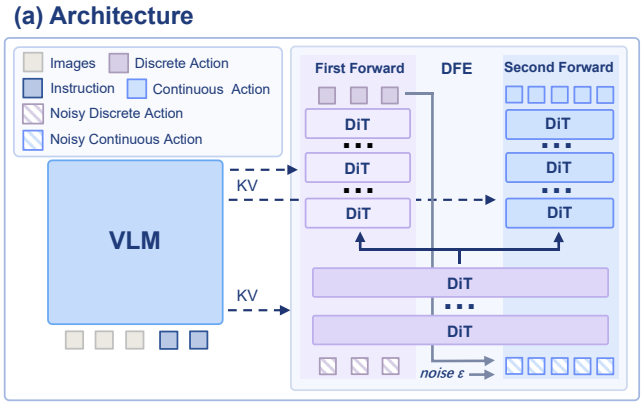
DISCRETE FORCING: INFUSING DISCRETE GUIDANCE INTO CONTINUOUS DENOISING FOR FEW-STEP ACTION EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient action generation in vision-language-action (VLA) models requires capturing both coarse action structure and fine-grained details. Discrete action tokens provide compact structural representations but sacrifice precision, while continuous action tokens offer high precision but often require multiple denoising steps. We introduce Discrete Forcing, a flow-matching framework that combines these representations through an explicit coarse-to-fine generation process. It first predicts discrete action tokens to establish a coarse action structure, then uses them to guide continuous action refinement. The discrete and continuous components share a common diffusion transformer backbone with specialized branches, maintaining a parameter count comparable to a conventional single-branch model while requiring only one forward pass per branch. Extensive evaluations across multiple benchmarks demonstrate improved performance and faster inference over a parameter-matched continuous action expert, with consistent performance gains as model capacity increases. Real-world experiments further demonstrate improvements on high-precision and dynamic manipulation tasks.



(b) Efficiency and performance

Metric	Backbone	DF (Ours)	Change
Efficiency			
NFE	10	**2**	1/5x
Latency (ms)	149.0	**29.0**	1/5x
Memory (GiB)	3.1	**3.1**	maintain
Performance			
RoboTwin	49.8	**59.3**	9.5%↑
GR1	43.9	**56.8**	12.9%↑
LIBERO	96.7	**97.6**	0.9%↑
Real-world	58.5	**74.9**	16.4%↑

Figure 1: **Overview of the Discrete Forcing (DF).** (a) Discrete Forcing Expert (DFE) performs discrete action prediction followed by continuous refinement within a partially shared Action DiT conditioned on VLM features. (b) This two-stage design reduces inference cost while consistently improving performance across simulation and real-world benchmarks.

1 INTRODUCTION

Across a wide range of domains, effective generation often proceeds from coarse structure to fine detail: architects establish the layout of a building before construction begins, just as filmmakers write the scenes before shooting (Baade et al., 2026). In latent diffusion (Ho et al., 2020), this coarse-to-fine progression emerges naturally from the denoising dynamics, shaped by the frequency distribution of natural images. Thus, pixel-space diffusion invariably means predicting low-frequency details before high-frequency details (Lee et al., 2025a). In contrast, actions lack a clearly defined

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.

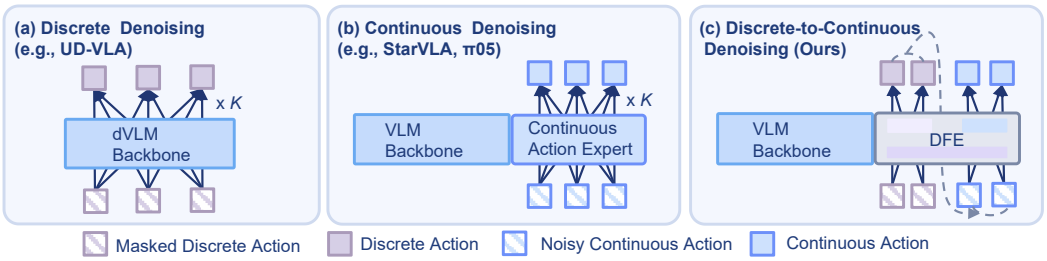


Figure 2: **Comparison of action generation paradigms.** Existing methods rely on K-step iterative denoising in either (a) discrete or (b) continuous action spaces, whereas (c) Discrete Forcing (Ours) first predicts a coarse discrete action and then performs a single continuous refinement.

frequency-domain structure. We pose the question: motivated by the coarse-to-fine generation principle observed in other domains, can action generation be modeled in a similar manner?

To address this question, we examine the action generation process from two perspectives in Figure 2: the representation space and the decoding order. **1)** In terms of representation, existing action generation methods operate in either discrete or continuous spaces. Discrete action tokens are typically produced by pretrained (Song et al., 2026; Liu et al., 2026; Dong et al., 2026) or rule-based tokenizers (Brohan et al., 2023); they provide compact representations that capture high-level action information, but often at the cost of reduced precision. Continuous actions, by contrast, offer higher precision and have become the predominant output representation in contemporary Vision-language-action models (VLAs) (Physical Intelligence et al., 2025; Bjorck et al., 2025; Wang et al., 2026; Team et al., 2026) and world-action models (WAMs) (Li et al., 2026; Bi et al., 2026a; Yang et al., 2026). However, directly denoising in the continuous action space is at odds with the coarse-to-fine generation principle and typically requires many denoising steps, resulting in slow inference. **2)** Beyond the representation space, the action generation order itself constitutes a broad design space: discrete-action models have explored autoregressive (Kim et al., 2024), block-wise (Song et al., 2026), and parallel (Song et al., 2025b) decoding, whereas continuous-action models generally employ either parallel (Chi et al., 2025; Wen et al., 2024; 2025) or coarse-to-fine generation (Gong et al., 2025).

Together, we seek to establish a sequential relationship across different action representations: first denoising discrete tokens that encode high-level information, and then conditioning the subsequent fine-grained denoising of continuous actions on these tokens. We propose Discrete Forcing, as shown in Figure 1 (a). In Discrete Forcing, we train a single flow-matching action expert over a discrete space and continuous space simultaneously with multiple time variables. The action expert first performs denoising in the discrete space to model the coarse structure of the action. Then, the generated discrete action effectively serves as a "scratchpad" and is used to initialize the continuous denoising process, thus removing the burden of modeling high-level information from the continuous branch. To unify the action generation process within a single architecture while explicitly distinguishing the modeling of discrete and continuous spaces, we employ several shared Diffusion Transformer (DiT) (Peebles & Xie, 2023) layers at the beginning of the action expert, followed by two separate branches for discrete and continuous modeling in its intermediate and later stages. The resulting architecture retains the same parameter count as a conventional single-branch DiT. Our experiments demonstrate that this design achieves the best performance. Benefiting from its explicit coarse-to-fine denoising process, Discrete Forcing enables highly efficient action modeling: high-quality actions can be generated with only one forward pass through each of the discrete and continuous branches, substantially improving inference efficiency.

