

π_{RL} : ONLINE RL FINE-TUNING FOR FLOW-BASED VISION-LANGUAGE-ACTION MODELS

Kang Chen^{♣,♡,*}, Zhihao Liu^{◇,♡,*}, Tonghe Zhang^{♣,*}, Zhen Guo[◇], Si Xu[◇], Hao Lin[◇], Hongzhi Zang[♠],

Quanlu Zhang[◇], Zhaofei Yu[♣], Guoliang Fan[◇], Tiejun Huang[♣], Yu Wang[♣], Chao Yu^{♠,♡,†}

* Equal Contributions † Corresponding Author: Chao Yu zoeyuchao@gmail.com

♠ Tsinghua University ♣ Peking University ◇ Institute of Automation, Chinese Academy of Sciences

♠ Carnegie Mellon University ◇ Infinigence AI ♡ Zhongguancun Academy

<https://github.com/RLinf/RLinf> <https://huggingface.co/RLinf>

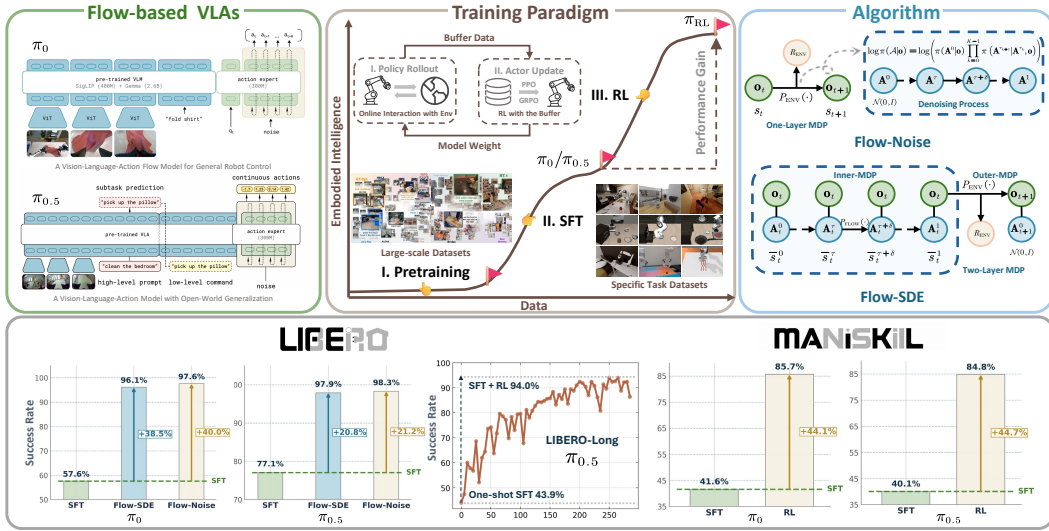


Figure 1: Overview of π_{RL} . π_{RL} , an RL framework featuring Flow-Noise and Flow-SDE two approaches, is designed to enhance the performance and generalization of SFT-aligned flow-based VLAs, represented by the π_0 and $\pi_{0.5}$. Results in LIBERO demonstrate that π_{RL} achieves significant gains over few-shot SFT, specifically boosting the $\pi_{0.5}$ one-shot SFT performance on the LIBERO-Long benchmark from 43.9% to 94.0%. Results in ManiSkill further show that π_{RL} can support large-scale multitask RL via heterogeneous parallel simulation, which includes a total of 4,352 task combinations.

ABSTRACT

Vision-Language-Action (VLA) models enable robots to understand and perform complex tasks from multimodal input. Although recent work explores using reinforcement learning (RL) to automate the laborious data collection process in scaling supervised fine-tuning (SFT), applying large-scale RL to flow-based VLAs (*e.g.*, π_0 , $\pi_{0.5}$) remains challenging due to intractable action log-likelihoods from iterative denoising.

We address this challenge with π_{RL} , an open-source framework for training flow-based VLAs in parallel simulation. π_{RL} implements two RL algorithms: (1) **Flow-Noise** models the denoising process as a discrete-time MDP with a learnable noise network for exact log-likelihood computation. (2) **Flow-SDE** integrates denoising with agent-environment interaction, formulating a two-layer MDP that employs ODE-to-SDE conversion for efficient RL exploration.

We evaluate π_{RL} on LIBERO and ManiSkill benchmarks. On LIBERO, π_{RL} boosts few-shot SFT models π_0 and $\pi_{0.5}$ from 57.6% to 97.6% and from 77.1% to 98.3%,

respectively. In ManiSkill, we train π_{RL} in 320 parallel environments, improving π_0 from 41.6% to 85.7% and $\pi_{0.5}$ from 40.0% to 84.8% across 4352 pick-and-place tasks, demonstrating scalable multitask RL under heterogeneous simulation. Overall, π_{RL} achieves significant performance gains and stronger generalization over SFT-models, validating the effectiveness of online RL for flow-based VLAs.

