HRAE: ONE-STEP ACTION GENERATION FOR FLOW-MATCHING VLAS VIA HIERARCHICAL RECURRENT ACTION EXPERT

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion and flow-based vision-language-action (VLA) policies achieve strong robotic control through repeated model evaluations to progressively refine actions, but such multi-step inference introduce substantial latency. Existing one-step approaches reduce this cost by compressing the external generative process, often through distillation, consistency objectives, or specialized training targets, which can require pretrained teachers or additional trajectory-level training machinery. However, reducing external sampling steps also removes the iterative computation performed during generation. Recent evidence suggests that such iterative computation, when appropriately supervised, is an important inductive bias for generative robotic control. This raises a key question: can we retain iterative computation while avoiding repeated external flow evaluations? We address this by moving iterative computation inside each action-expert evaluation. We introduce the **Hierarchical Recurrent Action Expert (HRAE)**, which couples a fast low-level latent state for recurrent action refinement with a persistent high-level latent state that integrates information across refinements and guides subsequent computation. By reusing the same low-level and high-level modules across hierarchical recurrent cycles, HRAE achieves substantial effective computational depth without proportionally increasing model size. We further pair HRAE with clean-action x-prediction, which aligns the prediction target with its internally refined action representation while remaining compatible with multi-step flow integration. Across tasks in LIBERO, LIBERO Plus, RoboTwin, and real-world manipulation, HRAE achieves strong in-distribution performance and robustness under distribution shifts with only one external flow evaluations. In real-world tasks, one-step HRAE outperforms a 10-step GR00T baseline while reducing action-expert latency from 44.73 ms to 6.85 ms. These results suggest that iterative computation need not be tied to repeated external flow integration and can instead be effectively realized within the action expert for efficient generative robotic control.

1 INTRODUCTION

Recent vision-language-action (VLA) models commonly combine a pretrained vision-language model (VLM) with a specialized continuous action expert for robotic control (Black et al., 2024; Liu et al., 2025; Community, 2026). Diffusion and flow-based generative control policies (GCPs) have demonstrated strong performance across a wide range of robotic manipulation tasks by progressively transforming noisy actions into executable trajectories through iterative denoising or flow integration (Chi et al., 2025; Black et al., 2024). However, this iterative generation process typically requires repeated action-expert evaluations, introducing substantial inference latency.

Therefore, a growing body of work seeks to compress multi-step diffusion or flow generation into one-step prediction. Consistency Policy (Prasad et al., 2024) and One-Step Diffusion Policy (Wang et al., 2024) distill multi-step denoising trajectories, FreqPolicy (Su et al., 2026) imposes frequency-domain consistency across flow timesteps, and SnapFlow (Luan et al., 2026) learns shortcut targets from the policy's own multi-step predictions. Mean Flow Policy (Zhan et al., 2026) instead predicts a mean velocity for long-range transitions, while Chen et al. (2026) improve one-step v-prediction by emphasizing highly corrupted states. These approaches primarily reduce external sampling steps

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.

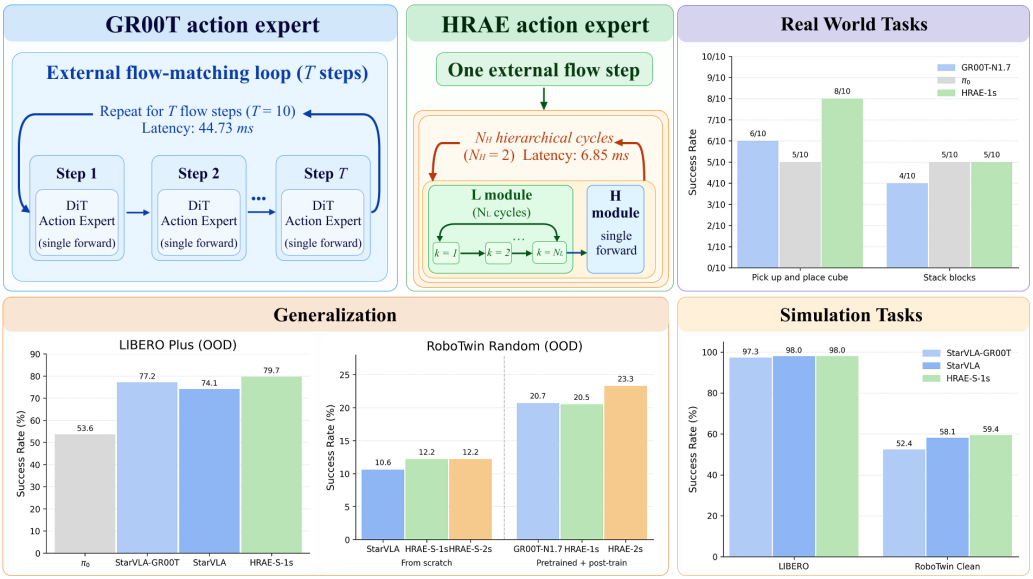

Figure 1: **Overview of HRAE and evaluation results.** GR00T performs iterative action refinement through repeated external flow evaluations, whereas HRAE internalizes iterative computation within the action expert through hierarchical recurrence, enabling one-step external flow inference. HRAE demonstrates strong performance across simulation, OOD generalization, and real-world manipulation, while reducing action-expert latency from 44.73 ms for 10-step GR00T to 6.85 ms.

by modifying trajectory-level supervision, prediction targets, or timestep sampling, with some additionally relying on teacher-dependent or auxiliary training machinery.

