100\%$), which is how Fig. 3 traces task success as a function of how often reasoning is invoked.

The random-routing baseline (gray curves in Fig. 3) answers the complementary question of *which* states should reason. It replaces only the gating decision with random selection: each control state is routed to predictive reasoning independently at random with a fixed probability $r$. The backbone, and every other component of the inference pipeline are unchanged, so a learned router and the random baseline that invoke reasoning equally often differ only in which states are selected. Their comparison at matched reasoning ratios therefore isolates the value of choosing the right states from the value of reasoning more often.

**Detailed analysis.** Figure 3 shows the relationship between predictive reasoning and task success across all three benchmarks. Starting from the $0\%$ reasoning point, introducing predictive reasoning improves performance, confirming that future-latent prediction remains useful at inference. However, the improvement is not proportional to the number of states routed to reasoning. Performance rises more rapidly when reasoning is first introduced and then improves more gradually as the reasoning ratio increases.

This behavior indicates that the benefit of predictive reasoning is state-dependent. A subset of decision states contributes a large fraction of the available reasoning gain, while many other states can be handled effectively through direct action prediction. As a result, increasing the reasoning ratio beyond these informative states provides progressively smaller returns.

**Benchmark performance.** The same overall trend appears across LIBERO, LIBERO-Plus, and RoboCasa GR1. On LIBERO, performance is already high without predictive reasoning, and selectively activating reasoning further improves task success before the curve gradually approaches the all-state result. LIBERO-Plus exhibits a larger overall benefit from predictive reasoning under distribution shift: performance continues to improve as more useful states are routed to reasoning, while the gain becomes progressively smaller toward the all-state operating point.

The effect is particularly clear on RoboCasa GR1. Direct action prediction achieves $72.0\%$ task success, while routing only $28\%$ of decision states to predictive reasoning increases success to $75.8\%$. Reasoning at every state reaches $77.7\%$, so the $28\%$ operating point recovers approximately $68\%$ of the full-reasoning improvement. Thus, a substantial fraction of the predictive-reasoning benefit can be obtained without applying reasoning uniformly throughout the episode.

**Learned versus random routing.** The random-routing curves in Fig. 3 further show that reasoning frequency alone does not explain the improvement. At the same reasoning ratio, the learned gating factor consistently achieves higher task success than randomly routing the same fraction of decision states, with the difference being more pronounced when the available reasoning budget is limited.

Because both settings use the same WAM policy after a state is routed to reasoning, their difference comes from which states are selected. The learned gating factor therefore concentrates predictive reasoning on states for which future information is more useful, whereas random routing spends the same amount of reasoning computation without accounting for state-dependent benefit.

