SOAR: TEMPORAL STRUCTURE-GUIDED ONE-STEP ACTION GENERATION FOR ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient robotic manipulation requires temporally coordinated actions with low inference latency. Multi-step diffusion and flow policies progressively refine action trajectories but require costly iterative inference, whereas one-step policies gain efficiency at the expense of temporal refinement. We introduce **SOAR**, a structure-guided one-step generative policy that improves action quality while preserving one-step efficiency. SOAR predicts an observation-conditioned discrete cosine transform (DCT) representation of future actions to capture temporal variations across multiple frequency bands to provide explicit structural guidance. Unlike velocity-parameterized one-step flow policies that recover clean actions indirectly from a predicted velocity field, SOAR directly generates clean action trajectories in action space. This aligns the model output with the executable control target and provides a direct interface for structural calibration. The predicted structure further corrects the generated trajectory according to their spectral discrepancy, without requiring an additional evaluation of the generative backbone. During training, the corresponding MeanFlow velocity is derived from the calibrated clean action and used for consistency learning, while an endpoint objective aligns training with the pure-noise condition used for one-step inference. Across 53 Adroit and Meta-World tasks, SOAR achieves 83.9% overall success with only 8.7 ms inference latency. When applied to the VLA model, $\pi_{0.5}$, SOAR further achieves 97.4% on LIBERO and 60.0% on RoboTwin 2.0. Real-world experiments and extensive ablations further validate its effectiveness and efficiency.

1 INTRODUCTION

Learning visuomotor policies from expert demonstrations enables robots to acquire complex manipulation skills without manually designing task-specific controllers. However, given the same observation and task goal, multiple action trajectories may succeed while differing in approach direction, motion timing, or contact strategy. Unlike deterministic behavior cloning, which can average across distinct action modes, diffusion- and flow-based policies address this ambiguity by modeling the action distribution itself and have shown strong performance in robotic manipulation (Chi et al., 2023; Zhang et al., 2025b).

Despite their effectiveness, generative policies often rely on iterative inference. Diffusion policies progressively denoise action trajectories, while flow-based policies typically integrate learned velocity fields over multiple steps. Such interactive strategy repeats network evaluations, introducing substantial inference latency that reduces responsiveness to dynamics and can leave predicted actions stale as the scene evolves (Black et al., 2025c). Some recent efforts seek to reduce inference steps through accelerated sampling, distillation, consistency learning, or shortened flow trajectories (Song et al., 2021; Salimans & Ho, 2022; Prasad et al., 2024; Hu et al., 2024; Su et al., 2025), even pushing them to a single network evaluation (Geng et al., 2025; Sheng et al., 2026; Fang et al., 2026). However, such efficiency comes at the cost of iterative refinement. Multi-step generation repeatedly revisits the action trajectory, progressively correcting its temporal organization; a one-step policy must instead determine the entire trajectory structure at once, with no subsequent opportunity for correction.

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.

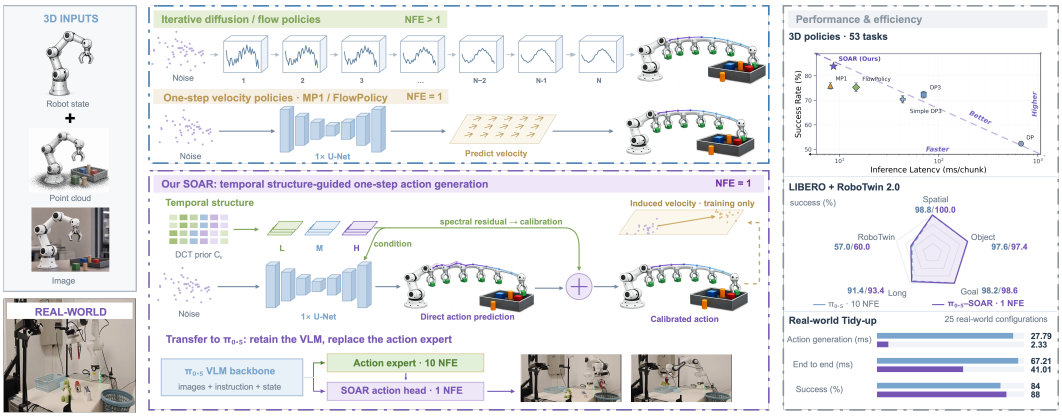


Figure 1: **Teaser of SOAR.** Iterative diffusion and flow policies require multiple refinement steps, while existing one-step policies typically generate actions through a learned velocity field. SOAR instead directly predicts a clean action chunk with a single U-Net evaluation, guided and calibrated by an observation-conditioned frequency representation; the corresponding transport velocity is used only for training. The same design replaces the multi-step action expert of $\pi_{0.5}$ while retaining its VLM backbone. Across Adroit (Rajeswaran et al., 2018), Meta-World (Yu et al., 2020), LIBERO (Liu et al., 2023a), RoboTwin2.0 Chen et al. (2025), and real-world steps, SOAR achieves competitive success rates with substantially lower action-generation latency at $\mathrm{NFE} = 1$.

To recover the temporal structure otherwise progressively refined through iterative generation, we propose **SOAR**, a temporal **S**tructure-guided **O**ne-step **A**ction generation framework for **R**obot manipulation, as shown in Fig. 1. Unlike velocity-parameterized one-step flow policies that recover clean actions indirectly, SOAR directly generates clean action trajectories in action space, aligning the model output with the executable control target and enabling direct structural calibration. Given the current observation, SOAR predicts a multi-band discrete cosine transform (DCT) representation of the future action sequence to capture its temporal structure (Ahmed et al., 1974; Zhong et al., 2025). This representation conditions action generation and calibrates the generated trajectory according to their spectral discrepancy, without additional evaluation of the generative backbone. During training, the calibrated action is mapped to the corresponding MeanFlow velocity for consistency and endpoint supervision. SOAR thus improves trajectory quality while retaining $\mathrm{NFE} = 1$ inference and millisecond-scale latency.

Our contributions are:

- We propose **SOAR**, a direct one-step action generation policy that generates clean action trajectories in action space with $\mathrm{NFE} = 1$ and millisecond-scale inference.
- We introduce **temporal structure-guided generation and calibration**, which explicitly models and corrects trajectory structure lost in one-step generation without additional backbone evaluations.
- SOAR consistently outperforms compared one-step policies on Adroit and Meta-World while retaining $\mathrm{NFE} = 1$, and further demonstrates competitive performance in the VLA setting on LIBERO and RoboTwin2.0.

2 Related Work

Visual Inputs for Robot Learning. Visuomotor policies commonly operate on either image observations or explicit 3D scene representations. Image-based methods such as Behavior Transformer, ACT, and RT-1 use learned visual representations for multimodal action prediction, action chunking, and large-scale robot control (Shafiullah et al., 2022; Zhao et al., 2023; Brohan et al., 2023). Because images encode geometry only through projection, several works instead reason explicitly in 3D. PerAct, Act3D, RVT, and GNFactor represent scenes using voxelized features, continuous 3D features, virtual views, or neural feature fields (Gervet et al., 2023; Goyal et al., 2023; Ze et al.,

2023). Explicit 3D representations have also been combined with generative action models, including 3D Diffuser Actor, DP3, and EquiBot (Ke et al., 2024; Ze et al., 2024; Yang et al., 2025). These studies highlight the value of 3D geometry for precise and data-efficient manipulation, although observation modality alone does not determine action-generation cost.

Diffusion-based Visuomotor Policies. Diffusion policies represent multimodal action distributions through progressive denoising (Ho et al., 2020). Diffusion Policy and subsequent methods such as ChainedDiffuser and HDP use iterative refinement to generate structured action trajectories (Chi et al., 2023; Xian et al., 2023; Ma et al., 2024). Reducing this inference cost has motivated accelerated samplers and distillation. DDIM shortens the sampling trajectory, while progressive distillation compresses multiple denoising steps into fewer evaluations (Song et al., 2021; Salimans & Ho, 2022). Consistency Models further enable one- or few-step sampling by enforcing consistency along the generative trajectory (Song et al., 2023; Song & Dhariwal, 2024). In robot learning, Consistency Policy, ManiCM, SDM Policy, and OneDP apply related consistency, distribution-matching, or distillation objectives to accelerate action generation (Prasad et al., 2024; Gao et al., 2026; Jia et al., 2024; Wang et al., 2024). These methods reduce the cost of iterative denoising, while many one-step variants obtain their one-step mapping by compressing or distilling a multi-step generative process.

Flow-based Visuomotor Policies. Flow Matching models generation as transport under a learned time-dependent velocity field (Lipman et al., 2023). Standard flow inference numerically integrates this field and therefore requires repeated network evaluations. Rectified Flow reduces integration difficulty by encouraging straighter transport paths, while AdaFlow adaptively determines the number of integration steps (Liu et al., 2023b; Hu et al., 2024). Consistency Flow Matching instead imposes consistency along flow trajectories to support efficient generation (Yang et al., 2024). For robotic manipulation, FlowPolicy applies consistency-based flow generation to point-cloud-conditioned control, and FreqPolicy introduces frequency-domain constraints for action-sequence consistency (Zhang et al., 2025b; Su et al., 2025). MeanFlow removes numerical integration at inference by directly predicting interval-averaged velocity (Geng et al., 2025), providing the basis for native one-step policies such as MP1, OMP, and OFP (Sheng et al., 2026; Fang et al., 2026; Li et al., 2026b). SOAR builds on this one-step formulation but introduces explicit trajectory-structure conditioning and direct action-space correction to improve the generated trajectory without additional sampling iterations.

3 METHOD

3.1 OVERVIEW

Given the latest T_o observations $\mathbf{o}_{1:T_o} = \{\mathbf{P}_{1:T_o}, \mathbf{q}_{1:T_o}\}$, we denote the normalized demonstrated action chunk as $\mathbf{x} = \mathrm{Norm}(\mathbf{A}^{\mathrm{gt}}_{1:T}) \in \mathbb{R}^{B\times T\times D_a}$. An observation encoder E_ϕ maps the observations to a global condition $\mathbf{c} = E_\phi(\mathbf{o}_{1:T_o})$.

As shown in Fig. 2, SOAR directly generates clean action trajectories in action space rather than recovering them from a predicted velocity field. An observation-conditioned frequency representation provides temporal guidance for one-step generation, and the resulting action is further calibrated according to its spectral discrepancy with the predicted structure. During training, the calibrated action induces the corresponding MeanFlow transport velocity for consistency learning and inference-endpoint alignment, while inference requires only a single evaluation of the action generator.