1 INTRODUCTION

Vision-Language-Action (VLA) models (Din et al., 2025) have emerged as a leading solution for general-purpose robots, effectively bridging the gap between high-level multimodal reasoning and low-level physical control (Firoozi et al., 2025). Conditioned on sensor inputs and language commands, VLAs (Team et al., 2024; Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025) can translate abstract instructions into executable robotic actions, thereby enabling intuitive and flexible human-robot interaction.

The training methodology for VLAs follows the standard pre-training and supervised fine-tuning (SFT) paradigm as shown in Fig. 1. Building on the pretrained Vision-Language Model (VLM) (Touvron et al., 2023; Beyer et al., 2024), VLAs are fine-tuned on large-scale, heterogeneous human demonstration datasets (O’Neill et al., 2024; Khazatsky et al., 2024), followed by SFT on the target task to align their capabilities with the specific embodiment and environment. However, reliance on SFT introduces a critical challenge: curating large-scale, high-quality expert trajectories is both laborious and costly (Din et al., 2025), and the models obtained via SFT tend to overfit to expert demonstrations (Fei et al., 2025).

Recent efforts (Zang et al., 2025; Li et al., 2025a; Tan et al., 2025; Liu et al., 2025b) have explored expanding the VLA training process with reinforcement learning (RL), establishing a pre-training, SFT, and RL paradigm as shown in Fig. 1, allowing VLAs to improve their performance beyond initial expert demonstrations through active environmental interaction and the development of more generalizable policies.

However, these RL advances have been largely confined to autoregressive VLAs, featuring OpenVLA (Kim et al., 2024) and OpenVLA-OFT (Kim et al., 2025), which employ discrete action decoders that generate output in an autoregressive or parallel fashion. This stands in stark contrast to diffusion- or flow-based VLAs, exemplified by the π series models π_0 (Black et al., 2024) and $\pi_{0.5}$ (Intelligence et al., 2025), which generate actions through iterative refinement in flow matching (Lipman et al., 2022), offering the advantages of generating action chunks in high-frequency and performing highly dexterous tasks (Black et al., 2024). Consequently, previous VLA-RL algorithms are incompatible with flow-based VLAs, and the fundamental challenge lies in how to characterize a logarithmic likelihood (Hutchinson, 1989; Chen et al., 2018) for the executed actions.

Contribution: We introduce π_{RL} , the first open-source framework for fine-tuning flow-based VLAs π_0 and $\pi_{0.5}$ with parallel online RL.

To address the intractable log-likelihood estimation problem in flow matching, we propose two solutions: Flow-Noise and Flow-SDE. Flow-Noise integrates a learnable noise network into the denoising process and models this stage as a discrete-time Markov decision process (MDP) for exact log-likelihood estimation. Flow-SDE converts the ordinary differential equation (ODE) denoising process into a stochastic differential equation (SDE) while maintaining equivalent marginal distributions for exploration, and builds a two-layer MDP that couples the denoising process with policy-environment interaction, along with a hybrid ODE-SDE sampling technique for training acceleration. Given the formulated MDP and the exact log-likelihood computation, π_{RL} undergoes further optimization via the proximal policy optimization (PPO) (Schulman et al., 2017) algorithm.

We conduct extensive experiments on the challenging multi-task benchmarks LIBERO (Liu et al., 2023) and high-fidelity simulator ManiSkill (Tao et al., 2024) to evaluate the effectiveness of π_{RL} optimization on the π_0 and $\pi_{0.5}$ models, with comprehensive findings summarized in Fig. 1.

Results on LIBERO. π_{RL} demonstrates substantial performance gains over the SFT baselines, with the average success rate of π_0 improving from 57.6% to 97.6%, and $\pi_{0.5}$ from 77.1% to 98.3%. Notably, on the LIBERO-Long task suite, π_{RL} boosts the performance of the $\pi_{0.5}$ one-trajectory SFT model from 43.9% to 94.0%, surpassing the 92.4% performance of the all-trajectories SFT model. Moreover, we compare against group relative policy optimization (GRPO) (Shao et al., 2024) as an alternative policy gradient algorithm, with the comparison showing that PPO consistently outperforms GRPO across all task suites.

Results in ManiSkill. We train the policy to pick 16 types of objects and place them on 17 different receptacles in 16 photorealistic scenes, with a total of 4,352 combinations. π_{RL} boosts the average success rate from 41.6% to 85.7% for π_0 and 41.9% to 84.8% for $\pi_{0.5}$, demonstrating π_{RL} ’s ability to support large-scale multi-task RL. Additionally, we also conduct experiments on the SIMPLER benchmark (Li et al., 2024), the success rate was elevated from 67.2% to 86.7% for π_0 and from 59.24% to 79.1% for $\pi_{0.5}$.