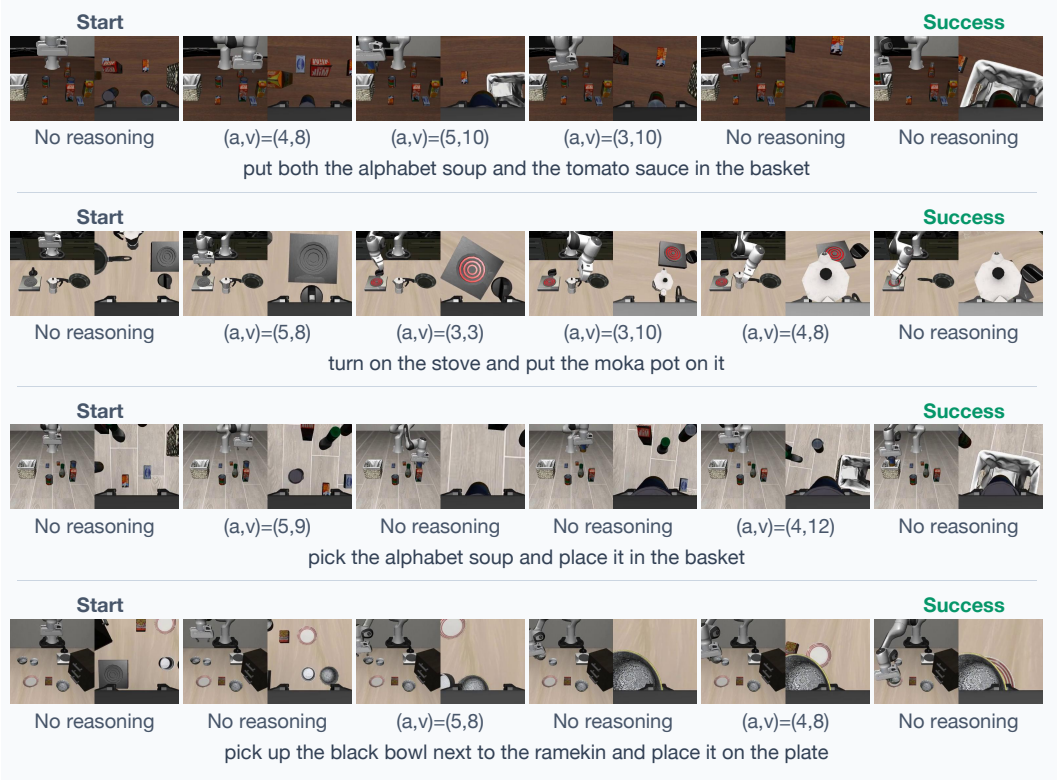


Figure 5: **Qualitative visualization of adaptive reasoning on LIBERO.** Each row shows one successful episode from the initial state to task completion. The text below each frame indicates the routing decision at the corresponding control step: *No reasoning* denotes direct action prediction, while $(A, V)$ denotes predictive reasoning with the selected reasoning intensity. The policy dynamically switches between direct prediction and different reasoning intensities across decision states, illustrating state-dependent allocation of inference computation.

**Implications for adaptive predictive reasoning.** These results reveal two complementary properties of test-time predictive reasoning. First, predictive reasoning cannot be uniformly removed, since enabling it improves task success across all three benchmarks. Second, it does not need to be applied uniformly either, because its benefit varies substantially across decision states and additional reasoning exhibits diminishing returns.

### B.5 Qualitative Visualization of Adaptive Reasoning

#### B.5.1 LIBERO

Figure 5 provides qualitative examples of the adaptive reasoning behavior on LIBERO. For each episode, we visualize several representative decision states from the start of execution to task completion, together with the routing decision at each state. The router dynamically alternates between direct action prediction (*No reasoning*) and predictive reasoning with different reasoning intensities $(A, V)$.

The examples show that predictive reasoning is not invoked uniformly throughout an episode. Instead, computation is allocated selectively across decision states: some states are handled directly, while others trigger different amounts of predictive reasoning. This state-dependent behavior allows the policy to retain predictive reasoning when it is useful while avoiding unnecessary computation at other steps.

