VERIVLA: ADAPTIVE TEST-TIME VERIFICATION FOR FLOW-BASED VLAS WITH LATENT WORLD MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-based Vision-Language-Action (VLA) policies are inherently stochastic: different noise seeds can produce action chunks of varying quality from the same observation. At test time, however, the policy typically commits to a single sample without an explicit estimate of its quality. In fine-grained manipulation, executing a suboptimal action can irreversibly disrupt task progress and lead to failure. To exploit policy stochasticity, we introduce a training-free two-axis trigger that uses two initial samples to measure within-chunk temporal variation and cross-sample variance, detecting critical states. Once triggered, a JEPA-based latent world model ranks candidate action chunks by predicting action-conditioned future latent states and the corresponding changes in task progress. By operating in latent space, the verifier avoids pixel reconstruction and enables efficient and accurate ranking. Across three real-world tasks (pen insertion, pipetting, and apple placement), our method raises the average completion score from 50.8% to 65.5%, with 326 ms of additional verification latency per triggered decision plus a 14 ms trigger cost per decision on a single RTX 4090. Consistent gains are also observed in simulation, where the average success rate rises from 66.4% to 73.6% in the clean setting of RoboTwin 2.0, from 62.8% to 69.6% in its randomized setting, and from 96.9% to 97.6% on LIBERO.

1 INTRODUCTION

Vision-Language-Action (VLA) models have emerged as a promising paradigm towards general robotic control (Brohan et al., 2023; Kim et al., 2024; Black et al., 2025a; Intelligence et al., 2025). In the pretraining stage, the field has evolved from VLMs autoregressively generating over discretized action tokens (Zitkovich et al., 2023; Kim et al., 2024; Pertsch et al., 2025) to more recent flow-matching action experts coupled with pretrained VLMs (NVIDIA et al., 2025; Black et al., 2025a; Intelligence et al., 2025), enabling scalable learning from diverse action trajectories. The paradigm shift was foreshadowed by diffusion-based visuomotor policies (Chi et al., 2023), which established iterative denoising as an effective means of learning multi-modal (*i.e.*, multi-peaked) action distributions. However, the stochasticity that benefits learning also introduces a unique challenge at deployment.

Given the same observation and instruction, different noise seeds yield samples of different quality during inference. However, the flow-based VLA model cannot distinguish which action chunk is most likely to succeed. During execution, a robot must commit to one sampled action chunk, which may irreversibly change the physical world. A good action chunk advances the task, while a poor one could directly lead to failure or accumulate errors that drift the trajectory out of distribution (Ross et al., 2011). We empirically show that the policy variance is non-uniform along the task, and failures tend to happen in high-variance segments of the action trajectory, as in Figure 1(a). What if we could

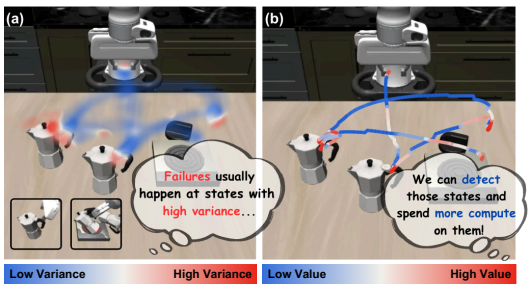


Figure 1: **Motivation.** (a) The heatmap of policy variance in the LIBERO environment. (b) Visualization of values of our proposed trigger.

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.

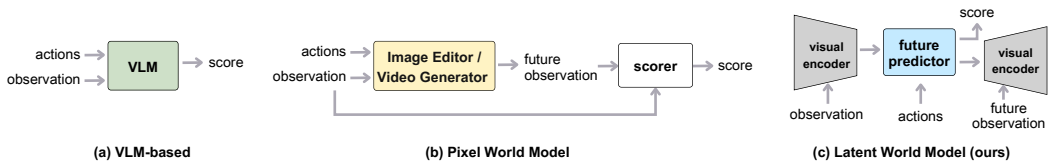


Figure 2: **Conceptual comparison between different paradigms. (a) VLM-based verifier:** Directly predict with the current observation and the action chunk. **(b) Pixel world model verifier:** Extract the score after generating the estimated future. **(c) Latent world model verifier:** Estimate the future latent and the score in a unified latent space.

detect these critical moments before committing to an irreversible action, and briefly "rewind" the decision by spending additional computation? This calls for two components: a trigger that decides *when* the states are critical and additional computation is worthwhile, and a verifier that determines *which* sampled action should be executed.

To answer the first question, we start with the policy's uncertainty. As shown in Figure 1(a), when the end-effector is close to a contact-rich interaction region, the trajectory follows a more complex motion (Pang et al., 2023), and internally, the policy uncertainty increases. We therefore combine temporal action variation and cross-sample uncertainty into a training-free two-axis trigger. Computed with two sampled action chunks, the trigger closely tracks the contact information from the environment and aligns well with the policy's intrinsic uncertainty (Figure 1(b)). This motivates us to detect with the trigger and enable the strategy of repeated sampling with verification when necessary.

Once repeated sampling is activated, the second question arises: how should candidate actions be evaluated? We argue that latent world models are a natural fit. The verifier evaluates the quality of action chunk samples by predicting the progress or relative ranking. Therefore, the model must understand world dynamics and inherit an internal representation of the states after the action chunk commits. In addition, verification is a low-dimensional problem: how much an action chunk advances the task is a scalar judgment, and most camera pixels are irrelevant to it. As a result, pixel-level world models have two pitfalls: (1) the image or video generation incurs substantial latency, making it difficult to deploy at real-time control frequencies; and (2) they solve the harder pixel generation before the easier progress estimation, making the quality of downstream verification unnecessarily dependent on generation fidelity. We show the paradigm differentiation in Figure 2. This motivates verification in a latent space, which compresses the pixel observation into compact and dynamic-rich features, where we can jointly predict the latent states and task progress.

Building on these observations, we recast action randomness not as a hazard to be suppressed but as a resource to select from, with the latent world model as a verifier. Specifically, we claim the following contributions:

1. **A latent world model as a verifier.** Rather than reconstructing pixels, we exploit the action-conditioned latent representation produced by a pretrained latent world model to estimate the progress of candidate action chunks.

2. **An empirical characterization of policy stochasticity in flow-based VLAs and the two-axis trigger.** We characterize policy stochasticity. Leveraging the temporal action variation and the cross-sampling variance, we propose a two-axis trigger that is action-based and training-free. It detects the critical states and reduces inference costs while preserving the benefits of test-time selection.

3. **A simple training, evaluation, and real-robot deployment framework.** On real-world manipulation tasks, each triggered round of repeated sampling and verification adds only 326 ms of latency. Our method improves average completion score by 14.7 percentage points over the base policy, outperforming baselines, including VLM-based verifiers and pixel-level world model verifiers.