To sum up, our contributions are:

- **RL for flow-based VLAs.** We introduce π_{RL} , the first online RL fine-tuning framework for flow-based π -series VLAs, featuring Flow-Noise and Flow-SDE, two distinct technical solutions that allow exact log-likelihood estimation in flow matching.
- **Superior Performance.** We demonstrate significant performance improvements and enhanced generalization of π_{RL} on the multi-task benchmarks LIBERO and ManiSkill.
- **Comprehensive Ablation.** We conduct thorough ablation studies on RL algorithms, critic designs, noise injection strategies, MDP formulations, and hyperparameters within flow-based VLAs, providing empirical insights for future research on RL for flow-based VLAs.
- **Open-source Code and Models.** We release all codes and model checkpoints to ensure reproducibility, hoping that our study helps to advance further research in this field.

2 RELATED WORK

2.1 VISION-LANGUAGE-ACTION MODELS

VLA models have recently achieved remarkable progress in robotics by integrating multimodal inputs to enable unified perception, reasoning, and control. This development has led to a series of architectures, including Octo (Team et al., 2024), RT (Brohan et al., 2022), OpenVLA, OpenVLA-OFT, π_0 , $\pi_{0.5}$, and GR00T (Bjorck et al., 2025). OpenVLA, which exemplifies the autoregressive VLA architecture, discretizes the action space into tokenized representations. This enables language-conditioned control by treating actions as part of the VLM’s vocabulary, but it inherently limits the resolution required for fine-grained motion. To achieve more dexterous and continuous physical behaviors, π_0 and $\pi_{0.5}$, as representatives of flow-based VLA architectures, introduce an action chunking architecture based on flow matching. This allows VLAs to model complex continuous action distributions, thereby achieving more dexterous physical behaviors.

In this work, we further fine-tune the π -series models with online RL algorithms, enhancing their performance and generalization capabilities through online interaction with the environment.

2.2 ONLINE RL FINE-TUNING FOR VLA MODELS

Recent research has increasingly focused on enhancing the performance and generalization of VLAs with online RL. For example, SimpleVLA-RL (Li et al., 2025a), building on the OpenVLA-OFT and GRPO, demonstrated that RL can improve long-horizon planning of VLA models under data scarcity. RL4VLA (Liu et al., 2025b) empirically evaluated PPO, GRPO, and direct preference optimization (DPO) (Rafailov et al., 2023) with stage-based sparse rewards, finding PPO to yield superior performance. VLA-RL (Lu et al., 2025) proposed a specialized robotic process reward model and enhanced the data processing pipeline. iRe-VLA (Guo et al., 2025b) proposed a framework that iterates between RL exploration and SFT updates. RIPT-VLA (Tan et al., 2025) applied the REINFORCE leave-one-out (RLOO) (Kool et al., 2018) algorithm to the QueST (Metz et al., 2024) and OpenVLA-OFT architectures. RLinf-VLA (Yu et al., 2025; Zang et al., 2025) provides a unified and efficient framework for scalable RL training of VLA models, supporting diverse VLA

architectures such as OpenVLA and OpenVLA-OFT, multiple RL algorithms like PPO and GRPO, and various simulators including LIBERO and ManiSkill. These works demonstrate the effectiveness of RL fine-tuning VLA models.

While these approaches demonstrate the potential of applying online RL to VLAs, their application to flow-based VLAs is hindered by the challenge of exact log-likelihood estimation.

2.3 RL FINE-TUNING FOR FLOW MODELS

Integrating RL with flow models is a promising way to transcend the limitations of imitation learning. To this end, Flow-GRPO (Liu et al., 2025a) converts the deterministic ODE into an equivalent SDE to enable stochasticity exploration, a foundation upon which subsequent works like Mix-GRPO (Li et al., 2025b) and TempFlow-GRPO (He et al., 2025) further accelerate training through hybrid ODE-SDE rollouts. ReinFlow (Zhang et al., 2025) injects learnable noise into the flow path and transforms it into a discrete-time Markov process with a tractable likelihood for stable policy gradient updates. Flow policy optimization (FPO) (McAllister et al., 2025) reframes policy optimization as maximizing the advantage-weighted ratio of the conditional flow matching loss. Policy-Agnostic RL (PA-RL) Mark et al. (2024) effectively fine-tunes diverse diffusion and Transformer architectures by distilling critic-optimized actions into the policy via supervised learning. Diffusion steering via reinforcement learning (DSRL) (Wagenmaker et al., 2025) refines the flow policy by performing RL in its latent-noise space, rather than modifying the policy parameters themselves.

While prior work has mostly focused on non-robotic tasks or small-scale, single-task robotics, we address the more challenging problem of fine-tuning large-scale flow-based VLAs for complex, multi-task robotic scenarios.