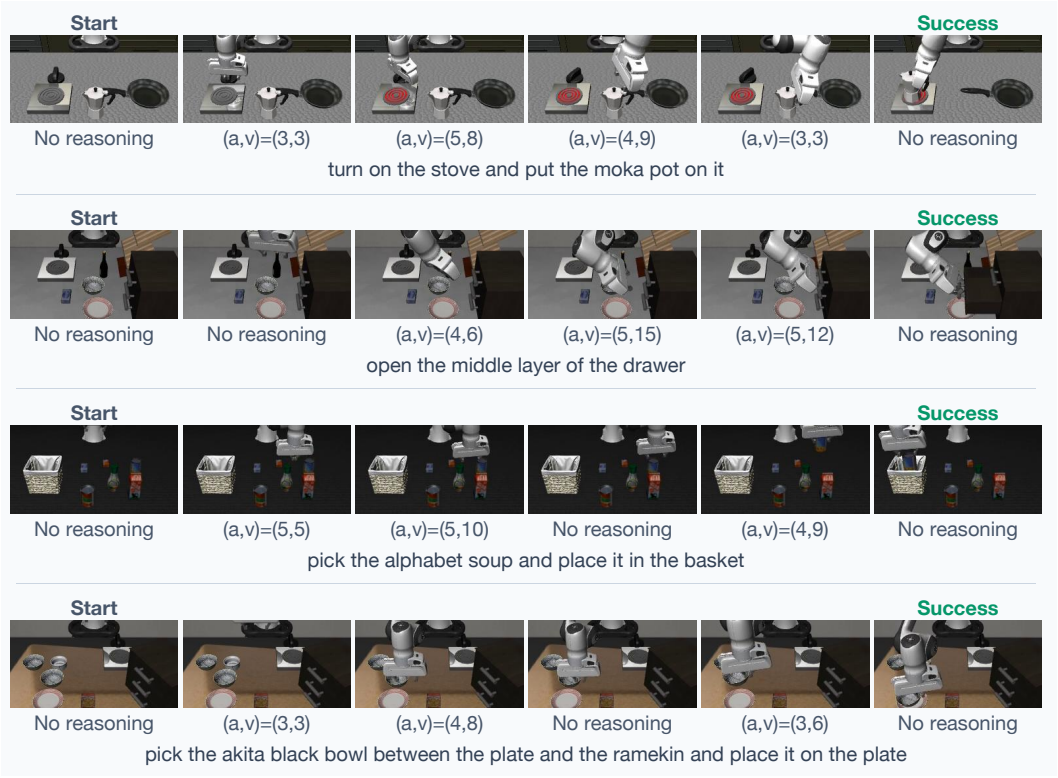


Figure 6: **Qualitative visualization of adaptive reasoning on LIBERO-Plus.** Each row shows one successful episode from the initial state to task completion. The text below each frame indicates the routing decision at the corresponding control step: *No reasoning* denotes direct action prediction, while $(A, V)$ denotes predictive reasoning with the selected reasoning intensity. The policy dynamically switches between direct prediction and different reasoning intensities across decision states, illustrating state-dependent allocation of inference computation.

### B.5.2 LIBERO-PLUS

Figure 6 illustrates how the router adjusts predictive reasoning throughout execution on LIBERO-Plus. Rather than following a fixed inference pattern, the policy changes both whether reasoning is used and the selected reasoning intensity $(A, V)$ as the task progresses.

Across the visualized episodes, direct action prediction is sufficient for some decision states, while others are assigned additional predictive reasoning with different intensities. The selected computation therefore changes together with the current manipulation stage, allowing the policy to use more predictive processing when needed and return to direct prediction when additional reasoning is unnecessary.

### B.5.3 ROBOCASA

Figure 7 shows how the routing behavior evolves over the course of manipulation on RoboCasa. The selected inference mode changes together with the current stage of execution: some states are handled through direct action prediction, while others invoke predictive reasoning with different intensities $(A, V)$.

The visualized trajectories show that reasoning demand changes as the robot moves through different parts of the task. States involving different interaction stages can receive different reasoning intensities, and the policy can switch back to direct prediction once additional predictive computation is no longer needed. This dynamic allocation illustrates how the router adjusts predictive computation