The overall objective is

$$\mathcal{L}_{\mathrm{SOAR}} = \lambda_{\mathrm{struct}}\mathcal{L}_{\mathrm{struct}} + \lambda_{\mathrm{act}}\mathcal{L}_{\mathrm{act}} + \lambda_{\mathrm{dis}}\mathcal{L}_{\mathrm{dis}} + \mathcal{L}_{\mathrm{MF}} + \lambda_{\mathrm{end}}\mathcal{L}_{\mathrm{end}}. \tag{1}$$

Here, $\mathcal{L}_{\mathrm{struct}}$ supervises trajectory structure, $\mathcal{L}_{\mathrm{act}}$ constrains the calibrated action, $\mathcal{L}_{\mathrm{MF}}$ is the standard MeanFlow objective, $\mathcal{L}_{\mathrm{dis}}$ is the dispersive feature regularizer inherited from MP1 (Sheng et al., 2026), and $\mathcal{L}_{\mathrm{end}}$ aligns training with the one-step inference condition.

3.2 FREQUENCY-DOMAIN TRAJECTORY-STRUCTURE LEARNING

A one-step generator must determine the temporal organization of an entire action chunk without iterative refinement. We therefore predict its structure explicitly in the frequency domain. Using the

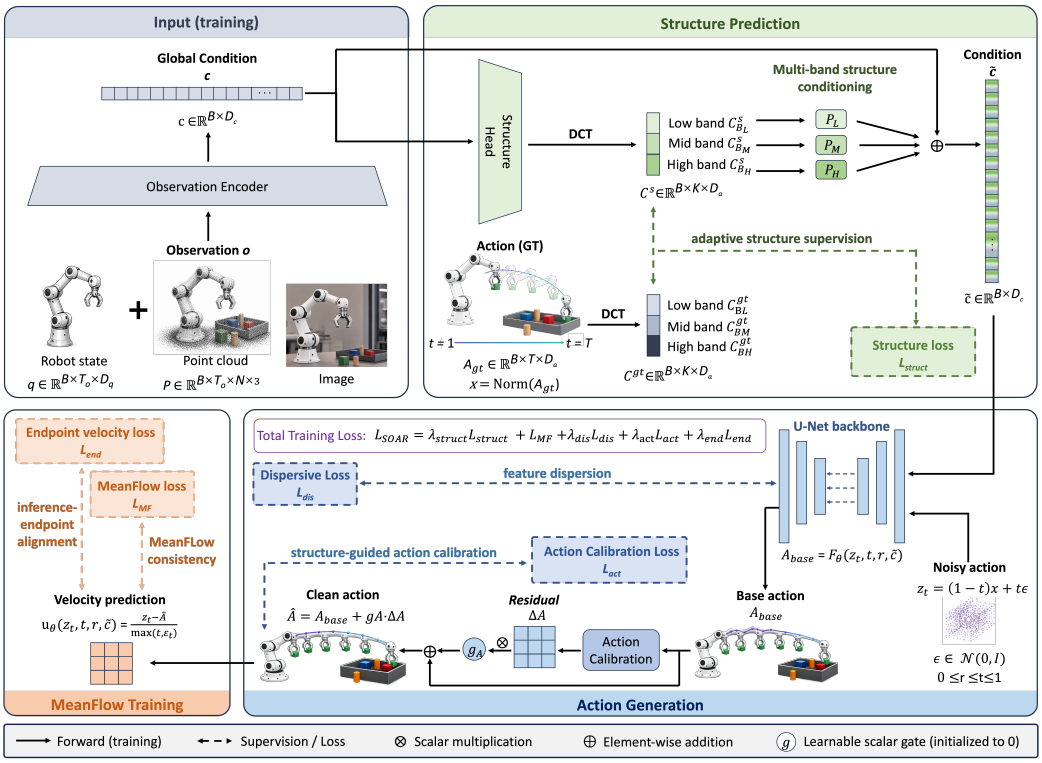


Figure 2: Overview of **SOAR**. Given an observation, SOAR predicts the frequency-domain structure of the future action trajectory and injects its multi-band features into the action generator. The conditioned U-Net directly generates a clean action trajectory, which is further calibrated using the predicted structure. During training, the calibrated action induces the corresponding MeanFlow transport velocity for consistency learning and inference-endpoint alignment. Inference requires a single U-Net evaluation.

discrete cosine transform (DCT) (Ahmed et al., 1974), $\mathbf{C}^{\mathrm{gt}} = \mathrm{DCT}(\mathbf{x})$, $\mathbf{C}^{s} = H_{\psi}(\mathbf{c})$, where $\mathbf{C}^{\mathrm{gt}}$ is used only for training supervision and $\mathbf{C}^{s}$ is predicted from the observation.

Multi-band structure conditioning. To capture temporal variations at different scales, we decompose $\mathbf{C}^{s}$ into low-, mid-, and high-frequency components and project them into the observation feature space. The resulting multi-band features are fused with the observation condition to obtain a structure-enhanced representation $\tilde{\mathbf{c}}$, which conditions the subsequent one-step action generator. The frequency partition and fusion mechanism are described in Appendix A.1.1.

Adaptive structure supervision. DCT coefficients can differ substantially in magnitude, causing a uniform spectral objective to overemphasize high-energy components. We therefore weight each coefficient according to its prediction error relative to its target energy:

$$\mathcal{L}_{\mathrm{struct}} = \frac{1}{B}\sum_{b=1}^{B}\sum_{k=0}^{K-1}\omega_{b,k}e_{b,k}. \tag{2}$$

Here, $e_{b,k}$ denotes the spectral prediction error and $\omega_{b,k}$ its adaptive weight. Frequencies that remain difficult to predict relative to their own scale receive stronger supervision. Their exact definitions are given in Appendix A.1.1. The predicted spectrum $\mathbf{C}^{s}$ is subsequently used for both action conditioning and calibration.

3.3 Structure-Guided Action Generation and Calibration

For training, we follow the standard linear probability path used in flow-based generative modeling (Lipman et al., 2023; Geng et al., 2025) to construct an intermediate noisy action state $\mathbf{z}_t$ between the demonstrated action and Gaussian noise. Details are provided in Appendix A.1.2.

Conditioned on the structure-enhanced representation $\widetilde{\mathbf{c}}$, a FiLM-conditioned 1D U-Net (Perez et al., 2018; Chi et al., 2023) directly predicts the clean base action

$$\mathbf{A}_{\text{base}} = F_\theta(\mathbf{z}_t, t, r, \widetilde{\mathbf{c}}). \tag{3}$$

Unlike velocity-parameterized policies that recover the action from a predicted transport field, F_θ directly outputs the clean action trajectory in action space.

Structure-guided action calibration. Structural conditioning alone does not explicitly guarantee that the generated trajectory follows the predicted structure. We therefore calibrate the base action according to their spectral discrepancy:

$$\widehat{\mathbf{A}} = \mathbf{A}_{\text{base}} + \tanh(\rho)\,\text{IDCT}\left[\mathbf{C}^s - \text{DCT}(\mathbf{A}_{\text{base}})\right]. \tag{4}$$

The spectral residual measures the mismatch between the predicted trajectory structure and the generated action, while $\text{IDCT}(\cdot)$ maps this correction back to action space. The learnable scalar ρ controls the correction magnitude and is initialized to zero, giving $\widehat{\mathbf{A}} = \mathbf{A}_{\text{base}}$ at the beginning of training. This provides a lightweight structure-constrained correction without introducing a second action generator.

The calibrated action is directly supervised by

$$\mathcal{L}_{\text{act}} = \text{SmoothL1}(\widehat{\mathbf{A}}, \mathbf{x}). \tag{5}$$

We additionally retain the dispersive feature regularizer $\mathcal{L}_{\text{dis}}$ from MP1 (Sheng et al., 2026), which acts on intermediate U-Net representations. Its formulation is provided in Appendix A.1.4.

3.4 MeanFlow Consistency and Endpoint Alignment

The calibrated action is further constrained through MeanFlow during training. Rather than predicting an independent velocity field, SOAR derives the corresponding transport velocity directly from the action:

$$\mathbf{u}_\theta(\mathbf{z}_t, t, r, \widetilde{\mathbf{c}}) = \frac{\mathbf{z}_t - \widehat{\mathbf{A}}}{\max(t, \varepsilon_t)}, \qquad \varepsilon_t = 10^{-5}. \tag{6}$$

Here, $\mathbf{u}_\theta$ denotes the generative transport velocity rather than the robot's physical velocity.

MeanFlow consistency. We follow the standard random-interval MeanFlow training objective $\mathcal{L}_{\text{MF}}$ (Geng et al., 2025; Sheng et al., 2026) to constrain the induced transport field. The interval sampling, self-guided target, JVP computation, and AdaptiveL2 regression follow the standard formulation and are summarized in Appendix A.1.2.

Inference-endpoint alignment. MeanFlow is trained over random (t, r) intervals, whereas one-step inference always operates at the fixed boundary $(t, r) = (1, 0)$. We therefore evaluate the same generator under this condition and explicitly align its prediction with the demonstrated action:

$$\mathcal{L}_{\text{end}} = \text{AdaptiveL2}\left(\mathbf{x} - \widehat{\mathbf{A}}_{\text{end}}\right), \tag{7}$$

where $\widehat{\mathbf{A}}_{\text{end}}$ is the calibrated action predicted under the actual one-step inference condition. This objective is equivalent to matching the corresponding endpoint transport velocities at $t = 1$; the derivation is given in Appendix A.1.3.

At inference, SOAR predicts $\mathbf{C}^s$ from the observation, constructs $\widetilde{\mathbf{c}}$, samples a Gaussian action latent, and evaluates $F_\theta(\boldsymbol{\epsilon}, 1, 0, \widetilde{\mathbf{c}})$ once. The resulting base action is then calibrated using equation 4. No JVP, iterative denoising, or numerical ODE integration is required, yielding $\text{NFE} = 1$.

4 Experiments

4.1 Simulation Experiment With 3D Inputs

Experimental Setup. We evaluate SOAR on two widely used 3D robotic manipulation benchmarks, Adroit (Rajeswaran et al., 2018) and Meta-World (Yu et al., 2020), following the evaluation protocols of prior work (Fang et al., 2026; Su et al., 2025). We consider two settings: 37 tasks consisting

Table 1: Performance on the 37-task benchmark with 3D inputs. We evaluate 37 challenging tasks over three random seeds and report the mean success rate (%) and standard deviation. Results marked with $*$ are reproduced. **Bold** denotes the best result.