3 PRELIMINARY

3.1 PROBLEM FORMULATION

We formulate the task as an MDP, defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P_0, P_{\text{ENV}}, R_{\text{ENV}}, \gamma)$. The state $s_t \in \mathcal{S}$ is defined as the robot observation \mathbf{o}_t and P_0 denotes the initial state distribution. Given the state, the flow policy predicts an action $a_t \sim \pi(\cdot|s_t) \in \mathcal{A}$, resulting in the state transition $s_{t+1} \sim P_{\text{ENV}}(\cdot|s_t, a_t)$ and a reward $R_{\text{ENV}}(s_t, a_t)$. The objective is to learn a policy π_θ that maximizes the expected γ -discounted return over a horizon of $T + 1$:

$$\mathcal{J}(\pi_\theta) = \mathbb{E}_{\pi_\theta, P_0} \left[\sum_{t=0}^T \gamma^t R_{\text{ENV}}(s_t, a_t) \right]. \quad (1)$$

With the policy gradient surrogate (Williams, 1992), the gradient of the return expectation can be approximated from sampled trajectories:

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{\pi_\theta, P_0} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t) \right]. \quad (2)$$

The advantage function, $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$, measures the relative merit of the action value $Q(s_t, a_t)$ over the state value $V(s_t)$, providing a low-variance signal for the policy update.

3.2 FLOW-BASED VISION-LANGUAGE-ACTION MODEL

A flow-based VLA model π_θ is designed to map the observation \mathbf{o}_t comprising RGB images, language tokens, and robot proprioception to a sequence of H future actions $\mathbf{A}_t = [a_{t,0}, \dots, a_{t,H-1}]$, formulated as $p(\mathbf{A}_t|\mathbf{o}_t)$. Within the model, the VLM extracts features from the visual and language inputs, while the flow matching expert is tasked with generating the actions. Specifically, the model learns a conditional vector field \mathbf{v}_θ that transforms a standard Gaussian noise distribution into the target action \mathbf{A}_t . This is achieved by minimizing the Conditional Flow Matching (CFM) loss, which aligns the predicted vector field \mathbf{v}_θ with the ground-truth vector field \mathbf{u} :

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{\tau, p(\mathbf{A}_t, \mathbf{o}_t), q(\mathbf{A}_t^\tau|\mathbf{A}_t)} \left[\|\mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t) - \mathbf{u}(\mathbf{A}_t^\tau|\mathbf{A}_t)\|_2^2 \right]. \quad (3)$$

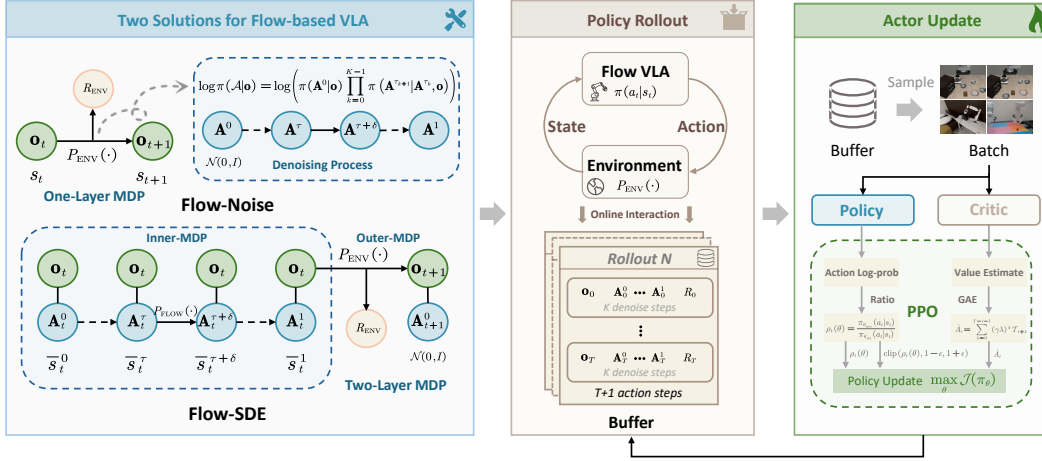


Figure 2: Two optimization methods in π_{RL} . Flow-Noise adds learnable noise in a one-layer MDP (Fig. 3), using the denoised joint likelihood for policy gradient. Flow-SDE builds a two-layer MDP with ODE-to-SDE conversion, and computes the likelihood directly.

Here, the conditional probability path $q(\mathbf{A}_t^\tau | \mathbf{A}_t)$ generates a noisy action¹ $\mathbf{A}_t^\tau = \tau \mathbf{A}_t + (1 - \tau)\epsilon$ from an action \mathbf{A}_t , random noise $\epsilon \sim \mathcal{N}(0, I)$, and a continuous time $\tau \in [0, 1]$ in flow matching. For this specific path, the corresponding ground-truth vector field is defined as $\mathbf{u}(\mathbf{A}_t^\tau | \mathbf{A}_t) = \mathbf{A}_t - \epsilon$.

During the inference, the action sequence is generated by first sampling a noise vector $\mathbf{A}_t^0 \sim \mathcal{N}(0, I)$, which is further iteratively refined by integrating the learned vector field \mathbf{v}_θ over a fixed number of steps based on the forward Euler method: $\mathbf{A}_t^{\tau+\delta} = \mathbf{A}_t^\tau + \mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t) \cdot \delta$.