We conduct extensive experiments across multiple benchmarks, demonstrating that our Discrete Forcing achieves superior performance and faster inference than a conventional continuous action expert with parallel parameters (Figure 1 (b)). We further validate the scalability of Discrete Forcing across models of different sizes, showing that its performance improves consistently with increasing model capacity. Real-world experiments further demonstrate that our Discrete Forcingyields performance gains on both high-precision and dynamic manipulation tasks. In summary, our contributions are as follows:

- We propose Discrete Forcing, a flow-matching framework that generates actions through discrete-to-continuous denoising, from coarse structure to precise details.
- We design a unified action expert with shared layers and specialized branches, enabling action generation with one forward pass per branch while matching the parameter count of a conventional DiT.
- Extensive benchmark and real-world experiments demonstrate superior performance, faster inference, and consistent improvements with increasing model capacity, including gains on high-precision and dynamic manipulation tasks.

2 Related Work

Action Representations in VLA Models. Action representations shape the learning objectives and decoding mechanisms of VLA policies. OpenVLA (Kim et al., 2024) predicts quantized action tokens autoregressively; Discrete Diffusion VLA (Liang et al., 2025) and DFM-VLA (Chen et al., 2026) instead reconstruct and refine token sequences through discrete denoising and probability velocity fields, respectively. UD-VLA (Chen et al., 2025a) extends discrete denoising to joint future-image and action generation. Continuous action modeling has developed through Diffusion Policy (Chi et al., 2025), RDT-1B (Liu et al., 2024), and diffusion-based VLAs such as TinyVLA and Diffusion-VLA (Wen et al., 2025; 2024), with π_0 and $\pi_{0.5}$ (Black et al., 2024; Physical Intelligence et al., 2025) adopting flow-matching action experts. Hybrid formulations combine both representations: HybridVLA (Liu et al., 2025) jointly trains autoregressive and diffusion-based action prediction and adaptively ensembles their outputs, while Knowledge Insulation (Driess et al., 2025) trains the backbone with discrete action supervision and blocks gradients from a continuous action expert to preserve knowledge transfer. These approaches establish complementary roles for discrete and continuous modeling, motivating an explicit study of how the two predictions should interact during action generation.

Coarse-to-Fine Action Generation. Coarse-to-fine policies organize action generation around intermediate predictions that guide subsequent refinement. CARP (Gong et al., 2025) learns multi-scale quantized action representations and generates them through coarse-to-fine next-scale prediction. Libra-VLA (Wei et al., 2026) introduces a discrete–continuous hierarchy in which coarse intent embeddings condition a separate diffusion refiner; an asynchronous intent buffer decouples the frequencies of planning and control. CF-VLA (Du et al., 2026) constructs an action-aware initialization by learning a continuous endpoint-velocity distribution, followed by a single local refinement step. Unlike prior hierarchical policies that use discrete representations as semantic intents or planning variables, Discrete Forcing constructs the coarse and fine stages as two representations of the same action trajectory, allowing the discrete prediction to directly define the source distribution of continuous flow matching.

Few-Step and Efficient Action Generation. Reducing action-generation cost has motivated parallel decoding, policy distillation, and training objectives tailored to limited inference budgets. PD-VLA (Song et al., 2025b) combines action chunking with parallel decoding, while OpenVLA-OFT (Kim et al., 2025b) integrates parallel prediction, continuous outputs, and L_1 regression. Consistency Policy (Prasad et al., 2024) enforces consistency along a pretrained diffusion policy's trajectories, and OneDP (Wang et al., 2025) distills a single-step generator through distribution matching to a diffusion teacher. For token-based VLAs, CEED-VLA (Song et al., 2025a) combines consistency distillation with early-exit decoding, and Fast-dVLA (Song et al., 2026) uses asymmetric distillation and pipelined block-parallel decoding. MIP (Pan et al., 2025) further shows that iterative two-step regression with intermediate supervision and controlled stochasticity can achieve strong control performance. Unlike these approaches that accelerate existing generation paradigms through parallelization or distillation, Discrete Forcing introduces a discrete-to-continuous generation process with one forward evaluation for each stage, directly optimized from action demonstrations.

3 METHOD

3.1 PRELIMINARY: DENOISING-BASED ACTION GENERATION

Diffusion-based policies formulate continuous action generation as an iterative denoising process. Given an action chunk $\mathbf{A}_t = [\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}]$, a noisy action $\mathbf{A}_t^\tau$ is constructed by interpolating the clean action with Gaussian noise:

$$\mathbf{A}_t^\tau = \tau\boldsymbol{\epsilon} + (1-\tau)\mathbf{A}_t, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{1}$$

where $\tau \in [0, 1]$ denotes the noise level. A conditional denoising network f_θ takes the noisy action together with the observation $\mathbf{o}_t$ and predicts the conditional vector field associated with this interpolation path. Following the flow-matching formulation, the model can be trained with

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{\mathbf{A}_t, \boldsymbol{\epsilon}, \tau}\left[\|f_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t) - (\boldsymbol{\epsilon} - \mathbf{A}_t)\|_2^2\right]. \tag{2}$$

At inference time, action generation starts from Gaussian noise and progressively integrates the predicted vector field from $\tau = 1$ to $\tau = 0$, yielding a continuous action chunk conditioned on the current observation.

3.2 ARCHITECTURE OF DISCRETE FORCING HEAD

As illustrated in Fig. 1 (a), Discrete Forcing integrates a VLM with a Discrete Forcing Expert built upon a partially shared Action DiT. The expert combines shared Transformer layers with specialized discrete and continuous branches, conditioned on layer-wise visual-language features from the VLM through cross-attention. This design supports coarse-to-fine action generation, in which discrete predictions capture the initial action structure and guide subsequent continuous refinement.

Partially shared dual-branch Action DiT. We construct a partially shared Action DiT with a discrete branch and a continuous branch. The first L_s Transformer layers share parameters across the two action representations, enabling the model to learn common action features. The subsequent layers and output heads are parameterized independently for discrete prediction and continuous refinement. This partially shared design allows each branch to generate its output with a single forward pass while maintaining the same parameter count as a conventional Action DiT, introducing no additional model-size overhead.

Discrete and continuous action representations. To construct the input representations, we first normalize the action chunk $\mathbf{A}_t$. For the discrete representation, each scalar entry is independently quantized into one of 255 action bins. Together with a special `[MASK]` token representing unknown action entries, these bins form a vocabulary of size $K = 256$. The resulting indices are mapped to discrete action tokens through a learnable embedding table. For the continuous representation, the normalized action chunk is projected into a sequence of continuous action tokens through an MLP-based action encoder. Both representations use the same hidden dimension before entering the Discrete Forcing Expert.

Within the Discrete Forcing Expert, interleaved self-attention and cross-attention layers model interactions among action tokens and incorporate VLM context, respectively. At each cross-attention layer, the corresponding layer-wise VLM representation provides visual and linguistic conditioning to both action representations throughout generation. The specialized branches do not perform cross-attention with each other. Finally, the discrete branch predicts categorical logits over the 255 action bins for each discrete action token, while the continuous branch predicts a vector field in the continuous action space. This partially shared design balances the learning of common action features with branch-specific specialization for discrete prediction and continuous refinement.

3.3 TRAINING

We train the discrete and continuous branches within a paired diffusion framework, while each action chunk $\mathbf{A}_t$ is assigned to only one branch for each training instance. The discrete branch learns to recover corrupted action tokens, whereas the continuous branch learns a vector field that maps a noisy source initialized from the discrete branch toward the continuous action trajectory.