2 Related Work

Test-time Enhancement for VLAs. Existing approaches can be broadly categorized according to whether the VLA policy is updated at deployment time. Test-time training methods, such as EVOLVE-VLA (Bai et al., 2025) and TT-VLA (Liu et al., 2026), perform online policy optimization during deployment using dense reward signals. Rather than relying on external reward estimation, T^2VLA (Chen et al., 2026b) derives intrinsic rewards from the policy's own confidence to enable self-supervised adaptation.

Another line of work keeps the pretrained VLA policy frozen and improves action quality through either gradient-free selection or gradient-based augmentation. Gradient-free methods (Hong et al., 2026; Xu et al., 2026) generally adopt a repeated-sampling-and-selection strategy in which a model-based verifier evaluates candidate actions. RoboMonkey (Kwok et al., 2025) demonstrates a test-time scaling law of autoregressive VLAs, and uses a VLM-based verifier to select the optimal action. GPC-Rank (Qi et al., 2026) follows a similar strategy, employing a lightweight pixel-generation model alongside a downstream estimator to rank candidate actions. DreamAvoid (Fan et al., 2026) combines a pretrained foundation-level, action-conditioned video generation model with an estimator and introduces a vision-based trigger to reduce computational overhead. In contrast, gradient-based methods (Qi et al., 2026) backpropagate gradients from the verifier score to refine sampled actions.

In this work, we follow the repeated-sampling-and-selection paradigm. Unlike prior verifier-based methods, which rely on VLMs or pixel-level world models followed by downstream signal extraction, our approach uses a latent world model to jointly predict dynamics and estimate task-specific progress, synergizing with an action-based trigger to allocate computation adaptively. We provide a conceptual differentiation table in Appendix A.

Latent World Models for Robotics. Building on advances in visual representation learning, prior research on latent world models has developed along two main directions. The first line of work (Sobal et al., 2025; Maes et al., 2026; Zhang et al., 2026c), represented by DINO-WM (Zhou et al., 2025) and V-JEPA 2-AC (Assran et al., 2025), utilizes pretrained visual encoders and trains dynamics predictors to perform zero-shot planning in latent space. Given a goal image, these methods employ search-based algorithms to identify suitable action proposals. The second direction combines latent world models with VLAs. VLA-JEPA (Sun et al., 2026) and Being-H0.7 (Luo et al., 2026) incorporate the prediction objective of latent world models into VLA training, while LaWAM (Chen et al., 2026a) introduces a latent subgoal as an additional conditioning signal. These integrated approaches improve overall manipulation performance and generalization.

3 Method

3.1 Problem Formulation

In the standard VLA setting, at each decision step t, a policy π_θ generates an action chunk with current observation o_t, language instruction l, and robot proprioceptive state s_t:

$$a_{t:t+H} \sim \pi_\theta(\cdot \mid o_t, l, s_t), \tag{1}$$

where $a_{t:t+H}$ denotes the H actions $a_t, \ldots, a_{t+H-1}$. Flow-matching VLAs (Intelligence et al., 2025) take random Gaussian noise as input, although this dependence is often implicit in the policy notation. We make it explicit:

$$a^{(i)}_{t:t+H} \sim \pi_\theta(\cdot \mid o_t, l, s_t, N_i), \tag{2}$$

where N_i denotes the random seed used to generate the Gaussian noise, with each sampling index i mapped bijectively to a unique seed. Repeated sampling therefore produces a set of diverse action chunks $\{a^{(i)}_{t:t+H}\}_{i=1}^{K}$.

Executing an action chunk changes the task progress, therefore we define the quality of an action chunk as its change in progress, referred to as **delta progress**,

$$\Delta p = p_{t+H} - p_t, \tag{3}$$

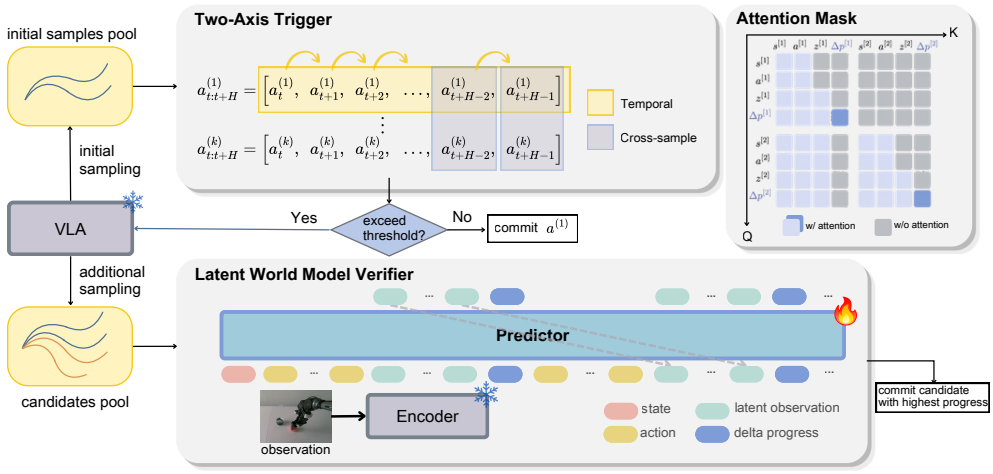


Figure 3: **(a) Overall pipeline.** The VLA policy generates two initial action chunk samples, based on which the two-axis trigger decides whether to activate repeated sampling. If triggered, the verifier evaluates K candidate chunks and selects the one with the highest estimated delta progress for execution. (b) **Verifier Architecture.** The JEPA-based latent world model autoregressively takes states, actions and observations as input and predicts delta progress in the latent space.

where p_t is the task progress at timestep t. Compared with absolute progress, delta progress matches the effect of the action chunk, making it a more direct and intuitive target for action evaluation (Liang et al., 2026a).

3.2 Overview

As illustrated in Figure 3, our framework comprises two tightly related components: Two-Axis Trigger and Latent World Model Verifier. Based on initial action chunk samples, the trigger identifies decision steps at which the policy is likely to produce unreliable actions and activates verification only in these challenging situations.

When the trigger is inactive, no additional action chunks are sampled, and the initial sample $a^{(1)}_{t:t+H}$ is executed directly. When the trigger is activated, the policy generates additional samples to expand the candidate action pool. The verifier V_ϕ then predicts the delta progress associated with each candidate:

$$\left(\hat{z}^{(i)}_{t+H}, \Delta\hat{p}^{(i)}\right) = V_\phi\left(s_t, a^{(i)}_{t:t+H}, o_t\right), \tag{4}$$

where $\hat{z}^{(i)}_{t+H}$ denotes the predicted visual latent at time step $t+H$, and $\Delta\hat{p}^{(i)}$ denotes the estimated delta progress of the sample i. The action chunk with the largest estimated delta progress is subsequently executed.

3.3 Two-Axis Trigger

Following prior work (Hu et al., 2024; Fan et al., 2026) and our empirical observations in Figure 1, an effective trigger should satisfy two criteria: (C1) activate in rich-contact regions, where manipulation is inherently more complex and failures concentrate; and (C2) activate at states where repeated sampling produces diverse candidate actions, providing the verifier with meaningful alternatives to choose from.