However, reducing external flow evaluations also removes the iterative computation performed during generation. Recent controlled experiments by Pan et al. (2026) suggest that the strength of generative control policies is not primarily explained by multimodal action modeling or a more expressive observation-to-action mapping. Instead, *supervised iterative computation*, together with an appropriate level of stochasticity, emerges as an important inductive bias for robotic control. This creates a fundamental tension: efficient one-step inference trys to remove external iteration, while iterative computation itself appears beneficial for control. Based on this, we ask whether iterative computation can be preserved *within* each action-expert evaluation, enabling high-quality action generation without repeated external flow integration.

To this end, we introduce a **Hierarchical Recurrent Action Expert (HRAE)** for flow-based VLA policies, inspired by hierarchical reasoning models Wang et al. (2025; 2026). HRAE not only rapidly updates the current action hypothesis, but also maintains a more persistent high-level latent state that integrates information across successive refinement steps and guides subsequent computation. As illustrated in Fig. 1, each HRAE evaluation consists of N_H hierarchical cycles, where each cycle performs N_L recurrent low-level updates followed by one high-level update. The low-level module progressively refines the action representation under multimodal conditioning and the current high-level state, while the high-level module integrates the refined representation and carries persistent guidance into the next cycle. By reusing the same low-level and high-level modules across recurrent cycles, HRAE achieves substantial *effective computational depth* without proportionally increasing parameter count. In this way, HRAE moves iterative action refinement from repeated external flow evaluations into the action expert itself, enabling high-quality action generation with only one external flow step.

Flow-based VLA policies such as GR00T commonly train DiT action experts with v-prediction. Li & He (2026) show that, under a linear flow path, clean-data x-prediction and v-prediction are algebraically interconvertible and can parameterize the same generative trajectory, despite defining different network prediction spaces. Our ablations in Sec. 4.4 show that this distinction is architecture-dependent: x-prediction and v-prediction perform similarly for the DiT action expert, whereas HRAE benefits substantially from x-prediction. We attribute this preference to HRAE's recurrent computation, which progressively refines the latent representation toward a coherent action

estimate. Therefore, directly predicting the clean action better aligns with this internal refinement than predicting a noise-dependent velocity field. Accordingly, we adopt clean-action x-prediction, enabling HRAE to produce a high-quality action estimate with a single external flow evaluation.

We evaluate HRAE across simulation, out-of-distribution generalization, and real-world manipulation. HRAE achieves comparable or better performance on LIBERO and RoboTwin Clean, while improving robustness on LIBERO Plus and RoboTwin Random with only one external flow evaluations. In real-world manipulation, one-step HRAE outperforms the 10-step GR00T baseline while reducing action-expert latency from 44.73 ms to 6.85 ms. These results show that hierarchical recurrent computation within the action expert can effectively reduce external flow integration steps.

Our main contributions are summarized as follows:

- We introduce the Hierarchical Recurrent Action Expert (HRAE), which combines a fast-evolving low-level latent state for iterative action refinement with a persistent high-level latent state that integrates and coordinates information across refinement steps within each external flow evaluation.
- We formulate HRAE with clean-action x-prediction, aligning its recurrent latent refinement with direct action estimation and enabling high-quality generation with a single external flow evaluation. This provides an architectural approach to one-step generation that complements existing methods based primarily on trajectory-level supervision or flow-time integration.
- Extensive experiments on LIBERO, LIBERO Plus, RoboTwin, and real-world manipulation demonstrate strong in-distribution performance and improved robustness under distribution shifts. Notably, one-step HRAE outperforms the 10-step GR00T baseline on real-world tasks while reducing action-generation latency from 44.73 ms to 6.85 ms.

2 Related Work

Flow-matching action generation and prediction space. Diffusion policies model continuous action chunks through iterative denoising (Chi et al., 2025; Liu et al., 2025). Flow matching and rectified flow (Lipman et al., 2022; Liu et al., 2022) extend this generative approach to velocity-field integration in VLAs: π_0 (Black et al., 2024) couples a pretrained VLM with an action expert through joint attention, while GR00T (Bjorck et al., 2025) conditions a diffusion Transformer (Peebles & Xie, 2023) on VLM features through cross-attention. Both use velocity prediction and repeated flow evaluations to generate actions.

Diffusion models admit different prediction spaces and loss weightings (Salimans & Ho, 2022; Karras et al., 2022), with recent work revisiting clean-data prediction (Li & He, 2026). Although x- and v-prediction are interconvertible under linear flows, their optimization depends on architecture: HRAE benefits from x-prediction, while our DiT baseline performs similarly with either.

One- and few-step generative control. One- and few-step policies reduce the latency of iterative action generation through distillation (Prasad et al., 2024; Wang et al., 2024; Gao et al., 2024), consistency flow training (Zhang et al., 2025; Yan et al., 2025), or frequency-domain consistency (Su et al., 2026). Mean Flow Policy (Zhan et al., 2026) learns mean velocities with an instantaneous-velocity constraint, while SnapFlow (Luan et al., 2026) and One-Step Flow Policy (Li et al., 2026) use shortcut supervision and self-consistency, respectively, for self-distillation. Alternatively, Chen et al. (2026) enables one-step generation under standard v-prediction by emphasizing highly corrupted actions during training. These approaches primarily accelerate generation by changing how longer-range transitions along the generative trajectory are learned. HRAE instead explores a complementary architectural direction by increasing the effective computational depth within each action-expert evaluation.