We quantize the normalized action chunk $\mathbf{A}_t$ into a discrete sequence $\mathbf{Z}_t = Q(\mathbf{A}_t)$. A subset of discrete action tokens is replaced by the learnable `[MASK]` token according to the sampled forcing

level, producing a corrupted sequence $\mathbf{Z}_t^\rho$, where $\rho \in [0, 1]$ denotes the discrete corruption level. The discrete branch is trained to recover the original action bins only at the masked positions.

Let $\mathcal{M}$ denote the set of masked token indices. The discrete training objective is defined as

$$\mathcal{L}_{\text{disc}} = -\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \log p_{\theta_d} \left(z_i \mid \mathbf{Z}_t^\rho, \mathbf{o}_t \right), \tag{3}$$

where z_i denotes the ground-truth discrete action bin at position i, and p_{θ_d} is the categorical distribution predicted by the discrete branch.

To couple coarse discrete structure with precise continuous generation, we construct the source distribution of the continuous branch from the ground-truth quantized action and Gaussian noise. we define

$$\mathbf{S}_t = (1 - \alpha)\boldsymbol{\epsilon} + \alpha Q^{-1}(\mathbf{Z}_t), \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4}$$

where $Q^{-1}(\cdot)$ maps discrete action bins back to the normalized continuous action space. We set $\alpha = 0.3$ in experiments. We then interpolate between this discrete-guided source and the target action chunk:

$$\mathbf{A}_t^\gamma = \gamma \mathbf{S}_t + (1 - \gamma)\mathbf{A}_t, \qquad \gamma \in [0, 1], \tag{5}$$

where γ denotes the continuous-time variable that parameterizes the flow from the source state to the target action. The continuous branch predicts the corresponding vector field and is optimized with

$$\mathcal{L}_{\text{FM}} = \mathbb{E}\left[\| f_{\theta_c}(\mathbf{A}_t^\gamma, \mathbf{Z}_t, \mathbf{o}_t) - (\mathbf{S}_t - \mathbf{A}_t) \|_2^2 \right]. \tag{6}$$

To enable one-step continuous denoising at inference time, we further introduce a reconstruction objective that trains the continuous branch to directly recover the target action from the source state. Given $\mathbf{S}_t$, we obtain

$$\hat{\mathbf{A}}_t^{\text{1step}} = \mathbf{S}_t - f_{\theta_c}(\mathbf{S}_t, \mathbf{Z}_t, \mathbf{o}_t), \tag{7}$$

and minimize

$$\mathcal{L}_{\text{1step}} = \left\| \hat{\mathbf{A}}_t^{\text{1step}} - \mathbf{A}_t \right\|_2^2. \tag{8}$$

For samples assigned to the continuous branch, the training objective is

$$\mathcal{L}_{\text{cont}} = \mathcal{L}_{\text{FM}} + \lambda_{\text{1step}} \mathcal{L}_{\text{1step}}. \tag{9}$$

For samples assigned to the discrete branch, we optimize

$$\mathcal{L}_{\text{disc}}^{\text{train}} = \lambda_{\text{disc}} \mathcal{L}_{\text{disc}}. \tag{10}$$

We apply $\lambda_{\text{1step}} = 0.5$ and $\lambda_{\text{disc}} = 0.1$ in experiments.

3.4 Inference

At inference time, Discrete Forcing generates an action chunk with only two forward passes (*i.e.*, 2 function evaluations (NFE)): one discrete prediction followed by one continuous refinement. Starting from a fully masked discrete action sequence $\mathbf{Z}_{\text{mask}}$, the discrete branch predicts a coarse action prior in a single forward pass:

$$\hat{\mathbf{Z}}_t = \arg\max_{\mathbf{Z}} p_{\theta_d} \left(\mathbf{Z} \mid \mathbf{Z}_{\text{mask}}, \mathbf{o}_t \right). \tag{11}$$

The argmax is applied independently at each discrete action position. The predicted discrete sequence $\hat{\mathbf{Z}}_t$ is first dequantized to construct the discrete-guided continuous source:

$$\hat{\mathbf{S}}_t = (1 - \alpha)\boldsymbol{\epsilon} + \alpha Q^{-1}(\hat{\mathbf{Z}}_t), \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{12}$$

Starting from $\hat{\mathbf{S}}_t$, which corresponds to $\gamma = 1$, the continuous branch performs a single denoising step:

$$\hat{\mathbf{A}}_t = \hat{\mathbf{S}}_t - f_{\theta_c}\left(\hat{\mathbf{S}}_t, \hat{\mathbf{Z}}_t, \mathbf{o}_t \right). \tag{13}$$

The resulting continuous action chunk $\hat{\mathbf{A}}_t$ is used for robot execution.

Table 1: **Comparison on the LIBERO benchmark.** Prior methods are grouped according to their number of function evaluations (NFE): iterative generation with more than two NFE, and fast generation with NFE ≤ 2. Best results are highlighted in bold.

Method	Parameters	NFE ↓	Spatial	Object	Goal	Long	Avg.
Iterative Generation (NFE $>$ 2)							
DreamVLA (Zhang et al., 2026)	1.6B	10	97.5	94.0	89.5	89.5	92.6
MemoryVLA (Shi et al., 2026)	7B	10	98.4	98.4	96.4	93.4	96.5
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	3B	10	**98.8**	98.2	**98.0**	92.4	96.9
FlowerVLA (Reuss et al., 2025)	1B	8	97.5	99.1	96.1	94.9	96.9
Fast Generation (NFE $\leq$ 2)							
MolmoAct-7B-D (Lee et al., 2025b)	7B	1	87.0	95.4	87.6	77.2	86.6
MIP (Pan et al., 2025)	3B	2	97.6	95.8	95.2	82.2	92.7
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	3B	2	97.2	93.6	**98.0**	90.4	94.8
CF-VLA (Du et al., 2026)	3B	2	98.0	99.2	96.6	92.0	96.5
StarVLA (Community, 2026)	1.5B	2	96.2	98.0	95.4	90.8	95.1
Discrete Forcing (Ours)	1.5B	2	**98.8**	**99.4**	97.0	**95.2**	**97.6**

4 EXPERIMENTS

4.1 SIMULATION EXPERIMENTS

Experiment Settings. We evaluate Discrete Forcing on three popular simulation benchmarks: LIBERO (Liu et al., 2023), RoboTwin 2.0 (Chen et al., 2025b), and RoboCasa-GR1 (Nasiriany et al., 2024), covering single-arm, dual-arm, and dexterous manipulation settings. We adopt StarVLA as the baseline across all benchmarks and follow its official training and evaluation protocols for a fair comparison. Discrete Forcing and StarVLA use matched model capacities, with approximately 1.5B parameters on LIBERO and 5.5B parameters on RoboTwin 2.0 and RoboCasa-GR1. Detailed benchmark descriptions are provided in the Appendix B.

On LIBERO, we compare Discrete Forcing with prior methods under different inference budgets, grouping them into iterative generation (NFE $>$ 2) and fast generation (NFE $\leq$ 2), as shown in Table 1. Among fast-generation methods, Discrete Forcing achieves the best average success rate of 97.6%, outperforming StarVLA by 2.5 percentage points under the same 1.5B parameter scale and two-NFE inference budget. Moreover, its overall average exceeds all iterative-generation methods listed in the table, despite requiring only two function evaluations.