At timestep t, we draw k action chunks $\{a^{(i)}_{t:t+H}\}_{i=1}^{k}$ from the VLA policy, where k is chosen to be much smaller than the full sampling budget K used during verification, so that the trigger incurs acceptable overhead. To avoid bias toward fast-moving stages, each action chunk is first normalized by its average speed. The sampled action chunks can be viewed along two complementary axes, as shown in Figure 3. The temporal axis of each action chunk captures C1, while the sample axis across repeated action chunks captures C2. Along the temporal axis, we measure the mean squared

changes between consecutive actions within each sampled chunk, defined as

$$T_{\text{temp}}(s_t) = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{H-1}\sum_{j=1}^{H-1}\|a_{t+j}^{(i)} - a_{t+j-1}^{(i)}\|^2. \tag{5}$$

Along the sample axis, we adopt the cross-sample variance, as proposed in Adaflow (Hu et al., 2024), measuring how much the policy diverges across sampled chunks:

$$T_{\text{cross}}(s_t) = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{H}\sum_{j=0}^{H-1}\left\|a_{t+j}^{(i)} - \bar{a}_{t+j}\right\|^2, \tag{6}$$

where $\bar{a}_{t+j} = \frac{1}{k}\sum_{i=1}^{k} a_{t+j}^{(i)}$ is the mean action at step $t+j$ over k samples. In practice, we use $k = 2$ as two samples are sufficient to reflect the cross-sample divergence.

Before deployment in a new environment, we perform a lightweight calibration by rolling out the base policy and compute T_{temp} and T_{cross} at every timestep for a small number of episodes to collect a reference set. The collected statistics form two empirical reference distributions for the environment. During execution, following the intuition of rank-based conformal calibration, the trigger values $T_{\text{temp}}(s_t)$ and $T_{\text{cross}}(s_t)$ are converted into their corresponding percentiles, $\tau_{\text{temp}}(s_t)$ and $\tau_{\text{cross}}(s_t)$, within their respective reference distributions. The overall trigger score is then defined as

$$\tau(s_t) = \max\big(\tau_{\text{temp}}(s_t),\ \tau_{\text{cross}}(s_t)\big). \tag{7}$$

The trigger fires when $\tau(s_t) > p$, where p is the only hyperparameter that decides how strict the trigger is. Since the trigger correlates well with the critical states, calibrating p is straightforward and few rollout episodes are sufficient.

3.4 Latent World Model Verifier

When the Two-Axis Trigger is activated, we perform additional sampling to obtain the remaining $K - k$ action chunk candidates, resulting in a total of K candidates, $\{a_{t:t+H}^{(i)}\}_{i=1}^{K}$. Unlike search-based planning with latent world model (Zhou et al., 2025; Assran et al., 2025), our verifier ranks action chunk candidates by predicting delta progress without the need for a goal image. This makes it compatible with real-time VLA inference.

Architecture. We instantiate the verifier using the Joint-Embedding Predictive Architecture (JEPA) world model framework (Assran et al., 2025; Mur-Labadia et al., 2026), which consists of a pretrained visual encoder and an action-conditioned autoregressive predictor. The encoder compresses the current observation into a latent representation, while the predictor models how the latent state evolves conditioned on robot actions.

To adapt the pretrained world model to our setting, we introduce embodiment-specific state and action encoders together with sub-chunk rollouts. For each embodiment, the encoders project the raw robot states and actions into the predictor embedding space. Each action chunk with horizon H is uniformly divided into M sub-chunks, enabling multi-step latent rollouts. This decomposition provides denser supervision while reducing inference overhead and mitigating error accumulation compared with rolling out individual actions.

Building upon the latent world model, we introduce a delta progress prediction branch. Unlike latent state prediction, which models task-agnostic environment dynamics, delta progress prediction estimates how much the executed action sub-chunk advances the specified task. Consequently, it requires task-specific information beyond the current observation, robot state, and action. For example, the same state transition corresponds to positive progress for the task "closing the closet", but negative progress for "opening the closet". We therefore introduce a task-specific learnable query token to condition delta progress prediction.

Following the autoregressive paradigm, the predictor first imagines the next latent state $\hat{z}$ and then predicts the corresponding progress increment $\Delta\hat{p}$ for each input action sub-chunk. As illustrated in Figure 3, an attention mask prevents latent state prediction from attending to the task query. Otherwise, task-specific information could leak into the latent dynamics model, encouraging the

predictor to exploit task semantics instead of learning task-agnostic environment dynamics. This design cleanly separates task-conditioned progress estimation from task-agnostic latent dynamics modeling, allowing the pretrained world model to be adapted for verification while preserving its original modeling objective.

Training. The verifier is optimized jointly for latent prediction and progress estimation, while the progress estimation serves as the primary goal. For each action sub-chunk, the delta progress loss is computed using Huber loss and rank loss,

$$\mathcal{L}_{\text{prog}} = \sum_{i=1}^{M} \big(\mathcal{L}_{\text{i, huber}} + \lambda_{\text{rank}} \mathcal{L}_{\text{i, rank}} \big), \tag{8}$$

where the Huber loss improves the regression accuracy, while the rank loss preserves the relative ordering within tasks to avoid the trivial averaging solution. Details are provided in Appendix B. The absolute progress is labeled autonomously with Robometer (Liang et al., 2026a), and then the delta progress is calculated based on the result. The visual prediction loss is defined following V-JEPA 2-AC (Assran et al., 2025),

$$\mathcal{L}_{\text{vis}} = \sum_{i=1}^{M} \| \hat{z}_{t+i \cdot \frac{H}{M}} - z_{t+i \cdot \frac{H}{M}} \|_1. \tag{9}$$

The total training loss is computed as

$$\mathcal{L} = \mathcal{L}_{\text{prog}} + \lambda_{\text{vis}} \mathcal{L}_{\text{vis}}. \tag{10}$$

We use both teacher forcing and self-forcing (Huang et al., 2025) during the training stage to balance learning efficiency and inference distribution alignment.

Inference. At inference time, the predictor conditions exclusively on its previously predicted latent frames and autoregressively rolls out over the M sub-chunks of each candidate action chunk. The verifier thus produces M delta progress estimates for each candidate. The robot then executes the action chunk with the largest cumulative predicted progress, computed as the sum of these M estimates.

To keep verification practical for real-world deployment, three latency optimizations are applied cumulatively: (O1) reusing KV cache throughout the autoregressive rollout; (O2) optimizing the inference pipeline by overlapping independent computations, including performing latent feature encoding in parallel with action generation of the VLA policy and reusing the VLM KV cache, and (O3) PyTorch-level optimization, including pre-compilation.