Iterative latent computation. Supervised iteration with suitable stochasticity helps explain generative control performance, as demonstrated by the regression-based Minimum Iterative Policy (Pan et al., 2026). Universal Transformers, looped Transformers, and recurrent-depth language models increase effective depth through parameter-shared latent computation (Dehghani et al., 2018; Giannou et al., 2023; Yang et al., 2024; Geiping et al., 2025). HRM couples fast and slow recur-

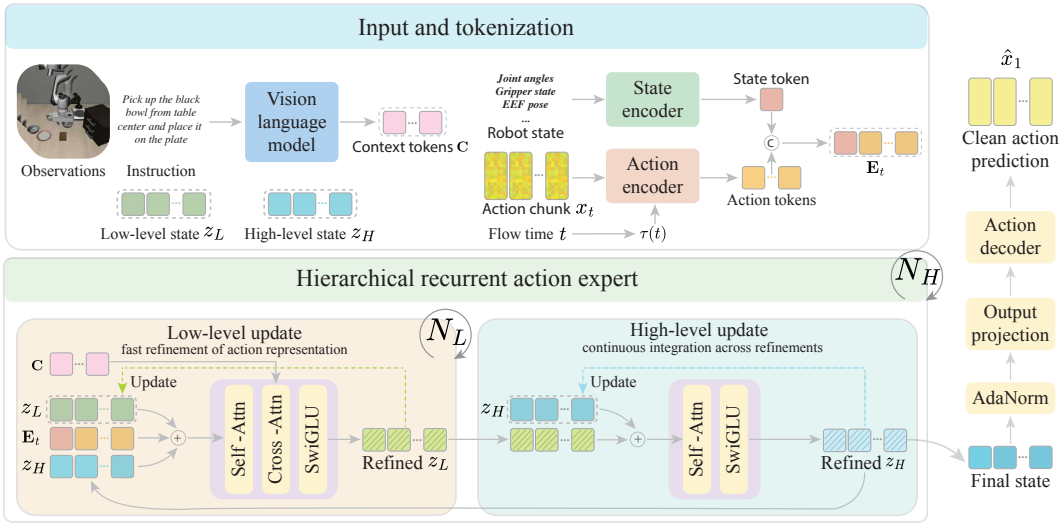

Figure 2: **Overview of our HRAE framework.** Given vision-language context, the current robot state, and a noisy action x_t, HRAE iteratively updates a low-level action representation z_L and a high-level latent z_H. At each low-level step, z_L is refined conditioned on z_H and the input context. After N_L times of low-level updates, z_H integrates the resulting information and conditions the subsequent update of z_L. The low-level and high-level modules share their parameters across N_H recurrent cycles, and the final state is decoded to predict the clean action chunk $\hat{x}_1$.

rent states (Wang et al., 2025), and HRM-Text extends this hierarchy to language-model pretraining (Wang et al., 2026). TRM explores recursive refinement with a single small network (Jolicoeur-Martineau, 2025). HRAE adapts hierarchical recurrence to multimodally conditioned continuous actions, supervising only the final prediction. Parameter sharing provides deep latent refinement within one external flow evaluation while keeping the action expert compact.

3 Method

Given multimodal observations $\mathbf{o}$, a language instruction ℓ, and robot proprioceptive state $\mathbf{s}$ as input, our model enables high-quality action generation with only one external flow evaluation by increasing the effective computation performed within each action-expert evaluation. We first outline the training and inference procedures for HRAE in Section 3.1. We then describe the design of its core module in Section 3.2. Finally, we present an approximate recurrent gradient strategy for efficient training in Section 3.3.

3.1 Flow Matching with Clean-Action Prediction

Our method follows the rectified flow framework (Lipman et al., 2022; Black et al., 2024). Given a continuous action chunk $\mathbf{a} \in \mathbb{R}^{T \times d_a}$, the forward process defines a linear interpolation between the action chunk and Gaussian noise:

$$\mathbf{x}_t = t\mathbf{a} + (1-t)\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad t \in [0,1]. \tag{1}$$

In contrast to standard rectified flow models which predict the velocity field, we optimize the network to predict the clean action chunk directly (Li & He, 2026):

$$\mathcal{L} = \mathbb{E}_{t,\mathbf{x},\boldsymbol{\epsilon}} \left\| \hat{\mathbf{x}}_1 - \mathbf{a} \right\|_2^2, \tag{2}$$

where $\hat{\mathbf{x}}_1 = f_\theta^{\mathrm{HRAE}}(\mathbf{x}_t, t, \mathbf{o}, \mathbf{s}, \ell)$. We find that this distinction interacts substantially with the action-expert architecture. As shown in Sec. 4.4, clean action and velocity prediction perform similarly for the DiT action expert, whereas HRAE benefits obviously from directly predicting the clean

action. We attribute this preference to HRAE's recurrent computation: its low-level and high-level states progressively refine an evolving action representation toward a coherent action estimate within each model evaluation. Direct action space prediction therefore aligns the network output with the result of this internal refinement, rather than requiring the refined representation to be converted into a noise-dependent instantaneous velocity.

Inference. For one-step inference, we evaluate HRAE once from the initial noisy action $\mathbf{x}_0 = \boldsymbol{\epsilon}$:

$$\mathbf{x}_1 = \hat{\mathbf{x}}_1. \tag{3}$$

For multi-step inference, the predicted clean action induces the flow update:

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \frac{\Delta t}{1-t}\left(\hat{\mathbf{x}}_1 - \mathbf{x}_t\right), \tag{4}$$

which allows the same x-predictor to be evaluated at intermediate flow states. We provide the corresponding derivation and the relation among different network prediction spaces in Appendix A.1.