We further evaluate Discrete Forcing on RoboTwin 2.0 and RoboCasa-GR1 in Table 2. Detailed task-level results are provided in Appendices C and D. On RoboTwin 2.0, Discrete Forcing achieves the highest average success rate of 59.32%, improving over StarVLA from 49.80% to 59.32%. Notably, this performance is achieved without any robot-specific pre-training, demonstrating the strong effectiveness and generalization capability of our approach. Considering the training data of RoboTwin 2.0 is only 50 demonstrations per task, the results further showcase the few-shot learning capabilities. On RoboCasa-GR1, Discrete Forcing achieves an average success rate of 56.8%, outperforming StarVLA by 12.9% and achieving the best performance among the compared methods. These results demonstrate that Discrete Forcing maintains substantial performance gains in more complex robot embodiments and higher-dimensional action spaces.

4.2 ANALYSIS OF MODEL SCALING

An ideal action expert should exhibit favorable scaling behavior, with performance improving as its parameter count increases. To evaluate the scalability of Discrete Forcing, we train both the StarVLA baseline and our Discrete Forcing on three model scales under the same training budget on RoboTwin 2.0: 1.0B, 2.6B, and 5.5B parameters, with action experts comprising 175M, 400M, and 900M parameters, respectively. As shown in Fig. 3, the performance of Discrete Forcing improves monotonically with model size, with no evident saturation within the evaluated scale range. This demonstrates the scalability of our method, indicating that increased model capacity enables it to learn a broader range of tasks and acquire more diverse capabilities.

Moreover, Discrete Forcing consistently achieves higher success rates and lower inference latency than the corresponding baseline on the same model scale. In particular, the 1.0B variant of Discrete

Table 2: **Mean task success rate (%) on RoboTwin 2.0 and RoboCasa-GR1.**

(a) RoboTwin 2.0

Method	Avg.
DP (Chi et al., 2025)	28.04
DP3 (Ze et al., 2024)	55.24
RDT (Liu et al., 2024)	34.50
H-RDT (Bi et al., 2026b)	37.20
TwinVLA (Im et al., 2026)	42.00
π_0 (Black et al., 2024)	46.42
UP-VLA (Zhang et al., 2025)	52.92
StarVLA (Community, 2026)	49.80
Discrete Forcing (Ours)	**59.32**

(b) RoboCasa-GR1

Method	Avg.
$\pi_{0.5}$ (Physical Intelligence et al., 2025)	37.0
GR00T-N1.6 (NVIDIA et al., 2025)	47.6
DiT4DiT (Ma et al., 2026)	50.8
LangForce (Lian et al., 2026)	52.6
FLARE (Zheng et al., 2025)	55.0
DualCoT-VLA (Zhong et al., 2026)	55.1
LDA (Lyu et al., 2026)	55.4
StarVLA (Community, 2026)	43.9
Discrete Forcing (Ours)	**56.8**

Table 3: **Inference efficiency comparison on LIBERO.** Action latency measures the action-generation module only, while End-to-End Latency additionally includes the VLM forward pass. P50 and P95 denote the median and 95th-percentile latency, respectively.

Method	Success Rate (%) ↑	NFE ↓	Action Latency (ms) ↓		End-to-End Latency (ms) ↓	
			P50	P95	P50	P95
StarVLA (NFE=10)	96.7	10	145.46	149.36	200.43	206.06
StarVLA (NFE=4)	96.1	4	58.05	59.88	119.86	123.57
StarVLA (NFE=2)	95.1	2	29.20	30.44	93.29	96.26
Discrete Forcing (Ours)	**97.6**	2	**24.91**	**29.69**	**78.95**	**83.89**

Forcing achieves 47.96% success, approaching the 48.60% performance of the 5.5B baseline, while being 2.97× faster and using 81.8% fewer parameters.

4.3 Inference Efficiency

To quantitatively evaluate the inference efficiency of Discrete Forcing, we conduct a detailed latency analysis. As shown in Table 3, reducing the number of denoising steps of StarVLA from 10 to 2 improves inference speed, but also leads to an obvious drop in success rate, revealing a clear trade-off between efficiency and performance. In contrast, our Discrete Forcing achieves a substantially higher success rate than the 2-step baseline while retaining the speed benefits of two-step denoising, owing to the structural advantages of its coarse-to-fine denoising scheme. Despite adopting a dual-branch architecture, Discrete Forcing maintains the same parameter count and peak memory usage as the standard action expert, with all variants requiring 3.09 GiB of peak GPU memory under our evaluation setting. Compared with the 10-step baseline, our Discrete Forcing achieves 5× acceleration on action generation and more than 2× acceleration during the complete end-to-end process, with an even higher success rate.

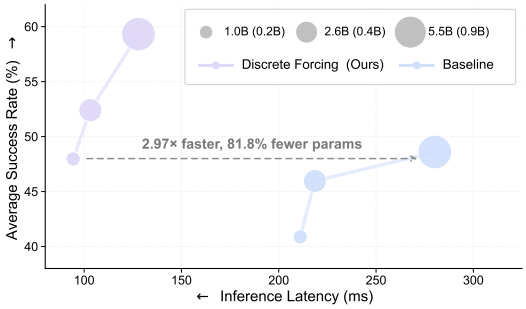


Figure 3: **Model scaling experiments.** Marker size denotes model parameter count, while the value in parentheses indicates the parameter count of the action expert. Our Discrete Forcing consistently improves with model scale while maintaining lower inference latency than the baseline.

4.4 Mechanism of Structural Action Generation

To better understand why Discrete Forcing outperforms the continuous-generation baseline, we investigate the internal action-generation process. We hypothesize that the first discrete prediction of Discrete Forcing facilitates the early formation of global action structure, compared with iterative continuous generation.

We begin with an examination of several action chunks. For visualization, we convert the translational actions into cumulative relative trajectories. Given translational increments $\Delta \mathbf{p}_h \in \mathbb{R}^3$, $h =$

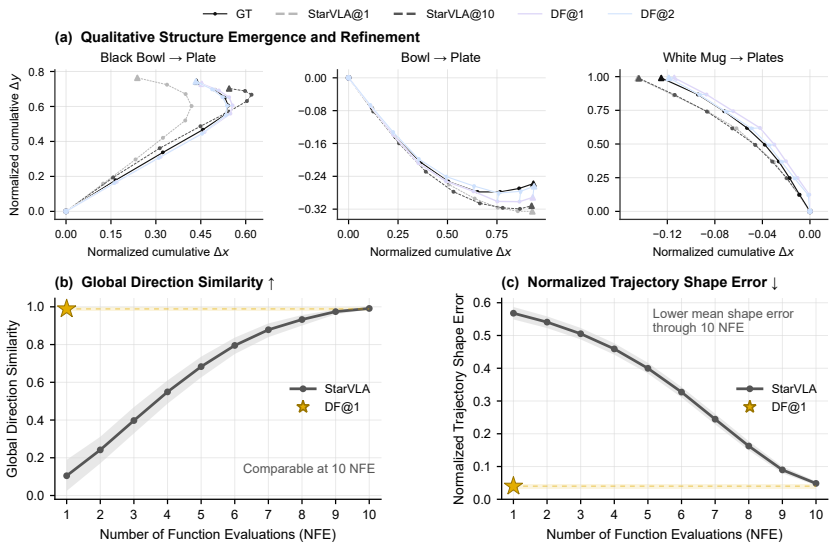


Figure 4: **Comparison of the action generation mechanisms between our method and the continuous-generation baseline.** DF denotes Discrete Forcing. DF@1 denotes the output after the first discrete forward pass, whereas DF@2 denotes the output after the subsequent continuous refinement. StarVLA@n denotes the action output after the n-th iterative forward pass.