4 Experiments

Our experiments address the questions in Section 1: *when* verification is needed and *how* to design the verifier. We evaluate task performance and deployment latency on real-world fine-grained manipulation, efficiency–performance trade-offs and generalization on RoboTwin 2.0, and improvements across two base policies and 40 tasks on LIBERO. Ablations assess the contributions of latent world modeling and predictor pretraining to progress estimation.

Trigger diagnostics. The trigger-only evaluation, including the Status AUC and Variance Correlation metrics, baselines, and detailed results, is reported in Appendix C. We retain Variance Correlation because it measures a central design intuition of our method—that difficult states should expose diverse candidate actions—even though the statistic is naturally related to the trigger construction. The main experiments below therefore focus on the end-to-end verifier and its task performance.

4.1 Real-world Experiments

Setup. We conduct real-world experiments on the AgileX PiperX robotic embodiment, equipped with one front camera (RealSense D455) and two wrist cameras (RealSense D435). To evaluate

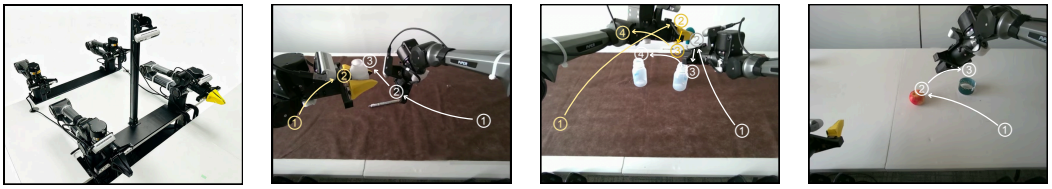


Figure 4: **Real-world experimental setup.** We conduct the real-world experiments on an AgileX PiperX dual arm robot, and design three fine-grained tasks, namely, pen insertion, pipetting and apple placement.

Table 1: **Comparison of different methods across tasks and metrics.**

Method	Validation Metrics		Completion Score (%) ↑			
	Rank Loss ↓	Huber Loss ↓	Pen Insertion	Pipetting	Apple Placement	Average
$\pi_{0.5}$	–	–	45.0	52.5	55.0	50.8
+ RoboMonkey	1.19	0.37	49.0	55.5	63.5	56.0
+ GPC-Rank	1.18	0.30	46.0	54.5	58.0	52.8
+ **Ours**	**1.09**	**0.25**	60.0	63.5	73.0	**65.5**

methods in different situations, we design three tasks with varying arm configurations (1 single-arm, 2 dual-arm), and key challenges. Namely, there are pen insertion (pick up the pen and insert it into the yogurt bottle), apple placement (pick up the apple and place it into the container), and pipetting (transfer liquid from the right bottle to the left with a pipette), as in Figure 4. The detailed task configurations are provided in Appendix C.

Implementation Details. We first collect 50 demonstration episodes to fine-tune the base policy, following common practice. Then we automatically roll out another 50 episodes, label them using Robometer (Liang et al., 2026a), and train one verifier for all tasks in the environment on a single H800 GPU. The rank-loss groups are formed from ordinary rollout transitions rather than multiple counterfactual candidates from one identical state (see Appendix B). Each reported evaluation uses 50 trials, one random seed, and one trained model. The same protocol is used in the simulation experiments. The test-time deployment is based on one RTX 4090 GPU, with the policy model and the verifier running on the same device. During testing, we set the trigger threshold to align with contact-rich phases. We use $k = 2$ samples for Two-Axis Trigger and $K = 10$ samples for the verifier.

Metrics. We utilize three metrics: two validation losses and one closed-loop completion score. Specifically, *rank loss* measures model's ability to rank the action chunk samples within one task. *Huber loss* measures how well the model regresses the delta progress value. *Completion score* reflects the system's closed-loop performance on the real-world tasks, where each stage is assigned a predefined score. For more details, please refer to Appendix D.

Baselines. We compare against the following methods: 1) Base policy ($\pi_{0.5}$) with single sampling; 2) VLM-based verifier following the setting in RoboMonkey (Kwok et al., 2025), adapted the training goal with delta progress estimation instead of the original preference ranking; 3) Pixel-level world model verifier following GPC-Rank (Qi et al., 2026).

Results. Table 1 shows a 14.7 percentage-point improvement in average completion score over the base policy. Our method outperforms the compared baselines. The three optimizations reduce additional verification latency to 326 ms per triggered decision, on top of the 14 ms trigger cost per decision (Figure 5). RoboMonkey performs competitively on coarse localization (e.g., apple placement) but is weaker on fine-grained manipulation, consistent with its lack of explicit dynamics modeling. GPC-Rank performs well when scoring real observations but degrades on predicted frames, suggesting that generation quality limits verification.

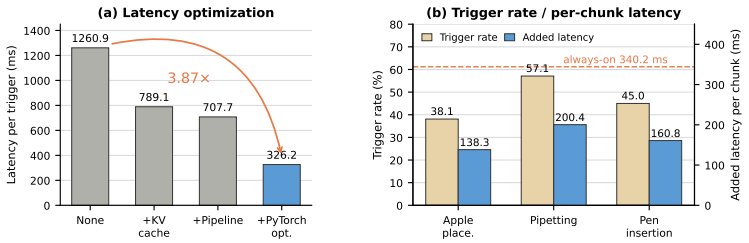


Figure 5: **Latency analysis.** (a) Additional verification latency for 10 candidates under successive optimizations; KV caching gives the largest reduction. (b) Trigger rate and average added latency per action chunk on the three real-world tasks.

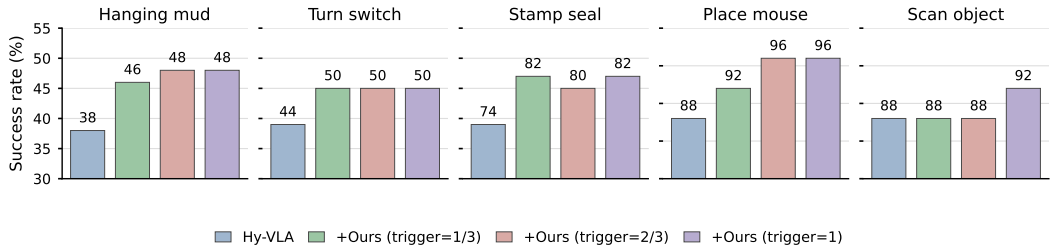


Figure 6: **Results on RoboTwin 2.0 clean with Sweeping Threshold.** For trigger $= r$, we adjust a general trigger threshold for all the tasks, so that approximately an r fraction of decision steps trigger verification.

Table 2: **Results on RoboTwin 2.0 randomized with always-on verifier.** Our method boosts the base policy on 4 of 5 tasks.

Method	Mug	Switch	Seal	Mouse	Scan	Avg.
Hy-VLA	16.0	50.0	72.0	80.0	**96.0**	62.8
+ Ours	**26.0**	**56.0**	**78.0**	**92.0**	**96.0**	**69.6**

Table 3: **LIBERO results.** Our method boosts the base policy, especially on LIBERO-10, the task suite that has not been saturated.