3.2 Hierarchical Recurrent Action Expert

As shown in Figure 2, HRAE performs iterative action refinement within a single action-expert evaluation through two coupled latent states: a fast-evolving low-level state $\mathbf{z}_L$ and a slower, persistent high-level state $\mathbf{z}_H$. The low-level module repeatedly refines the evolving action representation under multimodal conditioning, while the high-level module integrates information accumulated across low-level refinements and coordinates subsequent refinement cycles.

Input representation. HRAE takes $\mathbf{o}, \ell, \mathbf{s}$ and noisy action chunk $\mathbf{x}_t$ as input given continuous flow time $t \in [0, 1]$. As shown in Figure 2, we map observation $\mathbf{o}$ and language instruction ℓ into context tokens $\mathbf{C} \in \mathbb{R}^{M\times d}$ using a pretrained vision language model. The proprioceptive state is encoded by a state MLP encoder $E_s(\mathbf{s})$ into state token $\mathbf{e}_s \in \mathbb{R}^d$.

For the k-th noisy action in the chunk, we first project the action and combine it with a sinusoidal embedding of flow time t:

$$\mathbf{e}_{a,k} = E_{at}\left([P_a(\mathbf{x}_{t,k}); \boldsymbol{\tau}(t)]\right) \in \mathbb{R}^d, \tag{5}$$

where P_a is a learnable action projection, $\boldsymbol{\tau}(t) \in \mathbb{R}^d$ is the sinusoidal flow-time embedding, and E_{at} is a two-layer MLP with a SiLU activation.

The action-expert input is the normalized sequence of one state token and T action tokens:

$$\mathbf{E}_t = \mathrm{RMSNorm}\left([\mathbf{e}_s; \mathbf{e}_{a,1}; \ldots; \mathbf{e}_{a,T}]\right) \in \mathbb{R}^{(T+1)\times d}. \tag{6}$$

One-dimensional rotary positional embeddings are applied along this state–action sequence.

Fast low-level refinement. Let $i \in \{0, \ldots, N_H - 1\}$ index high-level cycles and $j \in \{0, \ldots, N_L - 1\}$ index low-level updates. At each update, the current high-level state and action-expert input are injected into the evolving low-level state:

$$\bar{\mathbf{z}}_L^{i,j} = \mathbf{z}_L^{i,j} + \mathbf{z}_H^i + \mathbf{E}_t. \tag{7}$$

Let $\mathcal{B}_L^{(\ell)}$ denote the ℓ-th low-level Transformer block. Each block applies self-attention over the state–action sequence, cross-attention to the multimodal context, and a SwiGLU feed-forward transformation:

$$\mathbf{u} = \mathbf{q} + \mathrm{SA}\left(\mathrm{AdaRMS}(\mathbf{q}, t)\right), \tag{8}$$

$$\mathbf{v} = \mathbf{u} + \mathrm{CA}\left(\mathrm{AdaRMS}(\mathbf{u}, t), \mathbf{C}\right), \tag{9}$$

$$\mathcal{B}_L^{(\ell)}(\mathbf{q}; \mathbf{C}, t) = \mathbf{v} + \mathrm{SwiGLU}\left(\mathrm{RMSNorm}(\mathbf{v})\right). \tag{10}$$

Here, $\mathbf{q}$ denotes the input to the current block and $\mathrm{AdaRMS}(\cdot, t)$ is an RMS normalization modulated by the flow-time embedding. Cross-attention alternates between the visual and language subsets of context tokens $\mathbf{C}$ across low-level blocks (Bjorck et al., 2025).

Let $\mathcal{F}_L$ denote the sequential composition of the D_L low-level Transformer blocks. Each recurrent refinement is then

$$\mathbf{z}_L^{i,j+1} = \mathrm{RMSNorm}_L\left[\mathcal{F}_L\left(\bar{\mathbf{z}}_L^{i,j}; \mathbf{C}, t\right)\right]. \tag{11}$$

The same $\mathcal{F}_L$ is reused across all N_L recurrent updates, increasing computation without introducing additional Transformer parameters.

Persistent high-level integration. After N_L low-level refinements, the high-level module integrates the resulting representation:

$$\bar{\mathbf{z}}_H^i = \mathbf{z}_H^i + \mathbf{z}_L^{i,N_L}, \qquad \mathbf{z}_H^{i+1} = \mathcal{F}_H\left(\bar{\mathbf{z}}_H^i, t\right), \tag{12}$$

where $\mathcal{F}_H$ consists of D_H timestep-conditioned self-attention and SwiGLU blocks. Unlike the low-level module, $\mathcal{F}_H$ does not directly query VLM features. Instead, it integrates the multimodally grounded low-level representation and provides the high-level context for the next refinement cycle.

At model initialization, two latent vectors $\mathbf{h}_0, \mathbf{l}_0 \in \mathbb{R}^d$ are sampled once using truncated-normal initialization (Wang et al., 2025; 2026). Each action-expert evaluation starts from:

$$\mathbf{z}_H^0 = \mathbf{h}_0, \qquad \mathbf{z}_L^{0,0} = \mathbf{l}_0, \tag{13}$$

which remain fixed throughout training and are reused for every action-expert evaluation. The latent states then evolve through the hierarchical recurrent computation. Since both modules share parameters across recurrent cycles, HRAE decouples computational depth from parameter depth. With D_L low-level blocks and D_H high-level blocks, one action-expert evaluation performs $N_H\left(N_L D_L + D_H\right)$ block evaluations while parameterizing only $D_L + D_H$ distinct blocks.