$1, \ldots, H$, we define $\mathbf{p}_h = \sum_{k=1}^{h} \Delta\mathbf{p}_k$. For visualization, each trajectory is further normalized by its path length to reduce the effect of motion magnitude. As shown in Fig. 4(a), DF@1 already exhibits a trajectory geometry close to the overall motion pattern of the ground truth, whereas StarVLA@1 remains less structured. DF@2 further introduces local adjustments around the discrete trajectory.

To further quantify this observation, we introduce two complementary trajectory-level metrics. **Global Direction Similarity** evaluates whether the predicted action chunk moves in the correct overall direction, while **Normalized Trajectory Shape Error** evaluates how well its trajectory geometry matches the ground truth after removing the effect of motion magnitude. Higher direction similarity and lower shape error indicate better global action structure. Detailed metric definitions and analysis settings are provided in Appendix E.

As shown in Fig. 4(b), DF@1 achieves a Global Direction Similarity of 0.989 with only one NFE. StarVLA progressively improves its direction consistency with additional NFEs and reaches a comparable level only at 10 NFEs. This indicates that the discrete prediction captures the global motion direction at an early generation stage. Fig. 4(c) further shows that this early structural advantage extends beyond the motion direction. DF@1 achieves a Normalized Trajectory Shape Error of 0.0399, which remains lower than that of StarVLA throughout the evaluated 1–10 NFE range. This suggests that the advantage of the discrete prediction is not limited to the overall motion direction, but also extends to the global geometry of the action trajectory. These results support our hypothesis that the discrete stage facilitates the early formation of structured action representations, providing a useful structural prior for subsequent continuous refinement.

4.5 Ablation Studies

We ablate the key design choices of Discrete Forcing on LIBERO-Long, as summarized in Table 4. The generation-order ablation shows that Discrete $\rightarrow$ Continuous achieves the best performance, supporting our coarse-to-fine formulation in which discrete prediction establishes action structure before continuous refinement. In contrast, using the same representation for both stages consistently degrades performance. Introducing explicit branch communication further reduces success rates, favoring independently specialized branches. Removing the discrete-guided source or the one-step training objective also hurts performance. Finally, replacing the ground-truth discrete source during training with the predicted discrete source yields nearly identical performance (95.3% vs. 95.2%), indicating that the continuous refiner is robust to errors in discrete prediction. We therefore use the ground-truth discrete source during training, which avoids an additional discrete forward pass

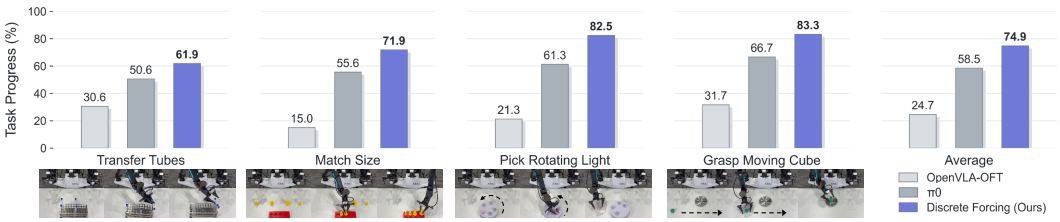


Figure 5: **Real-world experiments across four tasks.**

without sacrificing performance. Additional ablations on architectural and training configurations are provided in Appendix F.

4.6 Real-World Experiments

All real-world experiments are conducted using an ARX LIFT2 dual-arm robot equipped with three Intel RealSense D405 cameras: one head-mounted camera and one camera mounted on each arm. To evaluate our method under real-world manipulation challenges, we design four tasks that cover two complementary capabilities: precision manipulation and dynamic visuomotor response. **For precise tasks**, *Transfer Tubes* requires inserting 4 tubes into sequential slots, while *Match Size* requires matching 4 cylinders with slots of corresponding shapes and performing accurate insertion. **For dynamic tasks**, *Pick Rotating Light* evaluates the ability to track and grasp a continuously moving target. *Grasp Moving Cube* requires intercepting and grasping a laterally moving cube. To precisely measure task completion, we report task progress rather than binary success rate. We compare against OpenVLA-OFT (Kim et al., 2025a), a representative parallel continuous regression VLA, and π_0 (Black et al., 2024), a representative continuous flow-based policy. To demonstrate the versatility of our method and avoid potential limitations of evaluating real-world performance with an unpretrained StarVLA backbone, we instantiate our method on top of π_0 for real-world evaluation. We train a single model on trajectories collected from all real-world tasks, rather than training a separate model for each task. Detailed task settings and the definition of Task Progress are provided in Appendix H.

Table 4: **Ablation studies on LIBERO-Long.** We analyze the effect of generation order, architectural designs, and discrete conditioning strategies.

Variant	Success Rate (%)	Δ
Discrete Forcing (Ours)	**95.2**	–
Generation Order		
$Discrete \rightarrow Discrete$	93.6	−1.6
$Continuous \rightarrow Continuous$	91.8	−3.4
Architecture and Training Design		
w/o Discrete-guided Source	92.6	−2.6
w/o One-step Loss	93.4	−1.8
w/ Branch communication	90.4	−4.8
w/ Predicted discrete source (train) *(one extra forward)*	95.3	+0.1

As shown in Fig. 5, our method consistently outperforms both baselines across all four tasks, achieving an average Task Progress of 74.9%, corresponding to an improvement of 16.4%. The gains are particularly pronounced on Pick Rotating Light and Grasp Moving Cube, where successful execution depends critically on rapid visual grounding, dynamic target tracking, and timely action execution. These results indicate that our method improves not only precise manipulation but also responsiveness in latency-sensitive dynamic scenarios.

5 Conclusion

We presented Discrete Forcing, a coarse-to-fine action generation framework that unifies discrete and continuous representations within a single flow-matching action expert. By first generating discrete tokens that capture coarse action structure and then using them to initialize continuous denoising, our method combines compact high-level representations with precise action generation. Shared DiT layers and specialized branches enable this process while matching the parameter count of a conventional single-branch DiT, requiring only one forward pass per branch. Extensive evaluations demonstrate improved performance and faster inference over continuous action experts with comparable parameter counts, alongside consistent gains with increasing model capacity. Real-world experiments further validate its effectiveness on high-precision and dynamic manipulation tasks. These findings highlight discrete-to-continuous generation as a promising approach to efficient and precise robot action modeling.

AI USE STATEMENT

Generative AI tools were used solely to assist with manuscript organization, English language editing and polishing, and bibliography formatting. They were not used to generate experimental data, execute the reported evaluations, or generate references. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final manuscript.

ETHICS STATEMENT

This work studies robotic manipulation and does not involve human subjects, personally identifiable information, or sensitive personal data. The experiments are conducted in controlled simulation and laboratory environments. We are not aware of any specific ethical concerns arising from the experiments presented in this work.

REPRODUCIBILITY STATEMENT

The manuscript records the model formulation and the configuration values that can be verified from the current implementation. Appendix A collects these values, and Appendix C reports all available RoboTwin task-level results. Appendix D reports all available RoboCasa-GR1 task-level results. Appendix H reports the Real-world experimental Details.