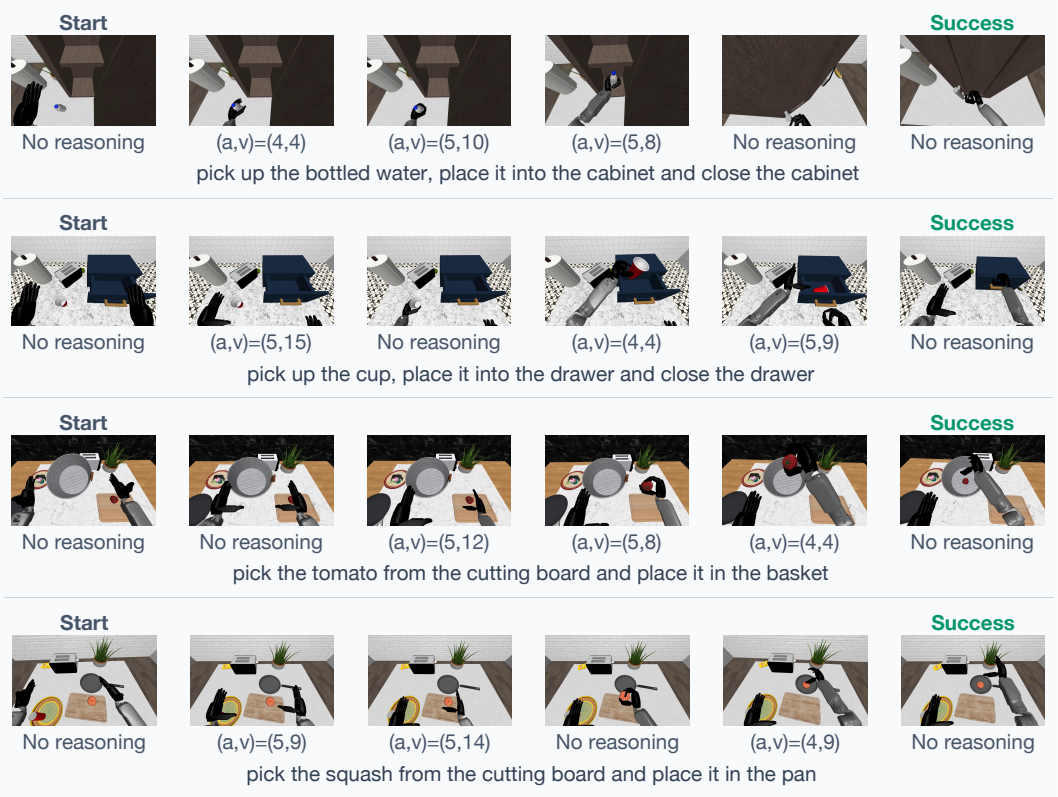


Figure 7: **Qualitative visualization of adaptive reasoning on RoboCasa.** Each row shows one successful episode from the initial state to task completion. The text below each frame indicates the routing decision at the corresponding control step: *No reasoning* denotes direct action prediction, while $(A, V)$ denotes predictive reasoning with the selected reasoning intensity. The policy dynamically switches between direct prediction and different reasoning intensities across decision states, illustrating state-dependent allocation of inference computation.

throughout the manipulation process rather than using a fixed reasoning configuration for the entire episode.

### B.6 RoboCasa GR1: Per-Task Results

Table 5 reports the per-task success rates of ThinkWAM-All-State on RoboCasa GR1 Tabletop, aggregating to the 77.7% reported in Table 3. Each task is evaluated with 25 rollouts. The 24 tasks fall into two groups: six pick-and-place tasks with fixed object-container pairings, and eighteen novel-arrangement pick-and-place tasks that pair source objects (cutting board, placemat, plate, tray) with target containers unseen in that combination during training.

### B.7 Qualitative Visualization of Adaptive Reasoning in Real-World Experiments

Figure 8 provides qualitative examples of adaptive reasoning in our real-world experiments. We visualize representative decision states from four manipulation tasks, including sequential button pressing(Press the yellow button, then press the blue button), object covering(Cover the blue block with the box), and deformable-object manipulation(Put the corn on the upper rack) and pick-and-place(Put the corn on the pink plate). For each state, we show whether the policy directly predicts actions or invokes predictive reasoning with a selected reasoning intensity $(A, V)$.

Table 5: Per-task success rates of ThinkWAM-All-State on RoboCasa GR1 Tabletop.

| Task | Rate (%) |
|---|---|
| PnPBottleToCabinetClose | 80.0 |
| PnPCanToDrawerClose | 92.0 |
| PnPCupToDrawerClose | 44.0 |
| PnPMilkToMicrowaveClose | 64.0 |
| PnPPotatoToMicrowaveClose | 88.0 |
| PnPWineToCabinetClose | 64.0 |
| PosttrainPnPNovelFromCuttingboardToBasketSplitA | 72.0 |
| PosttrainPnPNovelFromCuttingboardToCardboardboxSplitA | 80.0 |
| PosttrainPnPNovelFromCuttingboardToPanSplitA | 100.0 |
| PosttrainPnPNovelFromCuttingboardToPotSplitA | 88.0 |
| PosttrainPnPNovelFromCuttingboardToTieredbasketSplitA | 76.0 |
| PosttrainPnPNovelFromPlacematToBasketSplitA | 84.0 |
| PosttrainPnPNovelFromPlacematToBowlSplitA | 76.0 |
| PosttrainPnPNovelFromPlacematToPlateSplitA | 76.0 |
| PosttrainPnPNovelFromPlacematToTieredshelfSplitA | 56.0 |
| PosttrainPnPNovelFromPlateToBowlSplitA | 88.0 |
| PosttrainPnPNovelFromPlateToCardboardboxSplitA | 84.0 |
| PosttrainPnPNovelFromPlateToPanSplitA | 72.0 |
| PosttrainPnPNovelFromPlateToPlateSplitA | 84.0 |
| PosttrainPnPNovelFromTrayToCardboardboxSplitA | 84.0 |
| PosttrainPnPNovelFromTrayToPlateSplitA | 92.0 |
| PosttrainPnPNovelFromTrayToPotSplitA | 88.0 |
| PosttrainPnPNovelFromTrayToTieredbasketSplitA | 84.0 |
| PosttrainPnPNovelFromTrayToTieredshelfSplitA | 48.0 |
| Overall | 77.7 |

Across different tasks, the policy dynamically alternates between direct action prediction and predictive reasoning during execution. Reasoning is selectively activated at different stages of manipulation, while other states are handled without additional predictive computation. These examples further demonstrate that the learned routing strategy generalizes to real-world robot manipulation and allocates inference computation in a state-dependent manner.