Method	Spatial	Object	Goal	10	Avg.
$\pi_{0.5}$	**98.8**	98.2	98.0	92.4	96.9
+ Ours	**98.8**	**98.8**	**98.6**	**94.0**	**97.6**
StarVLA-GR00T	98.0	**99.4**	96.4	91.0	96.2
+ Ours	**98.4**	99.0	**97.8**	**93.6**	**97.2**

4.2 Simulation Results

To provide reproducible and quantifiable results, we evaluate the combination of the trigger and the verifier in simulation environments, including RoboTwin 2.0 (Chen et al., 2025) and LIBERO (Liu et al., 2023).

RoboTwin 2.0. RoboTwin 2.0 is a benchmark for bimanual manipulation with both clean and randomized settings. For RoboTwin 2.0, we choose the base policy to be Hy-Embodied-0.5-VLA (Zhang et al., 2026b), a recent state-of-the-art VLA model. We evaluate our method on 5 fine-grained tasks that result in relatively low success rate. In the clean setting, we sweep a trigger threshold for all tasks; the resulting clean-setting threshold sweep and trigger-efficiency analysis are shown in Figure 6. In the randomized setting, we keep verification always on. The verifier is trained only on clean-environment rollouts and is applied without adaptation to randomized configurations. Table 2 shows that our verifier can boost the base policy in randomized settings by 6.8 percentage points, showcasing generalization capability.

LIBERO. We evaluate our method on LIBERO, using $\pi_{0.5}$ and StarVLA-GR00T (Community, 2026) as the base policy. The verifier is trained from $\pi_{0.5}$ rollouts and then applied to both base policies; it is not jointly trained on rollouts from both policies. The results indicate that our method

Table 4: **Ablation on latent world modeling.** Completion score is averaged over 3 real-world tasks.

Method	Rank ↓	Huber ↓	Pen	Pipetting	Apple	Completion Score (%) ↑
w/o WM	1.21	0.26	52.0	55.5	58.0	55.2
w/ WM	**1.09**	**0.25**	**60.0**	**63.5**	**73.0**	**65.5**

Table 5: **Ablation on predictor pretraining.** The encoder remains pretrained in both variants.

Method	Rank ↓	Huber ↓	Pen	Pipetting	Apple	Completion Score (%) ↑
w/o pretrain	1.17	0.25	54.0	61.0	63.5	59.5
w/ pretrain	**1.09**	**0.25**	**60.0**	**63.5**	**73.0**	**65.5**

improves performance on LIBERO-10 suite, while maintaining strong performance on other suites. This illustrates that our single verifier fits 40 tasks and boosts performance across two policies.

4.3 Ablation Study

In this section, we discuss the design of the verifier, and specifically answer the following question: Do *latent world modeling* and *pretrained latent dynamics* improve progress estimation? We conduct two sets of ablation studies. First, we assess the role of latent world modeling by removing the visual prediction objective. Second, we investigate whether pretraining the predictor on the general-purpose DROID dataset (Khazatsky et al., 2024) leads to better performance.

4.3.1 Effect of World Modeling.

Does the test-time improvement stem from latent world modeling, or merely from a strong visual encoder? To answer the question, we implement a variant predicting solely the delta progress, instead of jointly predicting with visual latent. Therefore, the variant degenerates into an autoregressive transformer without world modeling. We train the variant on the same three real-world tasks. As shown in Table 4, the completion score decreases from 65.5% to 55.2%. These results suggest that without the dense world modeling objective, the model struggles to learn informative dynamics, leading to substantially weaker performance.

4.3.2 Effect of Predictor Pretraining

Our verifier builds on a frozen pretrained V-JEPA 2 encoder, which provides a rich visual prior learned from web-scale video data. In addition, the action-conditioned predictor is pretrained on 23k trajectories from the DROID dataset, offering a strong dynamics prior for robotic manipulation. To assess whether this predictor pretraining is beneficial for knowledge transfer, we train a variant without pretraining where the predictor is initialized randomly. In Table 5, the variant shows lower completion scores on real-world tasks, especially on the tasks with stronger generalization requirements, *e.g.* apple placement. In contrast, on the pipetting task, where bottle locations remain almost fixed and generalization is less critical, the performance degradation is relatively minor.

5 Conclusion

We introduced adaptive test-time verification for flow-based VLAs. A training-free two-axis trigger uses temporal action variation and cross-sample disagreement to allocate additional sampling. A JEPA-based latent world model then ranks candidates by predicting future latents and delta task progress, without reconstructing pixels or requiring goal images. Experiments show gains across real-world and simulation tasks and multiple base policies. On three real-world tasks, the average completion score rises from 50.8% to 65.5%, with 326 ms of additional verification latency per triggered decision. Ablations support the contributions of latent dynamics modeling and predictor pretraining.

AI USE STATEMENT

In this work, we used generative AI tools to assist with code writing, polish and refine the manuscript, and provide feedback on the research methodology and experimental design. We have reviewed all AI-assisted work and verified the accuracy and appropriateness of the generated code, text, and suggestions. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 describes the trigger, verifier architecture, and training objectives. Section 4 specifies the experimental settings and baselines. Appendix B provides loss definitions and training hyperparameters, Appendix C describes the trigger evaluation, and Appendix D documents task scores and computational costs.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Ziang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking Failure Modes of Generative Policies: Runtime Monitoring of Consistency and Progress. In *CoRL*, 2024. arXiv:2410.04640.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning, 2025. URL https://arxiv.org/abs/2506.09985.

Zechen Bai, Chen Gao, and Mike Zheng Shou. EVOLVE-VLA: Test-Time Training from Environment Feedback for Vision-Language-Action Models, 2025. URL https://arxiv.org/abs/2512.14666.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *RSS*, 2025a.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. In *NeurIPS*, 2025b. arXiv:2506.07339.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, 2023.

Jialei Chen, Kai Wang, Kang Chen, Shuaihang Chen, Feng Gao, Wenhao Tang, Zhiyuan Li, Weilin Liu, Zhuyu Yao, Boxun Li, Yuanbo Xu, and Chao Yu. LaWAM: Latent World Action Models for Efficient Dynamics-Aware Robot Policies, 2026a. URL https://arxiv.org/abs/2606.15768.

Siyao Chen, Jiakang Yuan, Jiaxin Wang, and Tao Chen. Trust Your Instincts: Confidence-Driven Test-Time RL for Vision-Language-Action Models, 2026b. URL https://arxiv.org/abs/2606.29892.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A Scalable Data Generator and Benchmark with Strong Domain Randomization for Robust Bimanual Robotic Manipulation, 2025. URL https://arxiv.org/abs/2506.18088.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. In *RSS*, 2023.

StarVLA Community. StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing, 2026. URL https://arxiv.org/abs/2604.05014.