REFERENCES

Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation. *arXiv preprint arXiv:2602.11401*, 2026.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026a.

Hongzhe Bi, Lingxuan Wu, Tianwei Lin, Hengkai Tan, Zhizhong Su, Hang Su, and Jun Zhu. H-rdt: Human manipulation enhanced bimanual robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18135–18143, 2026b.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. *arXiv preprint arXiv:2410.24164*, 2024. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Feilong Tang, Donglin Wang, and Haoang Li. Unified Diffusion VLA: Vision-Language-Action Model via Joint Discrete Denoising Diffusion Process. *arXiv preprint arXiv:2511.01718*, 2025a. URL https://arxiv.org/abs/2511.01718.

Jiayi Chen, Wenxuan Song, Jiaxin Fang, Ruiqing Yin, Jingbo Wang, Shuai Chen, Jieyuan Pei, Yikai Qin, Feifan Chen, Haodong Yan, Zhide Zhong, Wen Chen, Yan Wang, Yuxiang Gao, and Haoang Li. DFM-VLA: Iterative Action Refinement for Robot Manipulation via Discrete Flow Matching. *arXiv preprint arXiv:2603.26320*, 2026. URL https://arxiv.org/abs/2603.26320.

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

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.

Zibin Dong, Yicheng Liu, Shiduo Zhang, Baijun Ye, Yifu Yuan, Fei Ni, Jingjing Gong, Xipeng Qiu, Hang Zhao, Yinchuan Li, et al. Actioncodec: What makes for good action tokenizers. *arXiv preprint arXiv:2602.15397*, 2026.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge Insulating Vision-Language-Action Models: Train Fast, Run Fast, Generalize Better. *arXiv preprint arXiv:2505.23705*, 2025. URL https://arxiv.org/abs/2505.23705.

Fan Du, Feng Yan, Jianxiong Wu, Xinrun Xu, Weiye Zhang, Weinong Wang, Yu Guo, Bin Qian, Zhihai He, Fei Wang, and Heng Yang. CF-VLA: Efficient Coarse-to-Fine Action Generation for Vision-Language-Action Policies. *arXiv preprint arXiv:2604.24622*, 2026. URL https://arxiv.org/abs/2604.24622.

Zhefei Gong, Pengxiang Ding, Shangke Lyu, Siteng Huang, Mingyang Sun, Wei Zhao, Zhaoxin Fan, and Donglin Wang. Carp: Visuomotor policy learning via coarse-to-fine autoregressive prediction. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13460–13470. IEEE, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hokyun Im, Euijin Jeong, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. Twinvla: Data-efficient bimanual manipulation with twin single-arm vision-language-action models. In *International Conference on Learning Representations*, volume 2026, pp. 61969–61995, 2026.

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

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. *arXiv preprint arXiv:2502.19645*, 2025b. URL https://arxiv.org/abs/2502.19645.

Haeil Lee, Hansang Lee, Seoyeon Gye, and Junmo Kim. Beta sampling is all you need: Efficient image generation strategy for diffusion models using stepwise spectral analysis. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4215–4224. IEEE, 2025a.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025b.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

Shijie Lian, Bin Yu, Xiaopeng Lin, Laurence T Yang, Zhaolong Shen, Changti Wu, Yuzhuo Miao, Cong Huang, and Kai Chen. Langforce: Bayesian decomposition of vision language action models via latent action queries. *arXiv preprint arXiv:2601.15197*, 2026.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Tian Nian, Shunbo Zhou, Xiaokang Yang, Jiangmiao Pang, Yao Mu, and Ping Luo. Discrete Diffusion VLA: Bringing Discrete Diffusion to Action Decoding in Vision-Language-Action Policies. *arXiv preprint arXiv:2508.20072*, 2025. URL https://arxiv.org/abs/2508.20072.

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.

Chaoqi Liu, Xiaoshen Han, Jiawei Gao, Yue Zhao, Haonan Chen, and Yilun Du. Oat: Ordered action tokenization. *arXiv preprint arXiv:2602.04215*, 2026.

Jiaming Liu, Hao Chen, Pengju An, Zhuoyang Liu, Renrui Zhang, Chenyang Gu, Xiaoqi Li, Ziyu Guo, Sixiang Chen, Mengzhen Liu, Chengkai Hou, Mengdi Zhao, KC alex Zhou, Pheng-Ann Heng, and Shanghang Zhang. HybridVLA: Collaborative Diffusion and Autoregression in a Unified Vision-Language-Action Model. *arXiv preprint arXiv:2503.10631*, 2025. URL https://arxiv.org/abs/2503.10631.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: a Diffusion Foundation Model for Bimanual Manipulation. *arXiv preprint arXiv:2410.07864*, 2024. URL https://arxiv.org/abs/2410.07864.

Jiangran Lyu, Kai Liu, Xuheng Zhang, Haoran Liao, Yusen Feng, Wenxuan Zhu, Tingrui Shen, Jiayi Chen, Jiazhao Zhang, Yifei Dong, et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

NVIDIA, Johan Bjorck, Nikita Cherniadev Fernando Castañeda, 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. In *ArXiv Preprint*, March 2025.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much Ado About Noising: Dispelling the Myths of Generative Robotic Control. *arXiv preprint arXiv:2512.01809*, 2025. URL https://arxiv.org/abs/2512.01809.

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.

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. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency Policy: Accelerated visuomotor policies via consistency distillation. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.071. URL https://www.roboticsproceedings.org/rss20/p071.html.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-action flow policies. *arXiv preprint arXiv:2509.04996*, 2025.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 18567–18602, 2026.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Yuxin Huang, Han Zhao, Donglin Wang, and Haoang Li. CEED-VLA: Consistency Vision-Language-Action Model with Early-Exit Decoding. *arXiv preprint arXiv:2506.13725*, 2025a. URL https://arxiv.org/abs/2506.13725.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Zhijun Li, Donglin Wang, Lujia Wang, et al. Pd-vla: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13162–13169. IEEE, 2025b.

Wenxuan Song, Jiayi Chen, Shuai Chen, Jingbo Wang, Pengxiang Ding, Han Zhao, Yikai Qin, Xinhu Zheng, Donglin Wang, Yan Wang, et al. Fast-dvla: Accelerating discrete diffusion vla to real-time performance. *arXiv preprint arXiv:2603.25661*, 2026.

Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, et al. Xiaomi-robotics-1: Scaling vision-language-action models with over 100k hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proceedings of the AAAI conference on artificial intelligence*, volume 40, pp. 18638–18646, 2026.

Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-Step Diffusion Policy: Fast visuomotor policies via diffusion distillation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63399–63416. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25ba.html.

Yifei Wei, Linqing Zhong, Yi Liu, Yuxiang Lu, Xindong He, Maoqing Yao, and Guanghui Ren. Libra-VLA: Achieving learning equilibrium via asynchronous coarse-to-fine dual-system. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39799–39815. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1848. URL https://aclanthology.org/2026.acl-long.1848/.

Junjie Wen, Minjie Zhu, Yichen Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Chengmeng Li, Xiaoyu Liu, Yaxin Peng, Chaomin Shen, et al. Diffusion-vla: Generalizable and interpretable robot foundation model via self-generated reasoning. *arXiv preprint arXiv:2412.03293*, 2024.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, et al. 4d-wam: Infusing spatiotemporal awareness into world action models through trajectory fields. *arXiv preprint arXiv:2608.08023*, 2026.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*, 2025.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *Advances in Neural Information Processing Systems*, 38: 24195–24228, 2026.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, et al. Dualcot-vla: Visual-linguistic chain of thought via parallel reasoning for vision-language-action models. *arXiv preprint arXiv:2603.22280*, 2026.