		Adroit (3)				Meta-World (34)					
Methods	**NFE**	**Hammer**	**Door**	**Pen**	**Average**	**Easy (21)**	**Medium (4)**	**Hard (4)**	**Very Hard (5)**	**Average**	**Overall**
DP (Chi et al., 2023)	10	16.0 ± 10.0	34.0 ± 11.0	13.0 ± 2.0	21.0	50.7 ± 6.1	11.0 ± 2.5	5.25 ± 2.5	22.0 ± 5.0	36.5	35.2 ± 5.3
DP3 (Ze et al., 2024)	10	100.0 ± 0.0	56.0 ± 5.0	46.0 ± 10.0	67.3	87.3 ± 2.2	44.5 ± 8.7	32.7 ± 7.7	39.4 ± 9.0	68.8	68.7 ± 4.7
Simple DP3 (Ze et al., 2024)	10	98.0 ± 2.0	40.0 ± 17.0	36.0 ± 4.0	58.0	86.8 ± 2.3	42.0 ± 6.5	38.7 ± 7.5	35.0 ± 11.6	68.3	67.4 ± 5.0
AdaFlow (Hu et al., 2024)	–	45.0 ± 11.0	27.0 ± 6.0	18.0 ± 6.0	30.0	49.4 ± 6.8	12.0 ± 5.0	5.75 ± 4.4	24.0 ± 4.8	36.1	35.6 ± 6.1
CP (Prasad et al., 2024)	1	45.0 ± 4.0	31.0 ± 10.0	13.0 ± 6.0	29.7	69.3 ± 4.2	21.2 ± 6.0	17.5 ± 3.9	30.0 ± 4.9	51.8	50.1 ± 4.7
FlowPolicy (Zhang et al., 2025b)	1	98.0 ± 1.0	61.0 ± 2.0	54.0 ± 4.0	71.0	84.8 ± 2.2	58.2 ± 7.9	40.2 ± 4.5	52.2 ± 5.0	71.6	71.6 ± 3.5
MP1 (Sheng et al., 2026)	1	100.0 ± 0.0	69.0 ± 2.0	58.0 ± 5.0	75.7	88.2 ± 1.1	68.0 ± 3.1	58.1 ± 5.0	67.2 ± 2.7	79.2	78.9 ± 2.1
OMP (Fang et al., 2026)	1	100.0 ± 0.0	68.0 ± 3.0	60.0 ± 4.0	76.0	89.7 ± 0.7	$\mathbf{77.4 \pm 2.2}$	62.5 ± 3.1	$\mathbf{77.8 \pm 3.0}$	83.3	82.3 ± 1.6
SOAR (Ours)	**1**	$\mathbf{100.0 \pm 0.0}$	$\mathbf{77.5 \pm 2.4}$	$\mathbf{61.9 \pm 7.3}$	$\mathbf{79.8 \pm 2.9}$	$\mathbf{92.3 \pm 0.5}$	76.8 ± 0.7	$\mathbf{63.5 \pm 2.2}$	76.9 ± 1.8	$\mathbf{84.8 \pm 0.7}$	$\mathbf{84.4 \pm 0.5}$

Table 2: Performance on the 53-task benchmark with 3D inputs. We evaluate 53 challenging tasks over three random seeds and report the mean success rate (%) and standard deviation. Average inference latency is reported in milliseconds. Results marked with $*$ are reproduced. **Bold** denotes the best result.

		Adroit (3)				Meta-World (50)						
Methods	**NFE**	**Hammer**	**Door**	**Pen**	**Average**	**Easy (28)**	**Medium (11)**	**Hard (6)**	**Very Hard (5)**	**Average**	**Overall**	**Inference Latency (ms)**
DP* (Chi et al., 2023)	100	75.1 ± 3.5	23.7 ± 3.7	19.3 ± 1.8	39.4 ± 1.5	69.0 ± 1.6	32.8 ± 2.4	34.3 ± 2.9	31.5 ± 5.7	53.1 ± 1.1	52.3 ± 0.9	678.4 ± 6.9
DP3* (Ze et al., 2024)	10	100.0 ± 0.0	73.5 ± 2.1	56.7 ± 2.6	76.7 ± 1.6	83.1 ± 0.7	58.7 ± 3.6	54.1 ± 3.9	60.7 ± 1.5	72.0 ± 1.5	72.3 ± 1.4	70.7 ± 2.6
Simple DP3* (Ze et al., 2024)	10	100.0 ± 0.0	56.8 ± 1.2	56.1 ± 0.3	71.0 ± 0.4	82.0 ± 1.1	53.8 ± 3.1	54.3 ± 5.3	61.7 ± 4.5	70.4 ± 1.6	70.4 ± 1.5	43.4 ± 0.4
ManiCM (Gao et al., 2026)	1	100.0 ± 0.0	68.0 ± 1.0	49.0 ± 4.0	72.3	83.6	55.6	33.3	67.0	69.7	69.0 ± 4.6	–
SDM (Jia et al., 2024)	1	100.0 ± 0.0	73.0 ± 2.0	49.0 ± 4.0	74.0	86.5	65.8	35.8	71.6	74.4	74.8 ± 4.5	–
FlowPolicy* (Zhang et al., 2025b)	1	95.2 ± 2.4	57.3 ± 6.2	47.8 ± 11.0	66.8 ± 6.4	90.6 ± 0.5	56.0 ± 1.0	54.4 ± 11.3	62.7 ± 2.4	75.9 ± 1.6	75.3 ± 1.8	14.7 ± 0.0
FreqPolicy (Su et al., 2025)	1	98.0 ± 2.0	68.0 ± 5.0	52.0 ± 4.0	72.7	90.6	65.8	46.8	**79.4**	78.8	78.5 ± 2.6	–
MP1* (Sheng et al., 2026)	1	96.1 ± 2.3	69.2 ± 3.1	57.1 ± 6.1	74.1 ± 2.0	88.5 ± 0.8	59.5 ± 2.9	56.1 ± 3.2	65.7 ± 1.3	76.0 ± 1.5	75.9 ± 1.3	$\mathbf{8.1 \pm 0.0}$
SOAR (Ours)	**1**	$\mathbf{100.0 \pm 0.0}$	$\mathbf{77.5 \pm 2.4}$	$\mathbf{61.9 \pm 7.3}$	$\mathbf{79.8 \pm 2.9}$	$\mathbf{93.7 \pm 0.7}$	$\mathbf{74.3 \pm 0.6}$	$\mathbf{63.5 \pm 1.8}$	76.9 ± 1.8	$\mathbf{84.1 \pm 0.5}$	$\mathbf{83.9 \pm 0.3}$	8.7 ± 0.1

of 3 Adroit and 34 Meta-World tasks, and 53 tasks consisting of 3 Adroit and all 50 Meta-World tasks. For each task, we use 10 expert demonstrations for training and downsample the point cloud to 512 points. Each task is trained independently for 3,000 epochs with three random seeds (0, 1, and 2). All training and evaluation are conducted on a single NVIDIA RTX 4090 GPU. We use a batch size of 128 and optimize the policy with AdamW using a learning rate of 1×10^{-4}. The observation window and prediction horizon are set to 2 and 4, respectively. We evaluate the policy every 200 epochs. For each random seed, the five highest evaluation success rates are selected and averaged, and the final task performance is reported as the mean and sample standard deviation across the three seeds. SOAR performs one-step action generation at inference time with $\mathrm{NFE} = 1$.

Results. Tables 1 and 2 summarize the simulation results. SOAR achieves 84.4% and 83.9% overall success on the 37-task and 53-task benchmarks, respectively, while retaining $\mathrm{NFE} = 1$. The small performance change after expanding the evaluation from 37 to 53 tasks indicates that SOAR remains robust under broader task coverage. On the full 53-task benchmark, SOAR reaches 84.1% on Meta-World and shows particularly strong performance on the Medium and Hard subsets, outperforming the best baselines by 8.5% and 7.4%, respectively. This suggests that the advantage of SOAR is not limited to easier tasks near saturation, but extends to more challenging manipulation settings. The latency results in Table 2 and Fig. 1 show that these gains are obtained while retaining one-step inference. SOAR requires only 8.7 ms of inference latency, remaining close to the fastest one-step baseline, MP1, while improving overall success by 8.0%. Compared with DP3, SOAR reduces inference latency from 70.7 ms to 8.7 ms while increasing overall success from 72.3% to 83.9%. These results show that SOAR improves one-step action generation without relying on iterative refinement, providing a favorable performance–efficiency trade-off. Detailed per-task and reproduced results are reported in Appendix A.2.

4.2 Simulation Experiments with VLA Settings

Experimental Setup. We further evaluate whether SOAR can be transferred from 3D visuomotor policies to large-scale vision-language-action models. Specifically, we replace the original action expert of $\pi_{0.5}$ (Black et al., 2025a) with SOAR and evaluate the resulting policy on LIBERO (Liu et al., 2023a) and RoboTwin 2.0 (Chen et al., 2025). For both benchmarks, the vision-language backbone is initialized from the OpenPI pi05_base checkpoint, while the SOAR action module is

Table 3: Comparison with representative VLA methods on LIBERO. Results are success rates (%). **Bold** denotes the best result within each NFE group.