Xianzhe Fan, Yuxiang Lu, Shenyuan Gao, Xiaoyang Wu, Ruihua Han, Manling Li, and Hengshuang Zhao. DreamAvoid: Critical-Phase Test-Time Dreaming to Avoid Failures in VLA Policies, 2026. URL https://arxiv.org/abs/2605.11750.

Xiangdong Feng, Yuxuan Cheng, Chen Shi, Boyao Han, Yuxuan Yan, Yitong Hong, Zhuotao Tian, and Li Jiang. Denoising Tells When to Replan: Denoising-Variance Adaptive Chunking for Flow-Based Robot Policies. *arXiv preprint arXiv:2606.03847*, 2026.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40041–40076, 2025.

Yining Hong, Huang Huang, Manling Li, Li Fei-Fei, Leonidas Guibas, Jiajun Wu, and Yejin Choi. Learning from Trials and Errors: Reflective Test-Time Planning for Embodied LLMs, 2026. URL https://arxiv.org/abs/2602.21198.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. *NeurIPS*, 2024.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *NeurIPS*, 2025.

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.

Suhyeok Jang, Dongyoung Kim, Changyeon Kim, Youngsuk Kim, and Jinwoo Shin. Verifier-free Test-Time Sampling for Vision-Language-Action Models. In *ICLR*, 2026. arXiv:2510.05681.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An Open-Source Vision-Language-Action Model. In *CoRL*, 2024.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. RoboMonkey: Scaling Test-Time Sampling and Verification for Vision-Language-Action Models. In *CoRL*, 2025.

Jacky Kwok, Xilun Zhang, Mengdi Xu, Yuejiang Liu, Azalia Mirhoseini, Chelsea Finn, and Marco Pavone. Scaling verification can be more effective than scaling policy learning for vision-language-action alignment. *arXiv preprint arXiv:2602.12281*, 2026.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S. Huang, Luke Zettlemoyer, Dieter Fox, Yu Xiang, Anqi Li, Andreea Bobu, Abhishek Gupta, Stephen Tu, Erdem Biyik, and Jesse Zhang. Robometer: Scaling General-Purpose Robotic Reward Models via Trajectory Comparisons, 2026a. URL https://arxiv.org/abs/2603.02115.

Yuanchang Liang, Xiaobo Wang, Kai Wang, Shuo Wang, Xiaojiang Peng, Haoyu Chen, David Kim Huat Chua, and Prahlad Vadakkepat. Adaptive Action Chunking at Inference-time for Vision-Language-Action Models, 2026b. URL https://arxiv.org/abs/2604.04161.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *NeurIPS*, 2023.

Changyu Liu, Yiyang Liu, Taowen Wang, Qiao Zhuang, James Chenhao Liang, Wenhao Yang, Renjing Xu, Qifan Wang, Dongfang Liu, and Cheng Han. On-the-Fly VLA Adaptation via Test-Time Reinforcement Learning, 2026. URL https://arxiv.org/abs/2601.06748.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional Decoding: Improving Action Chunking via Guided Test-Time Sampling. In *ICLR*, 2025. arXiv:2408.17355.

Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, Haiweng Xu, Chaoyi Xu, Ziheng Xi, Yuhui Fu, and Zongqing Lu. Being-H0.7: A Latent World-Action Model from Egocentric Videos, 2026. URL https://arxiv.org/abs/2605.00078.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorld-Model: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels, 2026. URL https://arxiv.org/abs/2603.19312.

Lorenzo Mur-Labadia, Matthew Muckley, Amir Bar, Mido Assran, Koustuv Sinha, Mike Rabbat, Yann LeCun, Nicolas Ballas, and Adrien Bardes. V-JEPA 2.1: Unlocking Dense Features in Video Self-Supervised Learning, 2026. URL https://arxiv.org/abs/2603.14482.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering Your Generalists: Improving Robotic Foundation Models via Value Guidance. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=6FGlpzC9Po.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots, 2025. URL https://arxiv.org/abs/2503.14734.

Tao Pang, H. J. Terry Suh, Lujie Yang, and Russ Tedrake. Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-dynamic Contact Models, 2023. URL https://arxiv.org/abs/2206.10787.

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, 2025. URL https://arxiv.org/abs/2501.09747.

Han Qi, Haocheng Yin, Aris Zhu, Yilun Du, and Heng Yang. Inference-time enhancement of generative robot policies via predictive world modeling. *IEEE Robotics and Automation Letters*, 2026.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, 2011. URL https://arxiv.org/abs/1011.0686.

Florian Seligmann, Emiliyan Gospodinov, Enes Ulas Dincer, and Gerhard Neumann. VLA-FAIL: Efficient Task Failure Detection for Finetuned Vision-Language-Action Models. *arXiv preprint arXiv:2606.21386*, 2026.

Uladzislau Sobal, Wancong Zhang, Kyunghyun Cho, Randall Balestriero, Tim GJ Rudner, and Yann LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *NeurIPS*, 2025.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing Vision-Language-Action Model with Latent World Model, 2026. URL https://arxiv.org/abs/2602.10098.

Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From Foresight to Forethought: VLM-In-the-Loop Policy Steering via Latent Alignment. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2502.01828.

Changhua Xu, En Yu, Junyu Xuan, and Jie Lu. VGAS: Value-Guided Action-Chunk Selection for Few-Shot Vision-Language-Action Adaptation, 2026. URL https://arxiv.org/abs/2602.07399.

Haoran Zhang, Yifu Lu, Boyang Wang, Xuhui Kang, Yen-Ling Kuo, Zezhou Cheng, Mengdi Wang, and Odest Chadwicke Jenkins. Foresight: Failure Detection for Long-Horizon Robotic Manipulation with Action-Conditioned World Model Latents. *arXiv preprint arXiv:2606.23085*, 2026a.

He Zhang, Lingzhu Xiang, Haitao Lin, Zeyu Huang, Minghui Wang, Dingyan Zhong, Yubo Dong, Yihao Wu, Yongming Rao, Dongsheng Zhang, Wanjia He, Ling Chen, Kai Huang, Jiahao Chen, Sichang Su, Xumin Yu, Ziyi Wang, Chengwei Zhu, Xiao Teng, Yuchun Guo, Yufeng Zhang, Yuandong Liu, Rui Wang, Zisheng Lu, Han Hu, and Zhengyou Zhang. Hy-Embodied-0.5-VLA: From Vision-Language-Action Models to a Real-World Robot Learning Stack, 2026b. URL https://arxiv.org/abs/2606.14409.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, and Nicolas Ballas. Hierarchical Planning with Latent World Models, 2026c. URL https://arxiv.org/abs/2604.03208.

Chen Zhao, Zhuoran Wang, Haoyang Li, Shifeng Bao, Guanlin Li, Youhe Feng, Yang Li, Jie Tang, and Jing Zhang. Action Draft and Verify: A Self-Verifying Framework for Vision-Language-Action Model, 2026. URL https://arxiv.org/abs/2603.18091.