4 METHODOLOGY

Existing VLA-RL approaches leverage base models such as OpenVLA for discrete actions and OpenVLA-OFT for continuous actions. To compute the action log-likelihood $\log \pi_\theta(a_t | s_t)$, discrete models (Liu et al., 2025b) apply softmax to the output logits, while continuous models (Li et al., 2025a) treat the action as a Gaussian distribution, employing a prediction head to estimate the variance. As for the flow-based VLAs, directly computing the exact likelihood (Hutchinson, 1989) is inaccurate with few denoising steps. Moreover, the deterministic nature of its ODE sampling process precludes exploration, making its implementation within RL non-trivial. To this end, we propose Flow-Noise and Flow-SDE, two technical approaches that make flow-based VLAs amenable to RL, as depicted in Fig. 2.

4.1 FLOW-NOISE

Inspired by Reinflow (Zhang et al., 2025), we incorporate a learnable noise network into the flow matching denoising process and solve the problem within the standard one-layer MDP framework detailed in Sec. 3.1. By modeling the denoising stage as a discrete MDP, we can directly compute the log-likelihood of the denoised sequence, enabling equivalent policy optimization via RL.

4.1.1 STOCHASTICITY INJECTION

In Flow-Noise, we parameterize the noise schedule with a neural network, allowing the magnitude of the injected noise to be learned dynamically during training for greater flexibility, as shown in Fig. 3. We focus on the generation process within a single environment timestep t . For notational simplicity, we omit the time subscript t , e.g., writing \mathbf{A}^τ , and denote the predicted velocity $\mathbf{v}_\theta(\mathbf{A}^\tau, \mathbf{o})$ as \mathbf{v}^τ .

¹ \mathbf{A}_t^τ incorporates two temporal indices, t denotes the discrete time step for environment interaction and τ represents the continuous time variable in flow matching.

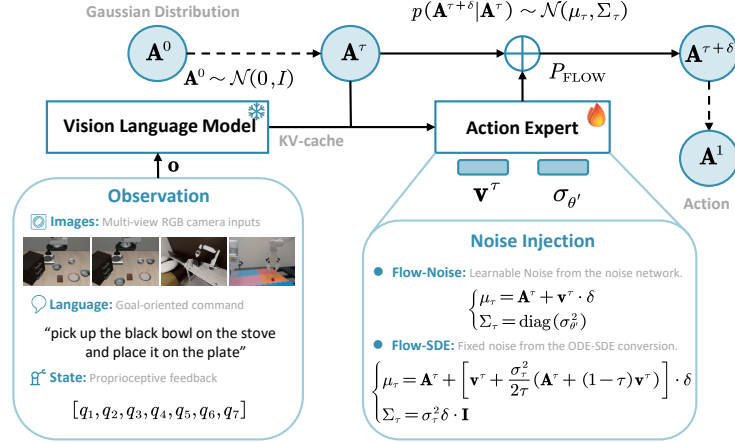


Figure 3: Illustration for the noise injection on the flow matching, exemplified by $\pi_{0.5}$, which integrates image, language, and state information for unified VLM input.

The step transition during the denoising process is modeled as an isotropic Gaussian distribution $p(\mathbf{A}^{\tau+\delta}|\mathbf{A}^\tau) \sim \mathcal{N}(\mu_\tau, \Sigma_\tau)$, where the mean is determined by the forward Euler update of the original ODE and the variance is controlled by the learnable noise network θ' :

$$\begin{cases} \mu_\tau = \mathbf{A}^\tau + \mathbf{v}^\tau \cdot \delta \\ \Sigma_\tau = \text{diag}(\sigma_{\theta'}^2) \end{cases} \quad (4)$$

Here, $\sigma_{\theta'}(\cdot)$ is the standard deviation learned from the noise injection network, conditioned on the action \mathbf{A}^τ , and the observation \mathbf{o} . The noise network is trained jointly with the velocity but discarded after fine-tuning, leaving a deterministic policy for inference.

4.1.2 LOG-LIKELIHOOD ESTIMATION

The primary challenge in applying policy gradient methods to flow-based VLAs stems from the intractable log-likelihood of the final executed action. In Flow-Noise, we address it by substituting the gradient of the joint log-likelihood of the entire denoising process into the policy optimization objective in Eq. (2), which is theoretically grounded in Reinfo (Zhang et al., 2025).

The inference process for action generation is discretized into K uniform steps, which defines a sequence of time points $\{\tau_0, \tau_1, \dots, \tau_K\}$. With the step interval defined as $\delta = 1/K$, the discrete timestep at the k -th point is $\tau_k = k \cdot \delta$, starting from $\tau_0 = 0$ and culminating at $\tau_K = 1$. Given the observation \mathbf{o} , the exact and tractable log probability for the entire denoising sequence $\mathcal{A} = (\mathbf{A}^0, \dots, \mathbf{A}^1)$ is depicted in Fig. 2 and formulated as:

$$\log \pi(\mathcal{A}|\mathbf{o}) = \log \left(\pi(\mathbf{A}^0|\mathbf{o}) \prod_{k=0}^{K-1} \pi(\mathbf{A}^{\tau_{k+1}}|\mathbf{A}^{\tau_k}, \mathbf{o}) \right). \quad (5)$$

Building on this, we can treat flow-based policy optimization within a standard MDP framework.