Action decoding. After the final high-level cycle, we retain the action-token positions $\mathbf{z}_{H,\mathrm{act}}^{N_H}$ from $\mathbf{z}_H^{N_H}$ and apply a timestep-conditioned output normalization:

$$\mathrm{AdaNorm}(\mathbf{z}, t) = \mathrm{LayerNorm}(\mathbf{z}) \odot (1 + \gamma(t)) + \beta(t), \tag{14}$$

where $\gamma(t)$ and $\beta(t)$ are predicted from the flow-time embedding. The clean-action prediction is then decoded as

$$\hat{\mathbf{x}}_1 = D_a\left(P_o\left(\mathrm{AdaNorm}(\mathbf{z}_{H,\mathrm{act}}^{N_H}, t)\right)\right), \tag{15}$$

where P_o is the output projection and D_a is the action-decoder MLP. Only the final clean-action prediction is directly supervised, and all intermediate recurrent states remain latent.

3.3 Approximate Gradient for Recurrent Credit Assignment

Deep recurrence increases effective computational depth, but full backpropagation through the unrolled trajectory requires storing all recurrent activations and introduces long gradient paths that can destabilize optimization (Wang et al., 2025; 2026). We therefore preserve the complete HRAE forward recurrence while truncating only its backward credit assignment.

Specifically, we use separate backward horizons K_H and K_L for the high-level and low-level modules, measured over high-level cycles. Only updates within the last K_H and K_L cycles are recorded in the backward graph for $\mathcal{F}_H$ and $\mathcal{F}_L$, respectively, while all recurrent updates are always executed in the forward pass. Algorithm 1 summarizes the procedure.

Here, GradMode(true) records an update for backpropagation, whereas GradMode(false) executes the same forward computation without retaining its gradient graph. Smaller horizons reduce activation memory and shorten recurrent gradient paths while leaving the forward computation, and hence the effective computational depth of HRAE, unchanged.

4 Experiments

We evaluate HRAE along four dimensions: (1) in-distribution manipulation performance, (2) robustness under out-of-distribution environment shifts, (3) real-world manipulation, and (4) the ablation of prediction space and hierarchical recurrent computation. Unless otherwise specified, HRAE uses a single external flow-matching step at inference.

Algorithm 1 HRAE with approximate recurrent gradient

Require: $\mathbf{E}_t, \mathbf{C}, t$; N_H, N_L; backward horizons K_H, K_L

$\mathbf{z}_H \leftarrow \mathbf{h}_0, \quad \mathbf{z}_L \leftarrow \mathbf{l}_0$
for $i = 0$ to $N_H - 1$ **do**
  for $j = 0$ to $N_L - 1$ **do**
    $\mathbf{z}_L \leftarrow \textsc{GradMode}(i \geq N_H - K_L)\,\{\mathcal{F}_L(\mathbf{z}_L + \mathbf{z}_H + \mathbf{E}_t, \mathbf{C}, t)\}$
    $\mathbf{z}_L \leftarrow \mathrm{RMSNorm}_L(\mathbf{z}_L)$
  end for
  $\mathbf{z}_H \leftarrow \textsc{GradMode}(i \geq N_H - K_H)\,\{\mathcal{F}_H(\mathbf{z}_H + \mathbf{z}_L, t)\}$
end for
$\hat{\mathbf{x}}_1 \leftarrow \mathrm{Decode}(\mathbf{z}_{H,\mathrm{act}}, t)$
Optimize $\mathcal{L}_x$

Table 1: **Performance of LIBERO and LIBERO Plus.** Models are fine-tuned only on LIBERO and evaluated on LIBERO and LIBERO-Plus. **1s** denotes one external flow-matching steps.

Method	LIBERO	LIBERO Plus							
		Camera	Robot	Language	Lighting	Background	Noise	Layout	Total
π_0	94.4	13.4	6.0	58.8	85.0	81.4	79.0	68.9	53.6
StarVLA-GR00T	97.3	58.8	51.4	82.6	96.6	95.0	85.3	81.8	77.2
StarVLA	**98.0**	52.5	49.8	**88.5**	95.7	**95.7**	73.0	76.9	74.1
HRAE-Scratch-1s	**98.0**	**60.4**	**53.7**	86.0	**97.2**	95.2	**91.5**	**83.7**	**79.7**

4.1 Experimental Setup

Robot pretraining. We initialize the vision-language backbone from Qwen3-VL-4B-Instruct (Bai et al., 2025) and pretrain the complete VLA policy on a heterogeneous mixture of InternData-A1 (Tian et al., 2026) and ABC-130K (Allshire et al., 2026). Specifically, we use 227k episodes from InternData-A1 and 50k episodes from ABC-130K, resulting in 278k robot episodes in total. The VLM is fully trainable during robot pretraining, while the HRAE action expert is randomly initialized and trained jointly with the VLM. We pretrain for 200K optimization steps with a global batch size of 1,536, a learning rate of 1×10^{-4}, and an action chunk size of 50.

HRAE configuration. Our default HRAE contains $N_H = 2$ high-level cycles, with $N_L = 3$ low-level recurrent updates in each cycle. Both the low- and high-level modules contain four Transformer blocks, i.e., $D_L = D_H = 4$. Unless otherwise specified, HRAE performs one external flow evaluation at inference. On the RoboTwin benchmark, we additionally evaluate two-step inference. We denote the one-step and two-step variants by **1s** and **2s**, respectively.

Benchmarks and baselines. We evaluate on LIBERO (Liu et al., 2023), LIBERO Plus (Fei et al., 2025), RoboTwin (Mu et al., 2025), and real-world manipulation tasks. For LIBERO and LIBERO Plus, we train policies on the in-distribution LIBERO demonstrations and evaluate them on both the original LIBERO benchmark and its out-of-distribution (OOD) counterpart, LIBERO Plus, following the official evaluation protocols. Similarly, for RoboTwin, we train policies on the Clean dataset and evaluate them under both the Clean and Random settings to assess in-distribution performance and generalization to randomized environments.