Gaoyue Zhou, Hengkai Pan, Yann Lecun, and Lerrel Pinto. DINO-WM: World Models on Pre-trained Visual Features enable Zero-shot Planning. In *ICML*, 2025.

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 *CoRL*, pp. 2165–2183, 2023.

A Extended Related Work

In this section, we provide an extended review of prior work.

A.1 Test-time Verification

Existing methods differ in how the verifier is constructed. *Model-based verifiers* train a separate scoring model: RoboMonkey (Kwok et al., 2025) fine-tunes a 7B VLM on synthetic action preferences, CoVer (Kwok et al., 2026) argues that scaling such verifiers beats scaling policy learning, and V-GPS (Nakamoto et al., 2024) re-ranks samples with an offline-RL value function. *Model-free* methods instead read the signal off the policy itself, via masked-condition divergence (Jang et al., 2026) or proximity to the discrete action tokens (Zhao et al., 2026); both measure agreement rather than progress, and neither reasons about the physical consequences of a candidate. *World-model verifiers* close this gap by imagining outcomes, but do so in two stages: GPC-RANK (Qi et al., 2026) and DreamAvoid (Fan et al., 2026) synthesize future pixels before scoring them, while FOREWARN (Wu et al., 2025) rolls out latents and then aligns a per-task finetuned VLM to narrate them. VeriVLA keeps the world model but removes the second stage, reading delta progress directly from predicted latents. Combined with a two-axis action-based trigger, verification is invoked adaptively.

Table 6: **Comparison of test-time verification methods.**

Method	Trigger	World Modeling	Verification Pipeline	Prediction Target
RoboMonkey (Kwok et al., 2025)	always-on	×	one-stage	action preference
FOREWARN (Wu et al., 2025)	always-on	✓	two-stage	latent, VLM narration
GPC-RANK (Qi et al., 2026)	always-on	✓	two-stage	pixel, reward
ADV (Zhao et al., 2026)	always-on	×	one-stage	proximity to discrete action tokens
MG-Select (Jang et al., 2026)	always-on	×	one-stage	masked-condition divergence
DreamAvoid (Fan et al., 2026)	vision-based	✓	two-stage	pixel, delta progress
VeriVLA	action-based	✓	one-stage	latent, delta progress

A.2 Other Test-time Enhancement

In addition to test-time verification, we provide an overview of other existing test-time enhancement methods, including test-time RL, adaptive action chunking, and online monitoring.

Test-time training and RL. Test-time training methods, such as EVOLVE-VLA (Bai et al., 2025) and TT-VLA (Liu et al., 2026), perform online policy optimization during deployment using dense reward signals. Rather than relying on external reward estimation, T^2VLA (Chen et al., 2026b) derives intrinsic rewards from the policy's own confidence to enable self-supervised adaptation. These methods update policy parameters and therefore require reward access, gradient computation, and tolerance for transient regression during adaptation.

Adaptive action chunking. A parallel line of work asks not *which* action to take but *how* to emit it. BID (Liu et al., 2025) resamples action chunks under backward coherence and forward contrast to trade off consistency against reactivity. RTC (Black et al., 2025b) targets the latency gap by generating the next chunk while the current one executes, freezing the actions guaranteed to run and inpainting the remainder. AAC (Liang et al., 2026b) adaptively sets the chunk size from action entropy, and DVAC (Feng et al., 2026) triggers replanning from denoising variance. The underlying intuition traces back to AdaFlow (Hu et al., 2024), which links the conditional

variance of the flow-matching loss to ODE discretization error and reduces to a one-step generator when the action distribution is unimodal. Our trigger shares this variance-based intuition, but these methods reshape the execution or decoding schedule of a single trajectory and never assess whether the emitted actions make task progress; our trigger instead gates a verifier that can reject and replace a chunk rather than merely smoothing or resizing it.

Online monitoring and failure detection. Runtime monitors aim to predict failure before it becomes irrecoverable. Sentinel (Agia et al., 2024) splits failures into erratic modes, detected by statistical temporal consistency of action-chunk distributions (STAC), and task-progression modes, detected via VLM video QA; the two detectors together catch 18% more failures than either alone. SAFE (Gu et al., 2025) shows that VLA internal features carry task-generic success/failure information and trains a lightweight scalar failure predictor on them, using conformal prediction for a calibrated time-varying threshold; VLA-FAIL (Seligmann et al., 2026) pairs a last-layer Mahalanobis distance with an action-chunk consistency term to remove the need for failure data. Closest to our setting, Foresight (Zhang et al., 2026a) extracts action-conditioned latents from V-JEPA 2-AC and predicts a per-timestep failure score, showing that world-model features generalize across policies better than policy-internal ones.

B Verifier Training

B.1 Loss Function

The Huber loss is defined as

$$\mathcal{L}_{\text{Huber}} = \frac{1}{N}\sum_{i=1}^{N} \ell_\delta\Big(\Delta\hat{p}^{(i)} - \Delta p^{(i)}\Big),$$

where

$$\ell_\delta(e) = \begin{cases} \frac{1}{2}e^2, & |e| \le \delta, \\ \delta\left(|e| - \frac{1}{2}\delta\right), & |e| > \delta, \end{cases}$$

and N is the number of regression samples in the training set.

The rank loss is defined as

$$\mathcal{L}_{\text{rank}} = \frac{1}{|P|}\sum_{(i,j)\in P} \max\Big(0,\, -(\Delta\hat{p}^{(i)} - \Delta\hat{p}^{(j)}) + (\Delta p^{(i)} - \Delta p^{(j)})\Big),$$

inspired by the preference estimation in RoboMonkey (Kwok et al., 2025) and adapted from DreamAvoid (Fan et al., 2026), where P is the set of valid sample pairs that come from the same task. Our collected training data consists of ordinary policy-rollout transitions: a group does not contain multiple counterfactual action chunks generated from exactly the same observation and robot state. The progress labels are supplied by Robometer. Thus, the rank loss supervises relative progress among rollout samples from the same task, rather than providing direct same-state candidate-ranking supervision. At test time, the verifier is applied to candidates sharing the current observation; this transfer relies on the learned action-conditioned dynamics and progress representation. We do not claim that the present data provides direct counterfactual supervision for this transfer. In practice, we set $\delta = 1$.

B.2 Hyperparameter

We list the training hyperparameters in Table 7.

C Trigger Diagnostics

To evaluate whether a trigger signal at a state $V_{\text{trigger}}(s_t)$ identifies moments that genuinely require more compute, we construct a reference label $\text{crit}(t) \in [0, 1]$ that is independent of the action or vision. We find the *critical moments* from privileged simulator state (object/gripper pose, joint

Table 7: **Training hyperparameters and resource configuration.** The table specifically shows the real-world adaptation experiments. The actual training epochs and duration can largely depend on the scale of the dataset.