4.2 FLOW-SDE

Inspired by Flow-GRPO (Liu et al., 2025a), we enhance stochastic exploration by converting the denoising process from ODE into an SDE formulation. We further construct a two-layer MDP to couple the denoising process with the policy-environment interaction following DPPO (Ren et al., 2024), while leveraging the hybrid ODE-SDE sampling technique to accelerate the training process.

4.2.1 STOCHASTICITY INJECTION

In Flow-SDE, we convert the deterministic ODE into an equivalent SDE that preserves the marginal probability density of the generated actions, as shown in Fig. 3.

The deterministic ODE sampling trajectory of the flow matching, especially the Rectified Flow (Liu et al., 2022), is described by the forward Euler method:

$$d\mathbf{A}^\tau = \mathbf{v}^\tau d\tau. \quad (6)$$

Building on the connection between the *probability flow* ODE and SDE (Song et al., 2020), we can transform the deterministic ODE in Eq. (6) into an equivalent SDE, with a drift term that corrects the original velocity and a diffusion term that introduces noise:

$$d\mathbf{A}^\tau = \underbrace{\left(\mathbf{v}^\tau - \frac{1}{2}g^2(\tau)\nabla \log q_\tau(\mathbf{A}^\tau) \right)}_{\text{Drift Term}} d\tau + \underbrace{g(\tau)d\mathbf{w}}_{\text{Diffusion Term}}, \quad (7)$$

where $g(\tau)$ is a scalar function controlling the noise schedule, $\nabla \log q_\tau(\mathbf{A}^\tau)$ is the score function of the marginal distribution q_τ and $d\mathbf{w}$ denotes a Wiener process.

As established in Flow-GRPO, the score function and the velocity field are critically linked by $\nabla \log q_\tau(\mathbf{A}^\tau) = -\frac{\mathbf{A}^\tau}{\tau} - \frac{1-\tau}{\tau}\mathbf{v}^\tau$. By substituting the score function with the velocity field in Eq. (7) and setting the noise schedule $g(\tau)$ to $\sigma_\tau = a\sqrt{\frac{\tau}{1-\tau}}$ with a controlling the noise level, we derive the final SDE formulation for the flow-matching sampler:

$$d\mathbf{A}^\tau = \left[\mathbf{v}^\tau + \frac{\sigma_\tau^2}{2\tau} (\mathbf{A}^\tau + (1-\tau)\mathbf{v}^\tau) \right] d\tau + \sigma_\tau d\mathbf{w}_\tau. \quad (8)$$

Discretizing this SDE reveals that the transition probability $p(\mathbf{A}^{\tau+\delta}|\mathbf{A}^\tau) \sim \mathcal{N}(\mu_\tau, \Sigma_\tau)$ is an isotropic Gaussian distribution, with the mean and variance formulated as:

$$\begin{cases} \mu_\tau = \mathbf{A}^\tau + \left[\mathbf{v}^\tau + \frac{\sigma_\tau^2}{2\tau} (\mathbf{A}^\tau + (1-\tau)\mathbf{v}^\tau) \right] \cdot \delta \\ \Sigma_\tau = \sigma_\tau^2 \delta \cdot \mathbf{I} \end{cases}. \quad (9)$$

4.2.2 MDP FORMULATION

While Flow-Noise substitutes the joint log-likelihood of the entire denoising sequence for the likelihood of the final executed action, we couple the denoising process of the flow matching with environmental interaction in Flow-SDE. Specifically, we embed the inner MDP defined during the denoising process into the high-level, outer-loop MDP with the environment \mathcal{M}_{ENV} in Sec. 3.1, formulating a two-layer MDP as shown in Fig. 2, with components defined with respect to the environment time t and denoising time τ .

- **State** $\bar{s}_t^\tau = (\mathbf{o}_t, \mathbf{A}_t^\tau)$ is the tuple of the observation \mathbf{o}_t and the action state \mathbf{A}_t^τ .
- **Action** \bar{a}_t^τ is defined as the next sampled denoised action in the inner-loop and the executed action for the outer loop:

$$\bar{a}_t^\tau = \begin{cases} \mathbf{A}_t^{\tau+\delta} & \text{if } \tau < 1 \\ \mathbf{A}_t^1 & \text{if } \tau = 1 \end{cases}, \quad (10)$$

where $\mathbf{A}_t^{\tau+\delta} = \mu_\tau + \sigma_\tau \sqrt{\delta} \cdot \epsilon$, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is the randomly sampled noise.