A Implementation Details.

Table 7 summarizes the implementation details and training configurations used for our LIBERO experiments, including model architecture, optimization settings, objective weights, sampling strategies, and evaluation protocol.

B Detailed Simulation Experimental Settings

We evaluate our method on three simulation benchmarks: LIBERO (Liu et al., 2023), RoboTwin 2.0 (Chen et al., 2025b), and RoboCasa-GR1 (Nasiriany et al., 2024).

LIBERO consists of four suites, namely LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. It employs a Franka single-arm manipulator and evaluates generalization across different spatial layouts, objects, task goals, and long-horizon manipulation tasks.

RoboTwin 2.0 is a large-scale benchmark for bimanual robotic manipulation, comprising 50 diverse dual-arm tasks that cover a broad range of interaction patterns, including object placement, tool use, stacking, handover, articulated-object manipulation, and coordinated bimanual behaviors.

RoboCasa-GR1 consists of 24 tabletop manipulation tasks performed by a GR-1 humanoid robot. It complements the single-arm and dual-arm benchmarks above by evaluating generalist manipulation on a humanoid embodiment across diverse household scenarios.

C RoboTwin 2.0 Task-Level Results

While the main text reports the aggregate performance on RoboTwin 2.0, Table 5 provides the corresponding task-level success rates for a more fine-grained evaluation. We evaluate all 50 tasks following the benchmark protocol, with 50 evaluation episodes conducted for each task. The reported average is computed across all task-level success rates.

D RoboCasa-GR1 Task-Level Results

The main text reports the average success rates on RoboCasa-GR1 for comparison across different methods. For a more fine-grained evaluation, Table 6 provides the per-task success rates for methods whose task-level results are available. Following the standard evaluation protocol, we evaluate each of the 24 RoboCasa-GR1 tasks over 50 episodes. Although $\pi_0.5$ is included in the aggregate comparison in the main text, we omit it from this table because StarVLA (Community, 2026) reports only its overall performance on RoboCasa-GR1 without a per-task breakdown.

E Structural Action Analysis Details

Global Direction Similarity. We evaluate the global motion direction of the predicted action chunk. Specifically, we compute the cosine similarity between its net displacement and that of the ground-

truth trajectory:

$$S_{\text{dir}} = \frac{\mathbf{d}^{\text{pred}} \cdot \mathbf{d}^{\text{gt}}}{\|\mathbf{d}^{\text{pred}}\|_2 \|\mathbf{d}^{\text{gt}}\|_2 + \epsilon}, \qquad \mathbf{d} = \sum_{h=1}^{H} \Delta \mathbf{p}_h. \tag{14}$$

A higher value indicates better agreement in the global motion direction.

Normalized Trajectory Shape Error. To evaluate trajectory geometry independently of the overall motion magnitude, we normalize each relative trajectory by its path length:

$$L = \sum_{h=1}^{H} \|\Delta \mathbf{p}_h\|_2, \qquad \bar{\mathbf{p}}_h = \frac{\mathbf{p}_h}{L + \epsilon}, \tag{15}$$

and compute the average point-wise distance between the predicted and ground-truth normalized trajectories:

$$E_{\text{shape}} = \frac{1}{H} \sum_{h=1}^{H} \left\| \bar{\mathbf{p}}_h^{\text{pred}} - \bar{\mathbf{p}}_h^{\text{gt}} \right\|_2. \tag{16}$$

Lower values indicate better preservation of the global trajectory shape.

Evaluation Protocol. We conduct the analysis offline on LIBERO training demonstrations, as the goal is to probe the action-generation dynamics rather than evaluate policy generalization. We select five task identities spanning LIBERO-Object, LIBERO-Goal, LIBERO-Spatial, and LIBERO-10, and sample 40 action chunks of horizon $H = 8$ from each task using a fixed random seed, resulting in 200 action chunks in total.

The statistics in Fig. 4 (b,c) are computed by pooling all sampled action chunks. Since each task contributes the same number of samples, all five tasks receive equal weight. We report the mean and its 95% percentile bootstrap confidence interval, estimated from 2,000 bootstrap resamples.

F Extra Ablation Results

As shown in Table 8, introducing a small number of shared Action DiT layers substantially improves performance, with two shared layers achieving the highest success rate of 95.2%. These results suggest that limited early sharing facilitates common action-relevant representation learning, whereas excessive sharing restricts the specialization required by the discrete prediction and continuous refinement branches. We further ablate the composition of the discrete-guided continuous source. Table 9 shows that $\alpha = 0.3$ achieves the best performance among the tested settings.

G Real-World Experimental Details

H Real-world Tasks

Each task is evaluated over 20 independent trials. We report the completion rate of each milestone as the task progress rate. Representative execution trajectories of the four real-world tasks are shown in Fig. 7.

Transfer Tubes. Four test tubes are initially inserted into randomly selected slots on the right side of a test-tube rack. The robot transfers the tubes one by one to four designated target slots on the left side of the rack. The task contains eight milestones: grasping each test tube and inserting it into the corresponding target slot. The task emphasizes repeated fine-grained grasping and precise insertion over a relatively long manipulation sequence. The training dataset contains 200 episodes.

Match Size. Four cylinders of different sizes are randomly positioned in front of a base containing four correspondingly sized holes. The robot must identify the size correspondence between each cylinder and its target hole and insert all four cylinders into the correct locations. Each cylinder is grasped by the robot arm that is closer to it after random initialization. The task contains eight milestones: grasping and inserting each of the four cylinders. The task requires accurate size-based matching and precise bimanual insertion, as errors in target selection, alignment, or orientation can lead to failure. The training dataset contains 200 episodes.

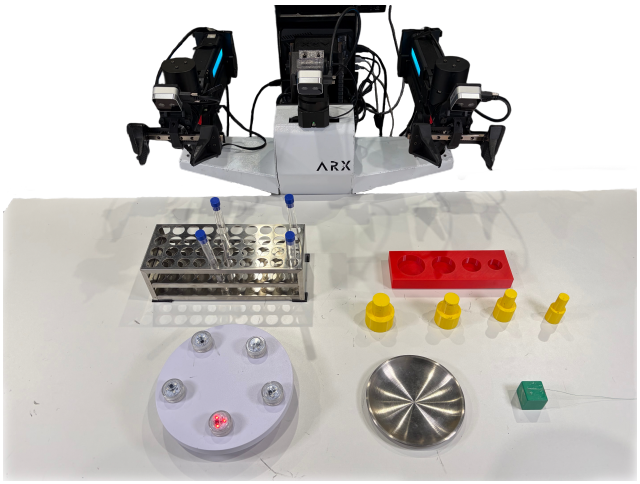


Figure 6: **Real-world experimental setup.**

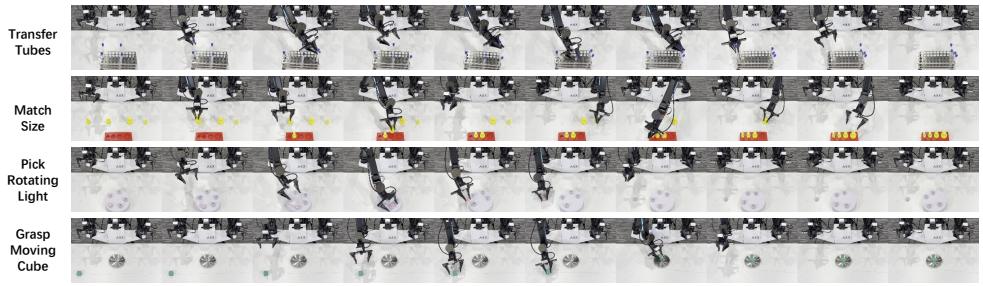


Figure 7: **Trajectories of real-world manipulation tasks.** From top to bottom, we show representative executions of Transfer Tubes, Match Size, Pick Rotating Light, and Grasp Moving Cube.