Methods	Publication	NFE	Spatial	Object	Goal	Long	Average
UniVLA (Bu et al., 2025)	RSS, 2025	AR	96.5	96.8	95.6	92.0	95.2
OpenVLA-OFT (Kim et al., 2025)	RSS, 2025	AR	97.6	98.4	97.9	94.5	97.1
VLA-Cache (Xu et al., 2025)	NeurIPS, 2025	AR	**98.3**	97.5	**98.3**	**95.4**	**97.4**
UnifiedVLA (Wang et al., 2026)	ICLR, 2026	AR	95.4	**98.8**	93.6	94.0	95.5
Dita (Hou et al., 2025)	ICCV, 2025	20	84.2	96.3	85.4	63.8	82.4
CronusVLA (Li et al., 2026a)	AAAI, 2026	100	**97.3**	**99.6**	**96.9**	**94.0**	**97.0**
DreamVLA (Zhang et al., 2025c)	NeurIPS, 2025	10	97.5	94.0	89.5	89.5	92.6
$\pi^*_{0.5}$ (Black et al., 2025a)	CoRL, 2025	10	**98.8**	97.6	98.2	91.4	96.5
FLOWER (Reuss et al., 2025)	CoRL, 2025	8	97.5	99.1	96.1	94.9	96.9
$\pi_{0.5}$ (Black et al., 2025a)	CoRL, 2025	10	**98.8**	98.2	98.0	92.4	96.9
InstructVLA (Yang et al., 2026)	ICLR, 2026	10	97.3	99.6	96.5	89.8	95.8
MemoryVLA (Shi et al., 2026)	ICLR, 2026	10	98.4	98.4	96.4	93.4	96.5
X-VLA (Zheng et al., 2026)	ICLR, 2026	10	98.2	98.6	97.8	**97.6**	98.1
Cosmos Policy (Kim et al., 2026)	ICLR, 2026	10	98.1	**100.0**	**98.2**	**97.6**	**98.5**
UniAct (Zheng et al., 2025)	CVPR, 2025	1	77.0	87.0	77.0	70.0	77.8
MolmoAct-7B-D (Lee et al., 2025)	arXiv, 2025	1	87.0	95.4	87.6	77.2	86.6
NinA (MLP) (Tarasov et al., 2025)	NeurIPS WS, 2025	1	87.8	98.2	90.2	92.8	90.9
EveryDayVLA (Chopra et al., 2025)	ICRA subm., 2026	1	96.8	95.6	91.0	82.0	91.4
PI-VLA (Jian et al., 2026)	Symmetry, 2026	1	79.5	73.4	73.3	66.6	73.2
MIP (π_0 arch.) (Pan et al., 2026)	ICLR, 2026	2	97.6	95.8	95.2	82.2	92.7
CF-VLA (Du et al., 2026)	ACM MM,2026	2	98.0	**99.2**	96.6	92.0	96.5
$\pi_{0.5}$**-SOAR**	–	**1**	**100.0**	97.4	**98.6**	**93.4**	**97.4**

Table 4: Comparison with representative methods on RoboTwin 2.0 under the clean setting. Results are average success rates (%). NFE denotes the number of action-generation network evaluations required for one policy inference. **Bold** denotes the best result.

Methods	Publication	Params	NFE	Success
Diffusion Policy (Chi et al., 2023)	RSS, 2023	0.1B	100	28.0
SeedPolicy (Gui et al., 2026)	arXiv, 2026	0.2B	100	**42.8**
DP3 (Ze et al., 2024)	RSS, 2024	0.3B	10	55.2
π_0 (Black et al., 2025b)	RSS, 2025	3.2B	10	46.4
$\pi_{0.5}$ (Black et al., 2025a)	CoRL, 2025	3.4B	10	**57.0**
ACT (Zhao et al., 2023)	RSS, 2023	0.1B	1	29.7
RDT (Liu et al., 2025)	ICLR, 2025	1.7B	5	34.5
FlowPolicy (Zhang et al., 2025b)	AAAI, 2025	0.3B	1	41.0
UP-VLA (Zhang et al., 2025a)	ICML, 2025	1.6B	1	52.9
StarVLA-α (Ye et al., 2026)	ECCV, 2026	3.8B	1	50.3
$\pi_{0.5}$**-SOAR**	–	2.9B	**1**	**60.0**

initialized from scratch. On LIBERO, we jointly train a single policy over all 40 tasks from the four standard suites. SOAR predicts 10-step action chunks and executes five actions before replanning. On RoboTwin 2.0, we jointly train a single policy over all 50 tasks under the demo_clean setting, with SOAR predicting 50-step joint-position action chunks. In both settings, each policy query requires only one action-expert evaluation ($\mathrm{NFE} = 1$). Detailed training and evaluation configurations are provided in Appendix A.3.

Results. Table 3 reports the results on LIBERO. Replacing the original $\pi_{0.5}$ action expert with SOAR reduces the number of action-generation evaluations from 10 to 1 while increasing the average success rate from 96.5% to 97.4%. With $\mathrm{NFE} = 1$, $\pi_{0.5}$-SOAR achieves 100.0% on Spatial, 97.4% on Object, 98.6% on Goal, and 93.4% on Long, outperforming the other one-step methods in average success rate. Its overall performance also remains close to higher-NFE methods such as X-VLA and Cosmos Policy. We further evaluate the same formulation on RoboTwin 2.0, where a single policy is jointly trained across 50 tasks and predicts substantially longer action chunks. As shown in Table 4, $\pi_{0.5}$-SOAR achieves an average success rate of 60.0% with $\mathrm{NFE} = 1$. Together with the LIBERO results, these experiments show that SOAR can replace an iterative VLA action expert with one-step action generation while maintaining competitive task performance across different multi-task settings.

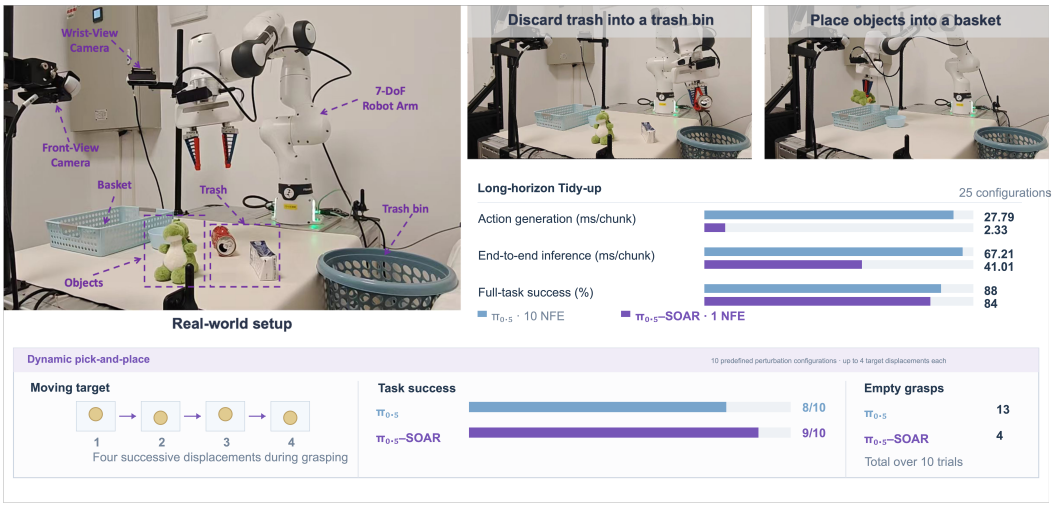


Figure 3: Real-world evaluation of $\pi_{0.5}$ and $\pi_{0.5}$-SOAR. **Left:** Experimental setup with a 7-DoF robot arm, wrist-view and front-view cameras, tabletop objects, a storage basket, and a trash bin. **Top right:** Example stages of the long-horizon Tidy-up task, including trash disposal and object placement. $\pi_{0.5}$-SOAR achieves 84% full-task success with $\mathrm{NFE} = 1$, compared with 88% for $\pi_{0.5}$ with $\mathrm{NFE} = 10$, while reducing action-generation latency from 27.79 ms to 2.33 ms and end-to-end inference latency from 67.21 ms to 41.01 ms. **Bottom:** Dynamic pick-and-place with up to four successive target displacements. $\pi_{0.5}$-SOAR succeeds in 9/10 trials and records 4 empty grasps, compared with 8/10 successful trials and 13 empty grasps for $\pi_{0.5}$.

4.3 Real-World Experiments

Experimental Setup. We compare $\pi_{0.5}$ (Black et al., 2025a) with $\pi_{0.5}$-SOAR in two real-world evaluations. The first is a long-horizon Tidy-up task, where the robot sequentially sorts multiple tabletop objects by discarding trash into a trash bin and placing the remaining objects into a storage basket. Both methods are evaluated on the same 25 object configurations with different object positions and arrangements.

The second evaluation considers dynamic pick-and-place under repeated target displacements. The robot is required to grasp a toy and place it into a fixed basket, while the target may be displaced up to four times during the grasping stage. We use 10 predefined perturbation configurations with the same displacement schedule for both policies.

Both methods use the same observations, action representation, and deployment setting. The $\pi_{0.5}$ baseline uses 10 action-generation steps, whereas $\pi_{0.5}$-SOAR replaces the original action generator with SOAR and predicts the full 16-step action chunk with a single network evaluation ($\mathrm{NFE} = 1$). For Tidy-up, we report full-task success rate, action-generation latency, and end-to-end inference latency. For dynamic pick-and-place, we report task success and the total number of empty grasps over the 10 trials. Detailed training and evaluation protocols are provided in Appendix A.4.

Results. As shown in Fig. 3, $\pi_{0.5}$ completes 22 of the 25 Tidy-up configurations, corresponding to an 88% full-task success rate, while $\pi_{0.5}$-SOAR succeeds in 21 configurations (84%). Thus, reducing action generation from 10 evaluations to one changes the Tidy-up result by one successful trial. The main difference lies in inference cost. Action-generation latency decreases from 27.79 ms for $\pi_{0.5}$ to 2.33 ms for $\pi_{0.5}$-SOAR, corresponding to a 91.6% reduction. End-to-end inference latency decreases from 67.21 ms to 41.01 ms. Since the vision-language component is unchanged, most of the reduction comes from replacing iterative action generation with the one-step SOAR module.

Under repeated target displacements, $\pi_{0.5}$-SOAR succeeds in 9 of 10 trials, compared with 8 of 10 for $\pi_{0.5}$. More notably, the total number of empty grasps decreases from 13 to 4 over the 10 perturbation trials. Because an empty grasp corresponds to a completed gripper-closing attempt after the target has moved away from the predicted grasp location, this metric reflects how often the policy continues to act on an outdated target position. The lower empty-grasp count indicates that

Table 5: Ablation study on the three Adroit tasks. Results are reported as mean success rate (%) ± sample standard deviation over three seeds. For each task and seed, the five highest evaluation success rates are averaged. The Adroit average is computed across Hammer, Door, and Pen within each seed before calculating cross-seed statistics. SS: structure supervision; MC: multi-band conditioning; AC: action calibration; EA: inference-endpoint alignment. **Bold** denotes the highest mean in the Door, Pen, and Adroit average columns.