- **Transition** $\bar{P}(\bar{s}_{t'}^\tau | \bar{s}_t^\tau, \bar{a}_t^\tau)$ defines how the state evolves, formulated as:

$$\bar{s}_{t'}^\tau = \begin{cases} (\mathbf{o}_t, \bar{a}_t^\tau) & \text{if } \tau < 1 \\ (\mathbf{o}_{t+1}, \mathbf{A}_{t+1}^0) & \text{if } \tau = 1 \end{cases}. \quad (11)$$

For $\tau < 1$, the inner loop transition $P_{\text{FLOW}}(\cdot)$ occurs between different denoised action states, where the observation \mathbf{o}_t remains fixed and the next action state is set by $\bar{a}_t^\tau = \mathbf{A}_t^{\tau+\delta}$. For $\tau = 1$, the final action $\bar{a}_t^\tau = \mathbf{A}_t^1$ interacts with the outer-loop environment, resulting in a new observation \mathbf{o}_{t+1} according to the environment dynamics $P_{\text{ENV}}(\cdot)$. Concurrently, the action state is reset from a standard normal distribution $\mathbf{A}_{t+1}^0 \sim \mathcal{N}(0, I)$.

- **Reward** $\bar{R}(\bar{s}_t^\tau, \bar{a}_t^\tau)$ is granted only upon completion of the denoising process and interaction with the environment:

$$\bar{R}(\bar{s}_t^\tau, \bar{a}_t^\tau) = \begin{cases} 0 & \text{if } \tau < 1 \\ R_{\text{ENV}}(\mathbf{o}_t, \mathbf{A}_t^1) & \text{if } \tau = 1 \end{cases}. \quad (12)$$

Within the two-layer MDP framework, the problem of estimating the action log-likelihood $\log \pi(a_t|s_t)$ is transformed into estimating $\log \pi(\bar{a}_t^\tau|\bar{s}_t^\tau)$, which is straightforward to compute due to the Gaussian nature of the transitions.

4.2.3 HYBRID ODE-SDE SAMPLING

In the formulated two-layer MDP framework, the effective trajectory length is the product of the environment interaction steps and the number of flow matching denoising steps. While this formulation enables RL training for flow-based VLAs, it significantly extends the MDP horizon compared to non-iterative VLA methods, which substantially increases both the training difficulty and the computational time required for optimization.

To this end, we adopt the mixed ODE-SDE rollout strategy, drawing inspiration from the text-to-image generation methods such as Mix-GRPO (Li et al., 2025b) and TempFlow-GRPO (He et al., 2025). Specifically, during the denoising process, a single step is randomly sampled as a stochastic SDE transition governed by $p(\mathbf{A}^{\tau+\delta}|\mathbf{A}^\tau) \sim \mathcal{N}(\mu_\tau, \Sigma_\tau)$, while the remaining steps follow deterministic ODE transitions defined by the update rule $\mathbf{A}^{\tau+\delta} = \mathbf{A}^\tau + \mathbf{v}^\tau \cdot \delta$.

Under this formulation, we treat the deterministic ODE transition between states as an environment-level wrapper and revise the state transition function of the previous two-layer MDP. Specifically, at each environment step t , a denoising time τ_t is randomly selected for the policy’s stochastic injection. The policy π acts on this state $\bar{s}_t^{\tau_t} = (\mathbf{o}_t, \mathbf{A}_t^{\tau_t})$, sampling the action $\mathbf{A}_t^{\tau_t+\delta}$ according to Eq. (10). The environment wrappers then execute all subsequent deterministic steps, ultimately transitioning to the next observation \mathbf{o}_{t+1} and the next action state $\bar{s}_{t+1}^{\tau_{t+1}} = (\mathbf{o}_{t+1}, \mathbf{A}_{t+1}^{\tau_{t+1}})$ at a newly sampled time τ_{t+1} . During this process, the state input and action output of the policy remain consistent with the previous two-layer MDP formulation, thus ensuring theoretical consistency.

4.3 POLICY OPTIMIZATION

4.3.1 ALGORITHM

Given the formulated flow policy MDP, our objective is to learn the optimal parameters θ^* for the policy π_θ that maximizes the expected discounted return $\mathcal{J}(\pi_\theta)$. To this end, we apply the widely adopted policy gradient algorithm PPO to optimize the policy.

π -series models (Black et al., 2024; Intelligence et al., 2025) adopt a chunk-based approach for action generation. Specifically, the policy outputs an entire sequence of H future actions $\mathbf{A}_t = [a_{t,0}, \dots, a_{t,H-1}]$ in response to each observation. In this approach, we treat the entire sequence as a single macro-step and define its corresponding reward $R_t = \sum_{j=0}^{H-1} r_{t,j}$ as the sum of the per-step rewards $r_{t,j}$, referred to as the chunk-level formulation in RLinf-VLA (Zang et al., 2025).

To effectively guide policy updates, PPO employs Generalized Advantage Estimation (GAE) (Schulman et al., 2015) to compute a low-variance estimate of the advantage, estimated as:

$$\hat{A}_t = \sum_{k=0}^{T-t} (\gamma\lambda)^k \mathcal{T}_{t+k}, \quad (13)$$

where the TD-error is $\mathcal{T}_t = R_t + \gamma V(s_{t+1}) - V(s_t)$. Here, $V(\cdot)$ is the state-value function derived from the critic network, γ is the discount factor, and λ is the parameter that balances the trade-off between bias and variance in the advantage estimate.