Hyperparameter	**Value**
Learning rate	5×10^{-4}
Weight decay	0.04
Batch size	32
Optimizer	AdamW
JEPA loss weight	1
rank loss weight	1
Crop size	256×256
Patch size	16
Tubelet size	2
sub-chunk number	5
Resource	**Specification**
GPU model	1 NVIDIA H800 (80GB)

angles, contact) using a rule: we mark frames where gripper-object or object-object contact state flips (grasp / release) for any task-relevant object. For each support moment, we assign a Gaussian kernel function. The *critical index* for each frame is then computed as the sum of all overlapping kernel contributions, clipped to a maximum of 1.

After labeling an episode with the continuous critical index, we compute the trigger value for each chunk $a_{t:t+H}$ and take the maximum critical index within that window. Thresholding the latter at 0.5 yields the binary label, and the trigger value serves as the score; we then compute the ROC curve and the Status AUC. We visualize two tasks from LIBERO and RoboTwin for reference:

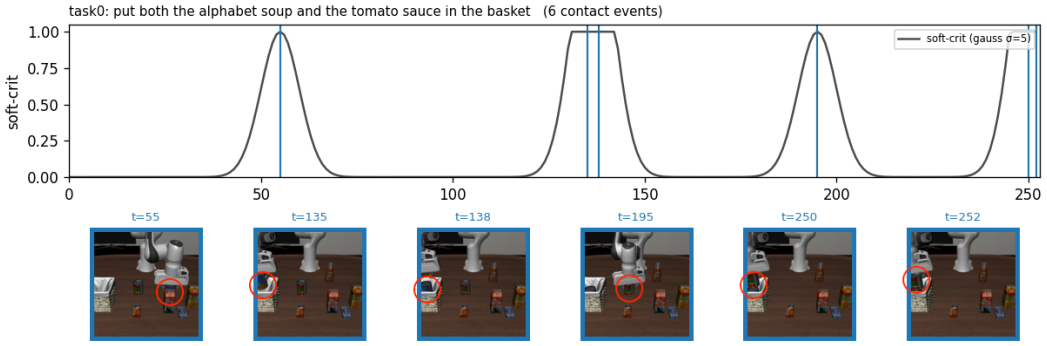


Figure 7: **Critical Index and Critical Moments in LIBERO.**

We evaluate the trigger in LIBERO and RoboTwin 2.0 with two complementary metrics. *Status AUC* is the AUC-ROC between the trigger value and a rule-based critical label derived only from simulator state: critical moments are when task-related objects switch between contact and non-contact. This avoids using the visual observations or policy actions that define the trigger and requires no manual annotation. *Variance Correlation* is Spearman's ρ between the trigger value and the variance of repeatedly sampled candidates. It measures whether the trigger activates when the policy is uncertain and supplies genuinely different alternatives.

We report Variance Correlation even though it is related to the trigger magnitude by construction. This metric is included as a diagnostic of the underlying design intuition, rather than as an independent claim of causal superiority: the trigger is motivated precisely by the expectation that challenging, contact-rich states exhibit both higher temporal complexity and greater cross-sample diversity. Reporting both metrics makes this relationship explicit and separates alignment with simulator-defined critical states from alignment with policy stochasticity.

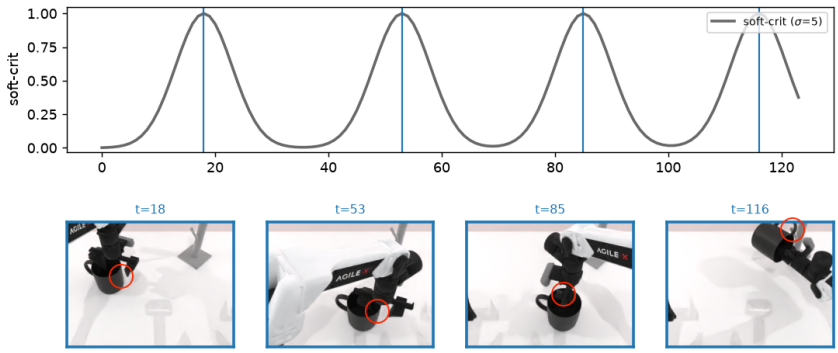


Figure 8: **Critical Index and Critical Moments in RoboTwin.**

Table 8: **Trigger performance analysis.** Status AUC and Variance Correlation use the definitions above. Each method calls k action chunk samples and therefore incurs the listed additional latency.

Method	Status AUC ↑	Variance Correlation ↑	k / add. latency (ms) ↓
random	0.487	–	–
DINO	0.702	0.630	**0 / 0**
ACE	0.588	0.716	2 / 14
T_{temp}	0.704	0.789	**1 / 0**
T_{cross}	0.694	0.877	2 / 14
Ours	**0.727**	**0.887**	2 / 14

Table 9: **Completion score breakdown for the three real-world tasks.**

Task	Stage 1	Score	Stage 2	Score
Pen Insertion	Grasp the pen	0.5	Insert the pen into the yogurt bottle	0.5
Pipetting	Aspirate water from the right bottle	0.5	Dispense water into the left bottle	0.5
Apple Placement	Grasp the apple	0.5	Place the apple into the container	0.5

Baselines. We compare the Two-Axis Trigger with a DINOv2-feature MLP, action chunk entropy (ACE), and its temporal and cross-sample components.

Results. Our training-free trigger obtains the best Status AUC (0.727) and Variance Correlation (0.887). The former indicates stronger alignment with simulator-defined contact-critical moments, while the latter is consistent with the intended uncertainty-and-diversity intuition. We emphasize that the correlation result is diagnostic: because candidate variance is one of the signals used to construct the trigger, it should be interpreted as evidence that the proposed statistic tracks the intended policy behavior, not as an independent performance objective. The trigger uses two samples and adds 14 ms per action chunk.

D Real-world Setting

D.1 Tasks

We conduct experiments on three real-world tasks, pen insertion, pipetting, and apple placement. Table 9 defines the completion score of each task.

D.2 COMPUTATIONAL COST

We report the inference computational cost in Table 10, including latency and GPU usage.

Table 10: **Latency and GPU memory usage breakdown.**

Latency	
Component	Latency (ms)
VLA policy	82
Two-Axis Trigger (k=2)	14
Additional sampling ($K-k$=8 more candidates)	80
Verifier forward (all K=10 candidates)	246
Total latency (deactivated)	96
Total latency (activated)	422
Additional latency (per activated)	326
GPU Memory Usage	
Component	GPU Memory (GB)
VLA policy	9.8
Verifier forward (all K=10 candidates)	8.7
Total GPU memory	18.5

D.3 TRIGGER CALIBRATION

We identify the timesteps at which the task enters a contact-rich/critical phase, and record the trigger values τ_t at those moments for 5 episodes per task. The threshold p is then set as the percentile cutoff that best separates these identified critical timesteps from the rest, so that the trigger reliably fires once the task enters such phases. This is a one-time, per-task statistical calibration on the policy's own output distribution, and requires no annotation beyond marking when critical phases begin.