We compare HRAE with π_0 (Black et al., 2024), GR00T-N1.7 (Bjorck et al., 2025), GalaxeaVLA (Liu et al., 2026), Xiaomi Robotics-0 (Cai et al., 2026), and StarVLA (Community, 2026). Their reported benchmark results are taken from Qwen-RoboMani (Yuan et al., 2026) and Robotwin Leaderboard in Community et al. (2026). We train **StarVLA-GR00T** (Community, 2026) from scratch to evaluate this architecture under OOD not reported in the original work.

4.2 Simulation Performance and OOD Generalization

LIBERO and LIBERO Plus. We first evaluate HRAE on LIBERO and its OOD counterpart LIBERO Plus. Models are trained on LIBERO and evaluated on the same tasks under the official LIBERO Plus perturbations, including camera, robot, language, lighting, background, noise, and

Table 2: **Out-of-distribution evaluation on RoboTwin Clean2Random.** Models are trained on RoboTwin Clean and evaluated under both the Clean and Random settings.

Method	RoboTwin Clean	RoboTwin Random	Method	RoboTwin Clean	RoboTwin Random
From-scratch training					
StarVLA-GR00T	52.4	4.8	HRAE-Scratch-1s	59.4	12.2
StarVLA	58.1	10.6	HRAE-Scratch-2s	**60.4**	**12.2**
Pretrained + RoboTwin post-training					
π_0	46.4	16.3	GalaxeaVLA	62.7	12.7
GR00T-N1.7	43.6	20.7	Xiaomi Robotics-0	**62.9**	18.2
HRAE-1s	59.6	20.5	HRAE-2s	62.5	**23.3**

Table 3: **Real-world manipulation evaluation.** We compare GR00T-N1.7, π_0, and HRAE on two real-world manipulation tasks. Success rates (%) and action-expert inference latency are reported.

Model	Flow Steps	Action Expert Latency (ms)	Pick up and place cube	Stack blocks
GR00T-N1.7	10	44.73	6/10	4/10
π_0	10	44.29	5/10	**5/10**
HRAE	**1**	**6.85**	**8/10**	**5/10**

layout shifts. As shown in Table 1, HRAE achieves 98.0% success on LIBERO using only one external flow step, matching the strongest baseline while outperforming π_0 and StarVLA-GR00T.

The advantage becomes more pronounced under distribution shift. HRAE-1s achieves an overall success rate of 79.7% on LIBERO Plus, compared with 74.1% for StarVLA and 53.6% for π_0. HRAE obtains particularly strong results under camera, robot, lighting, noise, and layout perturbations, while remaining competitive under language and background shifts. These results suggest that increasing iterative computation inside the action expert can improve robustness without relying on repeated external flow integration.

RoboTwin Clean-to-Random. As shown in Table 2, we further evaluate generalization under the stronger environmental randomization of RoboTwin. In the from-scratch setting, HRAE-1s already achieves 59.4% on Clean and 12.2% on Random with a single external flow evaluation, outperforming StarVLA at 58.1% and 10.6%, respectively. Using two flow evaluations further improves the Clean performance to 60.4%, while maintaining 12.2% on Random.

Robot pretraining substantially improves OOD robustness. HRAE-1s reaches 59.6% on Clean and 20.5% on Random, while HRAE-2s further improves to 62.5% and 23.3%, respectively. Compared with GR00T-N1.7 at 43.6% / 20.7%, HRAE-1s achieves substantially higher in-distribution performance with comparable OOD performance, while HRAE-2s improves both.

4.3 Real-World Evaluation and Efficiency

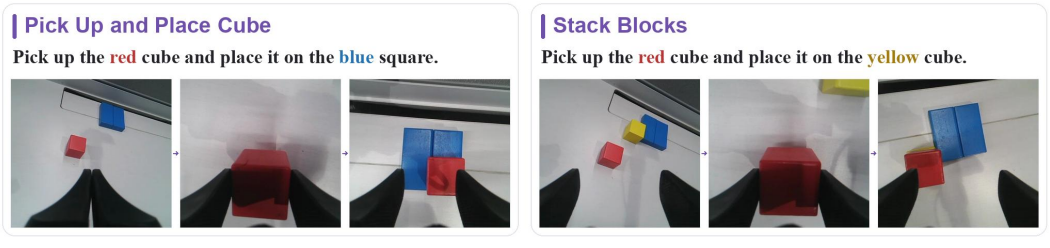

Figure 3: Real-world manipulation Tasks

Real-world manipulation. We evaluate HRAE on a real PIPER single-arm robot equipped with a left-wrist camera. As shown in Figure 3, we consider two manipulation tasks, *Pick up and place*

Table 4: **Ablation on prediction space and hierarchical recurrence.** We compare x-prediction and v-prediction for DiT and HRAE action experts, and ablate high-level recurrence in HRAE. All models are trained from scratch.

GR00T (DiT)		HRAE		
Prediction	**RoboTwin**	**Prediction**	**H-level Recurrence**	**RoboTwin**
v-pred	52.4	v-pred	✓	52.2
x-pred	52.5	x-pred	×	56.7
		x-pred	✓	**59.4**

cube and *Stack blocks*, with 50 training demonstrations per task. Each policy is post-trained for 20K steps using a global batch size of 256 across 8 GPUs and evaluated over 10 trials using task success rate. Detailed task information is provided in Appendix A.2.