### B.8 Ablation Details

Both ablations of Sec. 4.4 are evaluated on LIBERO-Plus with the same evaluation protocol as the full model. **w/o budget factor** keeps the gating factor and routes every reasoning state to the maximum intensity $(A, V) = (5, 25)$, removing the adaptive intensity selection while leaving the backbone unchanged. **w/o multi-intensity reasoning training** trains the same dual-route backbone with the synchronous shared clock only ($p_{\text{async}} = 0$), so the model never observes video–action refinement pairs off the synchronous diagonal.

The per-shift breakdown of the w/o multi-intensity variant shows that its largest degradation appears under sensor noise ($87.82 \rightarrow 83.76$, $-4.06$ points), with smaller uniform drops across the remaining shifts; this is consistent with the role of multi-intensity training in exposing the action policy to future latents at every refinement stage, which is most stressed when observations are corrupted.

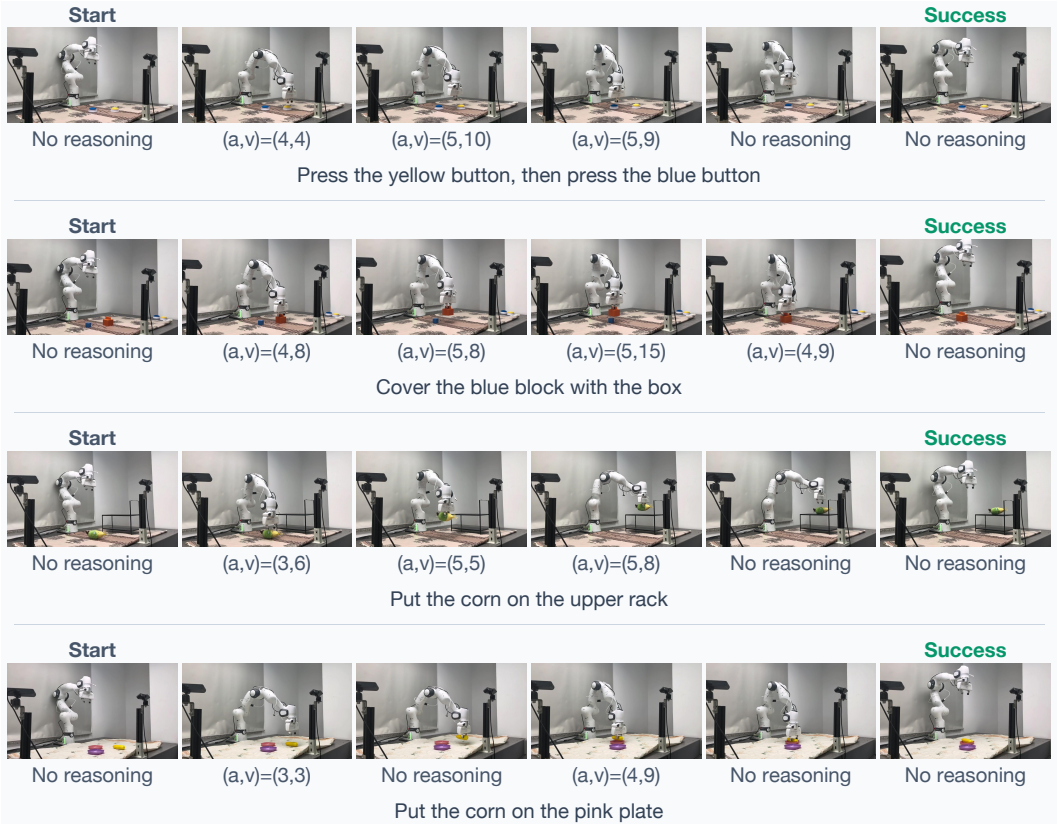


Figure 8: **Qualitative visualization of adaptive reasoning in real-world experiments.** Each row shows a successful execution of one manipulation task from start to completion. *No reasoning* denotes direct action prediction, while $(A, V)$ indicates predictive reasoning with the selected reasoning intensity. Across different manipulation tasks, the policy dynamically switches between direct prediction and different reasoning intensities according to the current decision state.