Variant	SS	MC	AC	EA	Hammer	Door	Pen	Adroit Avg.
Direct-action baseline	–	–	–	–	100.0 ± 0.0	73.6 ± 4.1	58.2 ± 4.3	77.3 ± 0.1
+ Structure supervision and multi-band conditioning	✓	✓	–	–	100.0 ± 0.0	75.4 ± 5.1	59.7 ± 5.3	78.4 ± 1.9
+ Action calibration	✓	✓	✓	–	100.0 ± 0.0	77.1 ± 2.0	59.3 ± 6.2	78.8 ± 2.3
SOAR	✓	✓	✓	✓	100.0 ± 0.0	77.5 ± 2.4	$\mathbf{61.9 \pm 7.3}$	$\mathbf{79.8 \pm 2.9}$
SOAR w/o multi-band conditioning	✓	–	✓	✓	100.0 ± 0.0	76.7 ± 5.0	58.7 ± 4.9	78.5 ± 1.5
SOAR w/o action calibration	✓	✓	–	✓	100.0 ± 0.0	$\mathbf{77.7 \pm 3.3}$	59.5 ± 6.3	79.1 ± 2.5

$\pi_{0.5}$-SOAR more often updates its grasping behavior before committing to an unsuccessful grasp after target displacement.

4.4 Ablation Studies

Ablation Setup. We conduct ablation studies on the three Adroit tasks: Hammer, Door, and Pen. Each task is trained independently for 3,000 epochs using 10 expert demonstrations and three random seeds. Following the main evaluation protocol, for each task and seed we average the five highest evaluation success rates. The Adroit average is first computed across Hammer, Door, and Pen within each seed, and is then reported as the mean and sample standard deviation across the three seeds. We study four components of SOAR: structure supervision (SS), multi-band conditioning (MC), action calibration (AC), and inference-endpoint alignment (EA). Starting from a direct-action baseline, we first add SS and MC as the structure-guidance stage, followed by AC and EA. SS learns an explicit frequency-domain representation of the future trajectory, while MC provides the predicted band-wise structure to the action generator. We additionally remove MC and AC from the full model to examine their roles in the complete configuration. All variants share the same backbone, optimization settings, and gate configuration, and retain one-step inference ($\mathrm{NFE} = 1$).

Results. The direct-action baseline achieves a 77.3% Adroit average. Adding SS and MC increases the average to 78.4%, with Door improving from 73.6% to 75.4% and Pen from 58.2% to 59.7%. This shows that introducing explicit trajectory structure into one-step action generation provides a consistent gain over direct action prediction.

Adding AC further raises the Adroit average to 78.8%. The main improvement appears on Door, which increases from 75.4% to 77.1%, while Pen remains similar. Adding EA yields the full SOAR model with the highest Adroit average of 79.8%. Relative to the variant without EA, Door improves by 0.4 percentage points and Pen by 2.6 points.

The removal experiments further support the roles of MC and AC. Removing MC reduces the Adroit average from 79.8% to 78.5%, with lower success on both Door and Pen. Removing AC reduces the average to 79.1%; Pen decreases from 61.9% to 59.5%, while Door remains nearly unchanged. These results show that MC and AC both contribute to the complete model, although their effects vary across tasks.Hammer remains at 100.0% for all variants, indicating that the added components do not degrade the performance already achieved by the direct-action baseline on this task.

5 Conclusion

We presented SOAR, a structure-guided framework for one-step action generation in robotic manipulation. By explicitly modeling trajectory structure and using it to guide and calibrate direct action prediction, SOAR improves action quality without reintroducing iterative sampling. Across 3D manipulation benchmarks, VLA settings, and real-world experiments, SOAR consistently retains NFE=1 inference while achieving competitive task performance and low latency. These results demonstrate that explicit temporal structure provides an effective way to improve one-step generative policies while preserving their efficiency.

AI USE STATEMENT

Generative AI tools assisted with experimental interpretation, methodological feedback, manuscript drafting and editing, and literature review. The authors take full responsibility for the final content and all AI-assisted materials.

REPRODUCIBILITY STATEMENT

Section 3 describes the SOAR architecture and training objectives. Appendix A.1 details the frequency-domain structure representation, MeanFlow training, endpoint alignment, and dispersive regularization. Section 4 describes the experimental setup and evaluation metrics. Appendix A.2 provides task-wise simulation results and inference latency measurements, while Appendices A.3 and A.4 describe the training configurations and evaluation protocols for the VLA benchmarks and real-world tasks, respectively.

REFERENCES

Nasir Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

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. In *Conference on Robot Learning*, 2025a.

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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, volume 38, 2025c.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H. Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.025.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *Robotics: Science and Systems*, 2025.

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

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026.

Samarth Chopra, Alex McMoil, Ben Carnovale, Evan Sokolson, Rajkumar Kubendran, and Samuel Dickerson. EveryDayVLA: A vision-language-action model for affordable robotic manipulation. *arXiv preprint arXiv:2511.05397*, 2025.

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. In *Proceedings of the ACM International Conference on Multimedia*, 2026. doi: 10.1145/3767308.3836126.

Han Fang, Yize Huang, Yuheng Zhao, Paul Weng, Xiao Li, and Yutong Ban. OMP: One-step MeanFlow policy with directional alignment. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

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. In *2026 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 84032–84054, 2025. doi: 10.52202/085713-2534.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3D: 3D feature field transformers for multi-task robotic manipulation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3949–3965. PMLR, 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. RVT: Robotic view transformer for 3D object manipulation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 694–710. PMLR, 2023.

Youqiang Gui, Yuxuan Zhou, Shen Cheng, Xinyang Yuan, Haoqiang Fan, Peng Cheng, and Shuaicheng Liu. SeedPolicy: Horizon scaling via self-evolving diffusion policy for robot manipulation. *arXiv preprint arXiv:2603.05117*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, and Yuntao Chen. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. AdaFlow: Imitation learning with variance-adaptive flow-based policies. In *Advances in Neural Information Processing Systems*, volume 37, pp. 138836–138858, 2024. doi: 10.52202/079017-4405.

Bofang Jia, Pengxiang Ding, Can Cui, Mingyang Sun, Pengfang Qian, Siteng Huang, Zhaoxin Fan, and Donglin Wang. Score and distribution matching policy: Advanced accelerated visuomotor policies via matched distillation. *arXiv preprint arXiv:2412.09265*, 2024.

Yina Jian, Di Tian, Xuan-Jing Chen, Zhen-Yuan Wei, Chen-Wei Liang, and Mu-Jiang-Shan Wang. PI-VLA: Adaptive symmetry-aware decision-making for long-horizon vision–language–action manipulation. *Symmetry*, 18(3):394, 2026. doi: 10.3390/sym18030394.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations, 2024. URL `https://arxiv.org/abs/2402.10885`.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations*, 2026.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

Hao Li, Shuai Yang, Yilun Chen, Xinyi Chen, Xiaoda Yang, Yang Tian, Hanqing Wang, Tai Wang, Dahua Lin, Feng Zhao, and Jiangmiao Pang. Towards efficient and robust manipulation via multi-frame vision-language-action modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18388–18396, 2026a. doi: 10.1609/aaai.v40i22.38903.

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

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

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

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*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=XVjTT1nw5z.

Xiao Ma, Sumit Patidar, Iain Haughton, and Stephen James. Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18081–18090, 2024. doi: 10.1109/CVPR52733.2024.01712.

Chaoyi Pan, Giridharan 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. In *International Conference on Learning Representations*, 2026.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 3942–3951, 2018. doi: 10.1609/aaai.v32i1.11671.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.071.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018. doi: 10.15607/RSS.2018.XIV.049.

Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-flow models. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 3736–3761. PMLR, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *CoRR*, abs/2202.00512, 2022. URL https://arxiv.org/abs/2202.00512.

Nur Muhammad Mahi Shafiullah, Zichen Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22955–22968, 2022. doi: 10.52202/068431-1668.

Juyi Sheng, Ziyi Wang, Peiming Li, and Mengyuan Liu. MP1: MeanFlow tames policy learning in 1-step for robotic manipulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18532–18539, 2026. doi: 10.1609/aaai.v40i22.38919.

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*, 2026.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WNzy9bRDvG.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 2023.

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 31574–31602, 2025. doi: 10.52202/085713-0937.

Denis Tarasov, Alexander Nikulin, Ilya Zisman, Albina Klepach, Nikita Lyubaykin, Andrei Polubarov, Alexander Derevyagin, and Vladislav Kurenkov. NinA: Normalizing flows in action. training vla models with normalizing flows. *arXiv preprint arXiv:2508.16845*, 2025.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *International Conference on Learning Representations*, 2026.

Zhendong Wang, Zhaoshuo 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, 2024. URL https://arxiv.org/abs/2410.21257.

Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. ChainedDiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2323–2339. PMLR, 2023.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5484.

Jingyun Yang, Ziang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. EquiBot: $\mathrm{SIM}(3)$-equivariant diffusion policy for generalizable and data efficient learning. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1048–1068. PMLR, 2025.

Ling Yang, Zixiang Zhang, Zhilong Zhang, Xingchao Liu, Minkai Xu, Wentao Zhang, Chenlin Meng, Stefano Ermon, and Bin Cui. Consistency flow matching: Defining straight flows with velocity consistency. *arXiv preprint arXiv:2407.02398*, 2024.

Shuai Yang, Hao Li, Bin Wang, Yilun Chen, Yang Tian, Tai Wang, Hanqing Wang, Feng Zhao, Yiyi Liao, and Jiangmiao Pang. InstructVLA: Vision-language-action instruction tuning from understanding to manipulation. In *International Conference on Learning Representations*, 2026.

Jinhui Ye, Ning Gao, Senqiao Yang, Jinliang Zheng, Zixuan Wang, Yuxin Chen, Pengguang Chen, Yilun Chen, Shu Liu, and Jiaya Jia. StarVLA-α: Reducing complexity in vision-language-action systems. In *European Conference on Computer Vision*, 2026.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1094–1100. PMLR, 2020.

Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. GNFactor: Multi-task real robot learning with generalizable neural feature fields. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 284–301. PMLR, 2023.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.067.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. UP-VLA: A unified understanding and prediction model for embodied agent. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74911–74922. PMLR, 2025a.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(14):14754–14762, 2025b. doi: 10.1609/aaai.v39i14.33617.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, XinQiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. In *Advances in Neural Information Processing Systems*, volume 38, 2025c.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Jinliang Zheng, Jianxiong Li, Dongxiu Liu, Yinan Zheng, Zhihao Wang, Zhonghong Ou, Yu Liu, Jingjing Liu, Ya-Qin Zhang, and Xianyuan Zhan. Universal actions for enhanced embodied foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22508–22519, 2025.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026.

Yiming Zhong, Yumeng Liu, Chuyang Xiao, Zemin Yang, Youzhuo Wang, Yufei Zhu, Ye Shi, Yujing Sun, Xinge Zhu, and Yuexin Ma. FreqPolicy: Frequency autoregressive visuomotor policy with continuous tokens. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

A Appendix

A.1 Additional Method Details

This appendix provides additional implementation details for the frequency-domain structure module, MeanFlow training, inference-endpoint alignment, and dispersive feature regularization. These details are omitted from the main text for clarity.