Pick Rotating Light. Five light modules are placed on a continuously rotating turntable, with only one illuminated in red. The robot is required to grasp a continuously moving red light and place it on the right side of the workspace. The task contains four milestones: locating and following the illuminated red light, lifting it from the turntable, moving it away from the rotating platform, and releasing it in the intended right-side area. Successful execution requires fast visual grounding and timely manipulation, as delays may cause the moving target to leave the intended grasp position. The training dataset contains 100 episodes.

Grasp Moving Cube. A block is initially placed on the right side of the workspace and pulled leftward by a rope. The robot is required to grasp the moving block, and place it onto a designated plate. The task contains three milestones: locating the moving cube, lifting and transporting it toward the plate, and successfully placing it on the plate. As the target remains in motion during the approach, successful execution requires rapid visuomotor response, precise interception, and timely grasping, followed by goal-directed placement. The training dataset contains 100 episodes.

Table 5: **Task-level success rates (%) on RoboTwin 2.0.** We report the success rate for each task, together with the average performance across all tasks.

Task	Success Rate	Task	Success Rate
adjust_bottle	100	place_can_basket	56
beat_block_hammer	78	place_cans_plasticbox	32
blocks_ranking_rgb	26	place_container_plate	98
blocks_ranking_size	20	place_dual_shoes	60
click_alarmclock	64	place_empty_cup	96
click_bell	60	place_fan	34
dump_bin_bigbin	80	place_mouse_pad	32
grab_roller	94	place_object_basket	76
handover_block	42	place_object_scale	68
handover_mic	90	place_object_stand	72
hanging_mug	24	place_phone_stand	68
lift_pot	50	place_shoe	80
move_can_pot	40	press_stapler	92
move_pillbottle_pad	62	put_bottles_dustbin	28
move_playingcard_away	92	put_object_cabinet	42
move_stapler_pad	18	rotate_qrcode	80
open_laptop	82	scan_object	56
open_microwave	14	shake_bottle_horizontally	96
pick_diverse_bottles	34	shake_bottle	100
pick_dual_bottles	62	stack_blocks_three	12
place_a2b_left	44	stack_blocks_two	54
place_a2b_right	46	stack_bowls_three	42
place_bread_basket	68	stack_bowls_two	80
place_bread_skillet	62	stamp_seal	24
place_burger_fries	90	turn_switch	46
Avg.	**59.32**		

Table 6: **Per-task success rate (%) on the RoboCasa-GR1 benchmark.**

Task	**DiT4DiT**	**GR00T-N1.6**	**StarVLA-π**	**LangForce**	**DualCoT-VLA**	**LDA**	**Ours**
BottleToCabinetClose	48.0	51.5	26.0	72.0	66.0	76.0	74.0
CanToDrawerClose	74.0	13.0	62.0	78.0	64.0	71.0	82.0
CupToDrawerClose	52.0	8.5	42.0	46.0	46.0	41.0	20.0
MilkToMicrowaveClose	50.0	14.0	50.0	56.0	58.0	52.0	62.0
PotatoToMicrowaveClose	36.0	41.5	42.0	36.0	30.0	41.0	40.0
WineToCabinetClose	42.0	16.5	32.0	46.0	38.0	57.0	56.0
FromCuttingboardToBasket	52.0	58.0	40.0	66.0	44.0	65.0	44.0
FromCuttingboardToCardboardbox	48.0	46.5	46.0	40.0	54.0	69.0	56.0
FromCuttingboardToPan	76.0	68.5	70.0	68.0	80.0	75.0	78.0
FromCuttingboardToPot	62.0	65.0	40.0	48.0	64.0	61.0	62.0
FromCuttingboardToTieredbasket	50.0	46.5	44.0	44.0	46.0	51.0	46.0
FromPlacematToBasket	50.0	58.5	44.0	54.0	48.0	53.0	54.0
FromPlacematToBowl	56.0	57.5	52.0	62.0	58.0	55.0	64.0
FromPlacematToPlate	32.0	63.0	50.0	52.0	74.0	59.0	74.0
FromPlacematToTieredshelf	18.0	28.5	28.0	24.0	26.0	24.0	28.0
FromPlateToBowl	56.0	57.0	52.0	54.0	50.0	53.0	56.0
FromPlateToCardboardbox	58.0	43.5	40.0	48.0	56.0	43.0	58.0
FromPlateToPan	68.0	51.0	36.0	54.0	70.0	55.0	68.0
FromPlateToPlate	58.0	78.7	48.0	78.0	76.0	61.0	72.0
FromTrayToCardboardbox	38.0	51.5	34.0	50.0	52.0	65.0	52.0
FromTrayToPlate	56.0	71.0	64.0	58.0	64.0	63.0	78.0
FromTrayToPot	54.0	64.5	44.0	62.0	70.0	55.0	48.0
FromTrayToTieredbasket	46.0	57.0	50.0	44.0	60.0	51.0	56.0
FromTrayToTieredshelf	38.0	31.5	28.0	22.0	28.0	33.0	36.0
Average	50.8	47.6	43.9	52.6	55.1	55.4	56.8

Table 7: **Implementation details on LIBERO.**

Configuration	Setting
Model parameters	1.55B total
Action DiT	24 layers per execution path, with 2 shared and 22 branch-specific layers; hidden dim. 1024
Action horizon	8
Optimizer	AdamW, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 10^{-8}
Learning rate	1×10^{-5} for both VLM and Action Expert
Batch size	16 per GPU on 4 H100 GPUs; global batch size 64
Training steps	80K optimizer steps
Discrete objective	$\lambda_{\text{disc}} = 0.1$
Discrete corruption	Beta$(1.5, 1.0)$-based schedule; mask positions are sampled uniformly without replacement
Continuous time sampling	Beta$(1.5, 1.0)$-based paired time schedule with $s = 0.999$
Evaluation	50 rollouts per task; replanning every 8 executed actions
Reported average	Arithmetic mean over the four LIBERO suite success rates

Table 8: **Ablation on the number of shared Action DiT layers on LIBERO-Long.**

# Shared Layers	Success Rate (%)
0	88.2
2	**95.2**
4	95.0
6	93.6
8	93.4
10	90.0
22	89.0

Table 9: **Ablation on the composition of the discrete-guided continuous source on LIBERO-Long.** α controls the contribution of the dequantized discrete action, while $1 - \alpha$ controls the Gaussian noise.

Discrete Action (α)	Gaussian Noise ($1 - \alpha$)	Success Rate (%)
0.3	0.7	**95.2**
0.5	0.5	93.2
0.7	0.3	91.0