As shown in Table 3, one-step HRAE achieves 80% success on *Pick up and place cube*, compared with 60% for GR00T-N1.7 and 50% for π_0. On *Stack blocks*, HRAE reaches 50%, matching π_0 and outperforming GR00T-N1.7 at 40%. Despite using only a single external flow evaluation, HRAE therefore achieves stronger or comparable real-world performance than the 10-step baselines. Further analysis of real-world failure cases is provided in Appendix A.3. Representative videos of failed task executions are provided in the supplementary material.

Inference efficiency. We measure action-expert latency with batch size 1 on an NVIDIA RTX 5880 Ada Generation GPU. For each task and model, we perform 5 warm-up runs followed by 50 timed inference runs. We first compute the mean latency over the 50 runs for each task and then average the resulting latency across the two tasks. As shown in Table 3, one-step HRAE requires only 6.85 ms, compared with 44.73 ms for 10-step GR00T-N1.7 and 44.29 ms for 10-step π_0.

4.4 Ablation Studies

Prediction space and action-expert architecture. As shown in Table 4, the effect of prediction space depends strongly on the action-expert architecture. For the DiT action expert, x-prediction and v-prediction perform nearly identically (52.5% vs. 52.4%), consistent with their algebraic correspondence under the same linear flow path. In contrast, HRAE improves substantially from 52.2% with v-prediction to 59.4% with x-prediction. This supports our hypothesis that clean-action prediction is better aligned with HRAE's recurrent latent refinement, which progressively resolves the representation toward a coherent action estimate.

Contribution of hierarchical recurrence. We further remove the high-level recurrence while retaining low-level iterative refinement. Under x-prediction, this reduces RoboTwin success from 59.4% to 56.7%, demonstrating that the gain does not arise solely from repeated low-level computation. The persistent high-level state provides complementary integration across low-level refinements and improves the final action prediction.

5 Conclusion

We introduce the **Hierarchical Recurrent Action Expert (HRAE)** for flow-based VLA policies, which moves iterative action refinement from repeated external flow evaluations into the action expert through hierarchical recurrent computation. By combining fast low-level refinement with persistent high-level integration and sharing parameters across recurrent cycles, HRAE achieves substantial effective computational depth within each model evaluation. We further find that clean-action x-prediction is particularly well aligned with the internally refined action representation. Experiments on LIBERO, LIBERO Plus, RoboTwin, and real-world manipulation show that HRAE matches or improves upon DiT-based action experts with only one external flow evaluation, while providing stronger OOD robustness. These results suggest that increasing the effective computational depth within the action expert is a promising alternative to increasing the depth of external flow integration for generative robotic control.

AI USE STATEMENT

We used generative AI tools only for language polishing and grammar checking. All AI-assisted revisions were carefully reviewed by the authors to ensure their accuracy and consistency with the intended meaning.

ETHICS STATEMENT

This work does not involve human subjects, sensitive personal data, or other issues covered by the ICLR Code of Ethics. Therefore, we do not anticipate any ethical concerns arising from this research.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the main experimental settings and configurations required to reproduce our results in the main paper. We provide additional details regarding our method and experiments in the appendix.

REFERENCES

Arthur Allshire, Himanshu Gaurav Singh, Ritvik Singh, Adam Rashid, Hongsuk Choi, David McAllister, Justin Yu, Yiyuan Chen, Huang Huang, Pieter Abbeel, et al. Scalable behavior cloning with open data, training, and evaluation. *arXiv preprint arXiv:2606.27375*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

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, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

Yitong Chen, Shiduo Zhang, Jingjing Gong, and Xipeng Qiu. Let it be simple: One-step action generation for vision-language-action models. *arXiv preprint arXiv:2606.05737*, 2026.

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

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

XPolicyLab Community, Tianxing Chen, Yue Chen, Tian Nian, Zijian Cai, Guangyu Chen, Wenwei Lin, Qiwei Liang, Peicheng Xiang, Kailun Su, et al. XPolicyLab: A unified standard and open ecosystem for robot policy evaluation and deployment. *arXiv preprint arXiv:2608.09892*, 2026.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

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.

Zifeng Gao, Guanxing Lu, Tianxing Chen, Wenxun Dai, Ziwei Wang, Chao Shang, Wenbo Ding, and Yansong Tang. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

Jonas Geiping, Sean Michael 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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=S3GhJooWIC.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 11398–11442, 2023.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

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

Shaolong Li, Lichao Sun, and Yongchao Chen. One-step flow policy: Self-distillation for fast visuomotor policies. *arXiv preprint arXiv:2603.12480*, 2026.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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.

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

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, et al. G0. 5: One autoregressive stream for robot reasoning and action. *arXiv preprint arXiv:2608.11739*, 2026.

Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. Snapflow: One-step action generation for flow-matching vlas via progressive self-distillation. *arXiv preprint arXiv:2604.05656*, 2026.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, et al. Robotwin: Dual-arm robot benchmark with generative digital twins. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27649–27660. IEEE, 2025.

Chaoyi Pan, Giridharan Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, et al. Much ado about noising: Dispelling the myths of generative robotic control. In *International Conference on Learning Representations*, volume 2026, pp. 90575–90614, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Yifei Su, Ning Liu, Dong Chen, Zhen Zhao, Kun Wu, Meng Li, Zhiyuan Xu, Zhengping Che, and Jian Tang. Freqpolicy: Efficient flow-based visuomotor policy via frequency consistency. *Advances in Neural Information Processing Systems*, 38:27769–27797, 2026.

Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 976–985, 2026.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

Guan Wang, Changling Liu, Chenyu Wang, Cai Zhou, Yuhao Sun, Yifei Wu, Shuai Zhen, Luca Scimeca, and Yasin Abbasi Yadkori. Hrm-text: Efficient pretraining beyond scaling. *arXiv preprint arXiv:2605.20613*, 2026.

Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, et al. Maniflow: A general robot manipulation policy via consistency flow training. *arXiv preprint arXiv:2509.01819*, 2025.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms, 2024. URL https://arxiv.org/abs/2311.12424.

Haoqi Yuan, Zhixuan Liang, Anzhe Chen, Ye Wang, Haoyang Li, Pei Lin, Yiyang Huang, Zixing Lei, Tong Zhang, Jiazhao Zhang, et al. Qwen-robotmanip technical report: Alignment unlocks scale for robotic manipulation foundation models. *arXiv preprint arXiv:2606.17846*, 2026.

Guojian Zhan, Letian Tao, Pengcheng Wang, Yixiao Wang, Yuxin Chen, Yiheng Li, Hongyang Li, Masayoshi Tomizuka, and Shengbo Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. In *International Conference on Learning Representations*, volume 2026, pp. 157132–157154, 2026.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14754–14762, 2025.

A Appendix

A.1 Prediction-Space Equivalence under the Linear Flow

Following the prediction-space terminology of Li & He (2026), we distinguish the quantity directly produced by the network from the space in which the loss is evaluated. A network whose direct output is the clean data, Gaussian noise, or velocity is referred to as x-, ϵ-, or v-prediction, respectively.

For the linear probability path used in our model,

$$\mathbf{x}_t = t\mathbf{x} + (1-t)\boldsymbol{\epsilon}, \tag{16}$$

the ground-truth velocity is

$$\mathbf{v} = \frac{\partial \mathbf{x}_t}{\partial t} = \mathbf{x} - \boldsymbol{\epsilon}. \tag{17}$$

These two equations uniquely determine the relationship among $\mathbf{x}$, $\boldsymbol{\epsilon}$, and $\mathbf{v}$.

x-prediction. When the network directly predicts the clean action $\hat{\mathbf{x}}$, the corresponding noise estimate is

$$\hat{\boldsymbol{\epsilon}} = \frac{\mathbf{x}_t - t\hat{\mathbf{x}}}{1-t}, \tag{18}$$

and the induced velocity is

$$\hat{\mathbf{v}}_t = \hat{\mathbf{x}} - \hat{\boldsymbol{\epsilon}} = \frac{\hat{\mathbf{x}} - \mathbf{x}_t}{1-t}. \tag{19}$$

ϵ-prediction. If the network instead directly predicts $\hat{\boldsymbol{\epsilon}}$, the clean-action estimate is

$$\hat{\mathbf{x}} = \frac{\mathbf{x}_t - (1-t)\hat{\boldsymbol{\epsilon}}}{t}, \tag{20}$$

with corresponding velocity

$$\hat{\mathbf{v}}_t = \frac{\mathbf{x}_t - \hat{\boldsymbol{\epsilon}}}{t}. \tag{21}$$

v-prediction. For direct velocity prediction $\hat{\mathbf{v}}_t$, the equivalent clean-action and noise estimates are

$$\hat{\mathbf{x}} = \mathbf{x}_t + (1-t)\hat{\mathbf{v}}_t, \tag{22}$$

$$\hat{\boldsymbol{\epsilon}} = \mathbf{x}_t - t\hat{\mathbf{v}}_t. \tag{23}$$

Therefore, under the linear path, x-, ϵ-, and v-prediction can be algebraically transformed into one another and parameterize the same underlying flow. They nevertheless correspond to different direct network outputs and can exhibit substantially different optimization behavior Li & He (2026).

One- and multi-step sampling. For our x-predictor, Eq. equation 19 converts the clean-action estimate into the corresponding vector field. An Euler step of size Δt gives

$$\mathbf{x}_{t+\Delta t} = \mathbf{x}_t + \frac{\Delta t}{1-t}\left(\hat{\mathbf{x}} - \mathbf{x}_t\right). \tag{24}$$

Starting from $\mathbf{x}_0 = \boldsymbol{\epsilon}$, a single step with $\Delta t = 1$ directly yields

$$\mathbf{x}_1 = \hat{\mathbf{x}}, \tag{25}$$

while smaller increments recover multi-step flow integration.

A.2 Real-World Task Instructions

We evaluate the real-world policies using the following language instructions. Each task contains 50 training demonstrations and is evaluated over 10 trials.

Table 5: **Language instructions for real-world manipulation tasks.**

Task	Language Instruction
Pick up and place cube	Pick up the red cube and place it on the blue square.
Stack blocks	Pick up the red cube and place it on the yellow cube.

A.3 Real-World Failure Analysis

We analyze the failure cases observed during real-world evaluation. Most failures arise from two sources. First, policy execution can become unstable under changes in object position, leading to inaccurate reaching, grasping, or placement. This suggests that local spatial perturbations remain challenging for all evaluated policies. Second, some episodes fail at the final stage of execution. For example, the robot may successfully move the object to the target region but fail to release it, resulting in an unsuccessful trial despite largely completing the task.

These failure modes occur more frequently for the baseline policies than for HRAE. The baselines are more sensitive to small positional variations and more often terminate with incomplete task execution, whereas HRAE exhibits more stable behavior across the evaluated configurations and more consistently completes the full manipulation sequence. This difference contributes to the higher real-world success rate of HRAE. We provide representative examples of failed task executions in the supplementary material.