A.1.1 Frequency-Domain Structure Details

Multi-band structure conditioning. Let K denote the number of DCT coefficients represented by the structure branch. We define the normalized frequency index as

$$\nu_k = \frac{k}{K-1}, \tag{8}$$

and partition the predicted spectrum into low-, mid-, and high-frequency groups:

$$\mathcal{B}_L = \left\{ k \mid \nu_k < \frac{1}{2} \right\}, \qquad \mathcal{B}_M = \left\{ k \mid \frac{1}{2} \le \nu_k < \frac{3}{4} \right\}, \qquad \mathcal{B}_H = \left\{ k \mid \nu_k \ge \frac{3}{4} \right\}. \tag{9}$$

Lower-frequency coefficients primarily describe slowly varying trajectory structure, while higher-frequency coefficients capture progressively shorter-scale temporal variations.

The three groups are independently projected into the observation feature space and fused with the global condition:

$$\widetilde{\mathbf{c}} = \mathbf{c} + g_L P_L \left(\mathbf{C}^s_{\mathcal{B}_L} \right) + g_M P_M \left(\mathbf{C}^s_{\mathcal{B}_M} \right) + g_H P_H \left(\mathbf{C}^s_{\mathcal{B}_H} \right), \tag{10}$$

where P_L, P_M, and P_H are learnable projections, and g_L, g_M, and g_H are learnable scalar gates. All gates are initialized to zero, yielding $\widetilde{\mathbf{c}} = \mathbf{c}$ at initialization. This initialization preserves the original observation condition at the beginning of training and introduces structural guidance progressively.

Adaptive structure supervision. DCT coefficients may differ substantially in scale. A uniform spectral objective can therefore be dominated by high-energy components even when lower-energy frequencies remain difficult to predict. We instead measure the prediction difficulty of each coefficient relative to its own target energy.

For sample b and frequency coefficient k, we define the prediction error and target energy as

$$e_{b,k} = \frac{1}{D_a} \sum_{d=1}^{D_a} \left(C^s_{b,k,d} - C^{\mathrm{gt}}_{b,k,d} \right)^2, \qquad E_{b,k} = \frac{1}{D_a} \sum_{d=1}^{D_a} \left(C^{\mathrm{gt}}_{b,k,d} \right)^2. \tag{11}$$

The relative prediction difficulty is

$$d_{b,k} = \frac{e_{b,k}}{\max(E_{b,k}, 10^{-6})}. \tag{12}$$

We convert the difficulty scores into normalized positive weights:

$$\omega_{b,k} = \frac{\exp\left(\gamma_s \, \mathrm{sg}[d_{b,k}]\right)}{\sum_{j=0}^{K-1} \exp\left(\gamma_s \, \mathrm{sg}[d_{b,j}]\right)}, \qquad \gamma_s = 2, \tag{13}$$

where $\mathrm{sg}[\cdot]$ denotes stop-gradient. The structure objective is

$$\mathcal{L}_{\mathrm{struct}} = \frac{1}{B} \sum_{b=1}^{B} \sum_{k=0}^{K-1} \omega_{b,k} e_{b,k}. \tag{14}$$

Thus, coefficients with larger prediction errors relative to their own target scale receive stronger supervision rather than being weighted only by their absolute magnitude.

A.1.2 MEANFLOW TRAINING DETAILS

Linear probability path. During training, we sample Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and construct the intermediate action state

$$\mathbf{z}_t = (1-t)\mathbf{x} + t\boldsymbol{\epsilon}, \qquad t \in [0,1]. \tag{15}$$

The path interpolates between the demonstrated clean action $\mathbf{x}$ at $t = 0$ and Gaussian noise at $t = 1$.

Interval sampling. Following MeanFlow (Geng et al., 2025; Sheng et al., 2026), we sample

$$a_1, a_2 \sim \mathcal{N}(-0.4, 1), \qquad s_i = \sigma(a_i), \qquad t = \max(s_1, s_2), \qquad r = \min(s_1, s_2), \tag{16}$$

where $\sigma(\cdot)$ denotes the sigmoid function. This guarantees $0 \le r \le t \le 1$. For half of the samples in each mini-batch, we further set $r = t$, combining finite-interval MeanFlow supervision with the zero-length flow-matching case.

Self-guided MeanFlow target. For the linear probability path, the instantaneous direction is

$$\mathbf{v} = \boldsymbol{\epsilon} - \mathbf{x}. \tag{17}$$

We evaluate the current transport field at (t, t) without gradient propagation and construct the self-guided direction

$$\widehat{\mathbf{v}} = w\mathbf{v} + (1-w)\,\mathrm{sg}\left[\mathbf{u}_\theta\left(\mathbf{z}_t, t, t, \widetilde{\mathbf{c}}\right)\right], \qquad w = 2. \tag{18}$$

The directional derivative of the transport field is evaluated through a Jacobian–vector product:

$$\dot{\mathbf{u}}_\theta = \mathrm{JVP}_{(\mathbf{z},t,r)}\left[\mathbf{u}_\theta; (\widehat{\mathbf{v}}, \mathbf{1}, \mathbf{0})\right]_{(\mathbf{z}_t,t,r)}. \tag{19}$$

The tangent in the r direction is set to zero.

The resulting MeanFlow target is

$$\mathbf{u}_{\mathrm{tgt}} = \mathrm{sg}\left[\widehat{\mathbf{v}} - (t-r)\dot{\mathbf{u}}_\theta\right]. \tag{20}$$

The corresponding objective is

$$\mathcal{L}_{\mathrm{MF}} = \mathrm{AdaptiveL2}\left(\mathbf{u}_\theta - \mathbf{u}_{\mathrm{tgt}}\right). \tag{21}$$

In SOAR, $\mathbf{u}_\theta$ is not predicted by an independent velocity head. It is induced by the calibrated action through equation 6 in the main text. Consequently, $\mathcal{L}_{\mathrm{MF}}$ constrains the same action that is ultimately produced by the policy.

Adaptive regression. For a per-sample residual $\boldsymbol{\delta}$, the AdaptiveL2 metric is implemented as

$$\mathrm{AdaptiveL2}(\boldsymbol{\delta}) = \frac{1}{B}\sum_{b=1}^{B} \mathrm{sg}\left[(m_b + 10^{-3})^{-1/2}\right] m_b, \qquad m_b = \mathrm{mean}\left(\boldsymbol{\delta}_b^2\right). \tag{22}$$

The detached factor adaptively rescales sample-wise regression errors while preventing gradients from modifying the weighting term itself.

A.1.3 INFERENCE-ENDPOINT ALIGNMENT DETAILS

Random-interval MeanFlow training does not explicitly optimize the fixed condition encountered during one-step inference. SOAR therefore introduces an additional objective at

$$(t, r) = (1, 0). \tag{23}$$

We independently sample

$$\boldsymbol{\epsilon}_{\mathrm{end}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{24}$$

and evaluate the same action generator under this inference condition. The corresponding ground-truth and model-induced endpoint velocities are

$$\mathbf{u}_{\mathrm{end}}^{\mathrm{gt}} = \boldsymbol{\epsilon}_{\mathrm{end}} - \mathbf{x}, \qquad \mathbf{u}_{\mathrm{end}}^{\theta} = \boldsymbol{\epsilon}_{\mathrm{end}} - \widehat{\mathbf{A}}_{\mathrm{end}}, \tag{25}$$

where $\widehat{\mathbf{A}}_{\mathrm{end}}$ denotes the calibrated action predicted at $(t, r) = (1, 0)$.

The velocity-space endpoint objective is

$$\mathcal{L}_{\text{end}} = \text{AdaptiveL2}\left(\mathbf{u}^{\theta}_{\text{end}} - \mathbf{u}^{\text{gt}}_{\text{end}}\right). \tag{26}$$

At $t = 1$, the two endpoint velocities share the same noise sample, yielding

$$\mathbf{u}^{\theta}_{\text{end}} - \mathbf{u}^{\text{gt}}_{\text{end}} = \mathbf{x} - \widehat{\mathbf{A}}_{\text{end}}. \tag{27}$$

Therefore, equation 26 is algebraically equivalent to

$$\mathcal{L}_{\text{end}} = \text{AdaptiveL2}\left(\mathbf{x} - \widehat{\mathbf{A}}_{\text{end}}\right), \tag{28}$$

which is the compact form used in the main text. The velocity-space formulation emphasizes that the objective aligns the actual MeanFlow sampling boundary rather than the robot's physical dynamics.

A.1.4 Dispersive Feature Regularization

Following MP1 (Sheng et al., 2026), we retain a dispersive regularizer on intermediate representations of the action U-Net. For hidden representation $\mathbf{h}^{(l)}$ at layer l, we compute the pairwise squared feature distance

$$d^{(l)}_{ij} = \left\|\mathbf{h}^{(l)}_i - \mathbf{h}^{(l)}_j\right\|_2^2. \tag{29}$$

Using normalized distances $\bar{d}^{(l)}_{ij}$, the regularizer is

$$\mathcal{L}_{\text{dis}} = \sum_l \log\left[\frac{1}{B^2}\sum_{i,j}\exp\left(-\frac{\bar{d}^{(l)}_{ij}}{\tau}\right)\right], \qquad \tau = 1. \tag{30}$$

Minimizing this objective encourages different samples to maintain discriminative hidden representations and helps reduce representation collapse or overly averaged predictions in one-step generation.

A.2 More Details on Simulation Experiment With 3D Inputs

A.2.1 More Details on Inference Latency Comparison

Table 6 reports the detailed inference latency on Adroit and Meta-World. Inference latency is measured as the wall-clock time required for one complete policy evaluation to generate an action chunk. All results are reported as mean $\pm$ standard deviation over three random seeds. For FlowPolicy, three repeated measurements are averaged for each seed before computing the final statistics. The latency increases with the number of network evaluations used for action generation. DP requires 678.4 ± 6.9 ms with $\text{NFE} = 100$, while DP3 and Simple DP3 require 70.7 ± 2.6 ms and 43.4 ± 0.4 ms, respectively, with $\text{NFE} = 10$. Among the one-step methods, FlowPolicy requires 14.7 ± 0.0 ms, MP1 requires 8.1 ± 0.0 ms, and SOAR requires 8.7 ± 0.1 ms. SOAR maintains similar latency across different tasks, ranging from 8.6 to 8.9 ms on Adroit and Meta-World. Compared with MP1, the additional structure prediction and action calibration introduce only a small latency increase, while the generative backbone is still evaluated once. These measurements confirm that SOAR retains millisecond-level inference with $\text{NFE} = 1$.

A.2.2 More Details on Performance

Table 7 provides the task-wise results on the 53-task benchmark. Each result is reported as mean success rate $\pm$ sample standard deviation over three random seeds. Most Easy tasks are close to saturation for several methods, while the differences become more evident on Medium, Hard, and Very Hard tasks. SOAR performs strongly on a range of non-saturated tasks, including Peg Insert Side, Push Wall, Sweep Into, Pick Out Of Hole, Shelf Place, Stick Pull, and Pick Place Wall. It also improves several Adroit and Easy tasks such as Door, Handle Pull, Handle Pull Side, and Lever Pull. Performance remains task-dependent, with other methods retaining an advantage on several tasks. The detailed results therefore indicate that the overall gain of SOAR comes mainly from broader improvements on non-saturated tasks.

Table 6: Inference latency comparison on the Adroit and Meta-World benchmarks. Inference latency measures the wall-clock time of one complete policy evaluation that generates an action chunk. Results are reported as mean latency (ms) ± standard deviation over three random seeds. For FlowPolicy, three repeated measurements are averaged for each seed. **Bold** denotes the best result.

			Adroit			Meta-World				
Methods	**Publication**	**NFE**	**Hammer**	**Door**	**Pen**	**Easy (28)**	**Medium (11)**	**Hard (6)**	**Very Hard (5)**	**Inference Latency (ms)**
DP (Chi et al., 2023)	RSS, 2023	100	671.6 ± 8.7	693.9 ± 16.9	676.4 ± 2.7	676.5 ± 6.8	672.7 ± 5.8	681.0 ± 9.2	696.7 ± 23.4	678.4 ± 6.9
DP3 (Ze et al., 2024)	RSS, 2024	10	69.0 ± 0.9	70.4 ± 1.8	80.6 ± 3.4	69.3 ± 1.7	69.9 ± 1.2	72.7 ± 2.3	76.5 ± 1.6	70.7 ± 2.6
Simple DP3 (Ze et al., 2024)	RSS, 2024	10	43.7 ± 0.3	43.3 ± 0.1	43.6 ± 0.4	43.3 ± 0.4	43.4 ± 0.4	43.3 ± 0.4	43.4 ± 0.3	43.4 ± 0.4
FlowPolicy (Zhang et al., 2025b)	AAAI, 2025	1	14.6 ± 0.1	14.7 ± 0.1	14.6 ± 0.0	14.7 ± 0.0	14.8 ± 0.0	14.7 ± 0.0	14.7 ± 0.1	14.7 ± 0.0
MP1 (Sheng et al., 2026)	AAAI, 2026	1	$\mathbf{8.2 \pm 0.0}$	$\mathbf{8.2 \pm 0.1}$	$\mathbf{8.2 \pm 0.0}$	$\mathbf{8.1 \pm 0.1}$	$\mathbf{8.1 \pm 0.1}$	$\mathbf{8.0 \pm 0.1}$	$\mathbf{8.1 \pm 0.1}$	$\mathbf{8.1 \pm 0.0}$
SOAR (Ours)	–	**1**	8.7 ± 0.3	8.6 ± 0.6	8.6 ± 0.4	8.7 ± 0.2	8.6 ± 0.0	8.6 ± 0.0	8.9 ± 0.1	8.7 ± 0.1

A.3 Additional VLA Experimental Details

This section provides additional implementation, training, and evaluation details for the VLA experiments in Section 4.2. In both benchmarks, SOAR replaces the iterative action expert of $\pi_{0.5}$ while retaining the original vision-language backbone and data-processing pipeline.

A.3.1 LIBERO

We evaluate $\pi_{0.5}$-SOAR on all four standard LIBERO suites (Liu et al., 2023a): LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. Each suite contains 10 tasks, resulting in 40 tasks in total. A single policy is jointly trained across all tasks. The vision-language backbone is initialized from the OpenPI `pi05_base` checkpoint. The original iterative action expert is replaced by SOAR, and the newly introduced SOAR parameters are initialized from scratch. We fine-tune the model for 30K optimization steps using AdamW with a global batch size of 256 on four NVIDIA A100-80GB GPUs. SOAR predicts a 10-step action chunk at each policy query. During evaluation, the first five predicted actions are executed before the policy receives a new observation and replans. Unlike the original $\pi_{0.5}$ action expert, which uses multiple network evaluations for action generation, SOAR generates the complete action chunk with a single action-expert evaluation ($\mathrm{NFE} = 1$). Each task is evaluated for 50 episodes.

A.3.2 RoboTwin 2.0

We further evaluate $\pi_{0.5}$-SOAR on RoboTwin 2.0 (Chen et al., 2025). We use all 50 tasks under the demo_clean setting and jointly train a single multi-task policy on the unified LeRobot dataset. The training set contains 27,500 trajectories and 6,075,103 frames. Each observation consists of three RGB views, including one external camera and two wrist cameras, together with a 14-dimensional proprioceptive state. All RGB observations are resized to 224×224 before being passed to the vision-language backbone. The vision-language backbone is initialized from the OpenPI pi05_base checkpoint. The original iterative flow-based action expert is replaced by SOAR, while the newly introduced SOAR parameters are initialized from scratch. The full model is fine-tuned for 50K optimization steps using AdamW with a global batch size of 16 on four NVIDIA A100-80GB GPUs with FSDP. We use a cosine learning-rate schedule with 1K warmup steps, a peak learning rate of 2.5×10^{-5}, and a final learning rate of 2.5×10^{-6}. Weight decay is set to 0.01, the gradient norm is clipped to 1.0, and an exponential moving average with decay 0.999 is maintained during training.

The policy predicts 50-step chunks of 14-dimensional absolute joint-position actions. To match the action interface of $\pi_{0.5}$, the physical state and action vectors are zero-padded to 32 dimensions before being processed by the action expert and truncated back to the original 14 dimensions before execution. At each policy invocation, SOAR generates the complete 50-step action chunk using a single action-expert evaluation ($\mathrm{NFE} = 1$). All 50 predicted actions are executed open-loop before the next policy query. We evaluate the final checkpoint on all 50 tasks using 100 episodes per task and report the average task success rate.

A.4 Real-World Experimental Details

Dataset and Task. The Tidy-up dataset contains 512 real-robot demonstrations collected at 30 Hz, with 461 trajectories used for training and 51 held out for model selection and offline evaluation. Each observation consists of three RGB views, the 7-DoF robot joint state, and the binary gripper state. Actions are represented by absolute joint targets together with a binary gripper command.

The demonstrations contain multi-stage object-sorting sequences. The robot identifies and manipulates multiple tabletop objects, discarding trash into a trash bin and placing the remaining objects into a storage basket. Each trajectory may therefore contain several consecutive grasping, transport, and placement operations.

Policy and Training. Both $\pi_{0.5}$ and $\pi_{0.5}$-SOAR use the same observation and action representations and predict 16-step action chunks. For $\pi_{0.5}$-SOAR, we initialize from a converged task-specific $\pi_{0.5}$ checkpoint, freeze the PaliGemma vision-language backbone, and train the SOAR action module. Training is performed for 30K optimization steps using AdamW with an effective batch size of 128 on two 80 GB GPUs.

Frequency-based conditioning is applied to the continuous joint-action channels, while the binary gripper channel is supervised separately. Temporal regularization is applied to the joint trajectory. Because gripper transitions are sparse, event-centered sampling increases the proportion of training samples containing gripper opening or closing events.

Temporal Conditioning. We use RTC during continuous execution with a maximum prefix delay of eight actions. Previously committed actions are provided as a temporal prefix when generating the next action chunk. SOAR generates the resulting action chunk with a single network evaluation ($\mathrm{NFE} = 1$).

Tidy-up Evaluation Protocol. Both policies are evaluated on the same 25 tabletop configurations with varied object positions and arrangements. An episode is considered successful only when all objects are correctly sorted into their designated receptacles. The full-task success rate is computed as

$$\text{Success Rate} = \frac{N_{\text{success}}}{25} \times 100\%. \tag{31}$$

This criterion evaluates completion of the entire sorting sequence rather than individual grasp or placement attempts.

Dynamic Perturbation Protocol. We additionally evaluate dynamic pick-and-place under repeated target displacements. The robot is required to grasp a toy placed on the tabletop and place it into a fixed basket. During the grasping stage, the toy is moved according to a predefined perturbation schedule containing up to four displacements.

We construct 10 perturbation configurations. For each configuration, the perturbation times and target locations are fixed before evaluation and are identical for both policies. All target positions lie within the reachable workspace, and the object orientation is kept approximately unchanged across displacements. The evaluation order of the two policies is interleaved to reduce the effect of changes in the physical setup over time.

If the robot grasps the object before all four perturbations have been applied, no further displacement is introduced and the robot proceeds with the placement stage. The episode is still included in the full-task success rate, while the actual number of perturbations experienced is recorded. Once the object has been successfully grasped, no further perturbation is applied.

The same task time limit is used for both policies. An episode is considered successful if the robot grasps the object, transports it to the basket, releases it, and the object remains stably inside the basket within the time limit. Timeouts, failure to place the object in the basket, and safety termination are counted as failures.

Perturbation Metrics. For the dynamic perturbation task, we report task success rate and empty-grasp count. The success rate is

$$\text{Success Rate}_{\text{dyn}} = \frac{N_{\text{success}}}{10} \times 100\%. \tag{32}$$

An empty grasp is counted when the gripper completes a closing attempt without securing the target object. A continuously closed gripper is counted as a single attempt. Dropping an object after a successful grasp is recorded separately and is not counted as an empty grasp. The average empty-grasp count is

$$N_{\text{empty}} = \frac{1}{N_{\text{trial}}} \sum_{i=1}^{N_{\text{trial}}} n_{\text{empty}}^{(i)}. \tag{33}$$

Failure cases are categorized as failure to grasp, object drop during transport, placement outside the basket, timeout, or safety termination.

Inference Latency. Inference latency is measured at the action-chunk level. For both policies, one measurement corresponds to the wall-clock time required to produce a complete 16-step action chunk from the current observation:

$$T_{\text{chunk}} = t_{\text{output}} - t_{\text{input}}. \tag{34}$$

Both methods use the same action horizon and deployment setting. We report latency in milliseconds per chunk rather than dividing by the number of actions, since all 16 actions are generated jointly within one policy inference.

Table 7: Detailed task-wise performance on the 53-task benchmark. Results are reported as mean success rate (%) ± sample standard deviation over three random seeds. **Bold** denotes the best result in each column.

			Adroit			Meta-World (Easy)		
Methods	**Publication**	**NFE**	**Hammer**	**Door**	**Pen**	**Button Press**	**Coffee Button**	**Plate Slide Back Side**
DP	RSS, 2023	100	75.1 ± 3.5	23.7 ± 3.7	19.3 ± 1.8	79.7 ± 3.8	83.0 ± 3.5	99.3 ± 0.6
DP3	RSS, 2024	10	$\mathbf{100.0 \pm 0.0}$	73.5 ± 2.1	56.7 ± 2.6	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
Simple DP3	RSS, 2024	10	$\mathbf{100.0 \pm 0.0}$	56.8 ± 1.2	56.1 ± 0.3	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
FlowPolicy	AAAI, 2025	1	95.2 ± 2.4	57.3 ± 6.2	47.8 ± 11.0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
MP1	AAAI, 2026	1	96.1 ± 2.3	69.2 ± 3.1	57.1 ± 6.1	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
SOAR (Ours)	-	1	$\mathbf{100.0 \pm 0.0}$	$\mathbf{77.5 \pm 2.4}$	$\mathbf{61.9 \pm 7.3}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$

	Meta-World (Easy)					
Methods	**Button Press Topdown**	**Button Press Topdown Wall**	**Button Press Wall**	**Peg Unplug Side**	**Door Close**	**Door Lock**
DP	42.7 ± 7.6	39.3 ± 5.7	87.7 ± 5.7	32.0 ± 12.3	$\mathbf{100.0 \pm 0.0}$	66.3 ± 5.9
DP3	96.7 ± 5.8	88.0 ± 6.2	$\mathbf{100.0 \pm 0.0}$	71.3 ± 4.7	$\mathbf{100.0 \pm 0.0}$	73.7 ± 12.1
Simple DP3	96.0 ± 6.1	87.3 ± 7.8	$\mathbf{100.0 \pm 0.0}$	65.3 ± 14.2	$\mathbf{100.0 \pm 0.0}$	74.3 ± 21.1
FlowPolicy	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{85.0 \pm 2.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
MP1	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.7 ± 0.6	67.0 ± 4.6	$\mathbf{100.0 \pm 0.0}$	97.3 ± 2.9
SOAR (Ours)	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	83.7 ± 6.8	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$

	Meta-World (Easy)							
Methods	**Door Open**	**Door Unlock**	**Drawer Close**	**Drawer Open**	**Faucet Close**	**Faucet Open**	**Handle Press**	**Handle Pull**
DP	93.3 ± 2.5	98.7 ± 1.2	97.7 ± 2.1	55.7 ± 9.5	98.3 ± 1.5	89.0 ± 3.0	77.7 ± 2.5	9.0 ± 7.0
DP3	81.7 ± 3.5	99.3 ± 1.2	$\mathbf{100.0 \pm 0.0}$	97.0 ± 1.7	78.3 ± 11.4	$\mathbf{100.0 \pm 0.0}$	84.0 ± 1.0	21.3 ± 3.2
Simple DP3	86.7 ± 6.4	99.3 ± 1.2	$\mathbf{100.0 \pm 0.0}$	96.3 ± 4.0	90.7 ± 8.1	$\mathbf{100.0 \pm 0.0}$	84.0 ± 6.6	11.0 ± 5.3
FlowPolicy	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	98.3 ± 2.9	13.7 ± 3.2
MP1	96.7 ± 0.6	98.7 ± 2.3	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	99.7 ± 0.6	41.3 ± 12.4
SOAR (Ours)	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{67.7 \pm 15.0}$

	Meta-World (Easy)							
Methods	**Handle Press Side**	**Handle Pull Side**	**Lever Pull**	**Plate Slide**	**Plate Slide Back**	**Dial Turn**	**Reach**	**Reach Wall**
DP	60.3 ± 6.1	6.3 ± 1.2	44.0 ± 8.7	91.0 ± 1.0	$\mathbf{100.0 \pm 0.0}$	23.3 ± 3.8	37.7 ± 5.7	31.0 ± 10.5
DP3	48.3 ± 10.8	50.3 ± 9.9	77.3 ± 5.5	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	55.3 ± 5.5	44.0 ± 9.5	67.3 ± 3.1
Simple DP3	45.0 ± 7.0	35.7 ± 11.0	72.3 ± 12.9	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	56.0 ± 2.0	41.3 ± 11.0	59.7 ± 8.1
FlowPolicy	$\mathbf{100.0 \pm 0.0}$	54.0 ± 5.6	79.3 ± 3.5	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	85.3 ± 9.1	$\mathbf{45.3 \pm 5.1}$	$\mathbf{76.3 \pm 3.1}$
MP1	$\mathbf{100.0 \pm 0.0}$	56.7 ± 7.1	79.7 ± 2.1	$\mathbf{100.0 \pm 0.0}$	97.7 ± 1.5	70.0 ± 8.2	14.3 ± 2.1	62.0 ± 7.0
SOAR (Ours)	$\mathbf{100.0 \pm 0.0}$	$\mathbf{84.7 \pm 8.5}$	$\mathbf{96.0 \pm 1.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{88.3 \pm 1.5}$	27.7 ± 4.0	74.7 ± 2.5

	Meta-World (Easy)			Meta-World (Medium)				
Methods	**Plate Slide Side**	**Window Close**	**Window Open**	**Basketball**	**Bin Picking**	**Box Close**	**Coffee Pull**	**Coffee Push**
DP	98.0 ± 2.0	99.0 ± 1.7	92.0 ± 5.3	80.7 ± 2.5	20.3 ± 0.6	36.7 ± 5.7	22.0 ± 5.3	54.3 ± 2.1
DP3	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	94.0 ± 4.4	84.0 ± 5.0	23.3 ± 5.1	50.3 ± 11.0	58.0 ± 25.1	$\mathbf{74.0 \pm 5.6}$
Simple DP3	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	93.7 ± 5.7	66.3 ± 4.9	21.3 ± 6.1	46.3 ± 8.1	53.0 ± 18.1	69.3 ± 3.2
FlowPolicy	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	97.7 ± 2.1	22.7 ± 4.7	49.7 ± 9.5	67.7 ± 4.0	64.0 ± 6.6
MP1	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	98.7 ± 2.3	80.0 ± 4.4	15.3 ± 9.1	49.3 ± 11.2	82.0 ± 6.0	59.3 ± 3.1
SOAR (Ours)	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{34.0 \pm 7.5}$	$\mathbf{65.7 \pm 5.5}$	$\mathbf{96.0 \pm 2.6}$	70.7 ± 3.2

	Meta-World (Medium)						Meta-World (Hard)		
Methods	**Hammer**	**Peg Insert Side**	**Push Wall**	**Soccer**	**Sweep**	**Sweep Into**	**Assembly**	**Hand Insert**	**Pick Out Of Hole**
DP	36.7 ± 15.3	31.0 ± 13.7	25.0 ± 15.6	8.0 ± 4.6	32.7 ± 10.6	13.0 ± 3.0	71.3 ± 7.0	10.3 ± 3.1	27.3 ± 8.6
DP3	93.7 ± 9.3	62.7 ± 4.5	74.3 ± 6.7	11.7 ± 5.0	98.7 ± 1.5	14.7 ± 4.2	94.0 ± 2.0	9.7 ± 2.3	15.7 ± 4.5
Simple DP3	89.3 ± 9.7	61.7 ± 11.2	58.3 ± 11.0	14.3 ± 4.9	91.0 ± 7.9	20.3 ± 4.5	93.7 ± 3.8	8.3 ± 0.6	14.7 ± 12.9
FlowPolicy	78.7 ± 6.0	63.7 ± 4.6	36.3 ± 7.0	14.7 ± 6.4	99.7 ± 0.6	21.0 ± 7.5	96.7 ± 5.8	9.7 ± 3.5	26.3 ± 14.6
MP1	98.3 ± 1.2	75.7 ± 3.5	72.7 ± 5.0	13.0 ± 5.6	93.7 ± 2.5	15.0 ± 1.7	98.0 ± 2.6	9.0 ± 2.6	11.7 ± 6.8
SOAR (Ours)	$\mathbf{100.0 \pm 0.0}$	$\mathbf{88.7 \pm 4.0}$	$\mathbf{99.7 \pm 0.6}$	$\mathbf{22.7 \pm 6.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{39.7 \pm 12.9}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{13.0 \pm 4.0}$	$\mathbf{42.3 \pm 1.5}$

	Meta-World (Hard)			Meta-World (Very Hard)					Overall
Methods	**Pick Place**	**Push**	**Push Back**	**Shelf Place**	**Disassemble**	**Stick Pull**	**Stick Push**	**Pick Place Wall**	**Average**
DP	17.7 ± 8.1	24.3 ± 8.5	54.7 ± 13.1	4.3 ± 0.6	62.3 ± 3.5	21.0 ± 4.4	53.7 ± 10.8	16.3 ± 12.9	52.3 ± 0.9
DP3	58.3 ± 3.5	72.0 ± 13.5	74.7 ± 12.7	15.7 ± 4.7	63.7 ± 6.1	45.0 ± 8.2	96.0 ± 4.4	83.3 ± 3.1	72.3 ± 1.4
Simple DP3	60.0 ± 3.6	73.3 ± 15.3	75.7 ± 7.8	23.0 ± 7.2	59.0 ± 4.4	49.3 ± 14.6	96.7 ± 4.2	80.7 ± 5.5	70.4 ± 1.5
FlowPolicy	50.7 ± 12.9	58.0 ± 25.6	$\mathbf{85.0 \pm 13.2}$	17.3 ± 3.8	$\mathbf{87.7 \pm 8.5}$	45.3 ± 11.5	98.7 ± 2.3	64.3 ± 19.3	75.3 ± 1.8
MP1	60.7 ± 8.4	74.7 ± 11.5	82.7 ± 1.2	24.0 ± 11.4	62.7 ± 5.0	68.7 ± 1.5	95.7 ± 7.5	77.3 ± 3.1	75.9 ± 1.3
SOAR (Ours)	$\mathbf{64.3 \pm 8.1}$	$\mathbf{76.7 \pm 4.7}$	84.7 ± 4.7	$\mathbf{43.3 \pm 13.3}$	79.7 ± 6.5	$\mathbf{73.0 \pm 2.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{88.7 \pm 3.1}$	$\mathbf{83.9 \pm 0.3}$