PPO constrains policy updates to a small trust region to prevent large, destabilizing updates, with the objective function:

$$\mathcal{J}(\pi_\theta) = \mathbb{E}_t \left[\min \left(\rho_t(\theta) \hat{A}_t, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right], \quad (14)$$

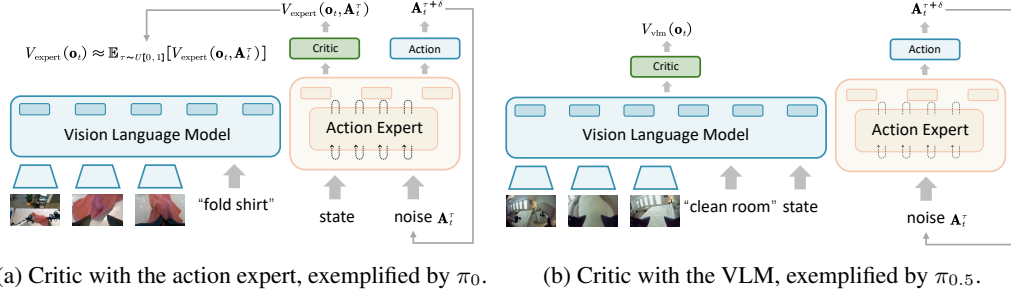


Figure 4: Illustration of the two critic placement configurations.

where the clip function, governed by a hyperparameter ϵ , restricts the ratio $\rho_t(\theta)$ to the interval $[1 - \epsilon, 1 + \epsilon]$ to ensure training stability.

Here, the probability ratio $\rho_t(\theta)$ between the updated and old policies takes the form of either:

$$\rho_t(\theta) = \frac{\pi_{\theta_{\text{new}}}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} \quad \text{or} \quad \rho_t(\theta) = \frac{\pi_{\theta_{\text{new}}}(\bar{a}_t^\tau | \bar{s}_t^\tau)}{\pi_{\theta_{\text{old}}}(\bar{a}_t^\tau | \bar{s}_t^\tau)}, \quad (15)$$

for the one-layer and two-layer MDP formulations, respectively.

4.3.2 CRITIC DESIGN

Following VLA-PPO works (Zang et al., 2025; Liu et al., 2025b), we employ a shared actor-critic architecture for memory-efficient value prediction as shown in Fig. 4. However, the two flow-based VLAs process the proprioceptive state differently: in π_0 , the state is fed into the action expert model, whereas in $\pi_{0.5}$, it is merged with prompt embeddings within the VLM.

To this end, for the $\pi_{0.5}$ variant, we attach the critic network directly to the VLM output, providing the value estimate $V_{\text{vlm}}(\mathbf{o}_t)$ conditioned on the integrated image, language, and state inputs. Conversely, for the π_0 variant, achieving the value prediction is non-trivial due to the coupled input structure, where the action expert requires both the noisy action \mathbf{A}_t^τ and the state. To this end, we approximate $V_{\text{expert}}(\mathbf{o}_t)$ by averaging the value estimates across the entire denoising trajectory, formulated as:

$$V_{\text{expert}}(\mathbf{o}_t) \approx \mathbb{E}_{\tau \sim U[0,1]} [V_{\text{expert}}(\mathbf{o}_t, \mathbf{A}_t^\tau)]. \quad (16)$$

5 EXPERIMENTAL RESULTS

5.1 SETUP

Environments. We perform experiments based on LIBERO (Liu et al., 2023) and ManiSkill (Tao et al., 2024). LIBERO is built on a CPU-based simulation platform and assesses knowledge transfer in robotic multi-task and lifelong learning across four manipulation tasks: Spatial, Object, Goal, and Long. ManiSkill serves as a high-fidelity, GPU-parallelized simulation platform. Within ManiSkill, we adopt the SIMPLER benchmark (Li et al., 2024) as our primary testbed. To further evaluate the multi-task learning capability of π_{RL} , we follow the setup of RL4VLA (Liu et al., 2025b) and construct 4,352 pick-and-place task combinations as an extended benchmark, named as *MultiTask*.

Flow-based VLAs. We conduct experiments based on π_0 and $\pi_{0.5}$. π_0 introduces the flow-matching action expert (300M) built upon a pre-trained PaliGemma (3B) to leverage broad semantic knowledge from internet-scale data. $\pi_{0.5}$ further utilizes co-training across heterogeneous data sources (e.g., multi-robot data, web data, and high-level semantic predictions) for broader generalization.

Implementation Details. Given the poor performance of pre-trained models on LIBERO and ManiSkill, we initially perform SFT with expert demonstrations using open-source code from openpi (Black et al., 2024; Intelligence et al., 2025). For the SFT stage, we fine-tune the full 3.3B model. In the subsequent RL phase, we freeze the VLM parameters and only fine-tune the 300M action expert model, driven by GPU memory efficiency and RL4VLA findings that RL contributes more significantly to action generalization. Additionally, we build our RL algorithm upon the RLinf (Yu et al.,

Table 1: Evaluation results on the LIBERO benchmark, evaluated based on the success rate (%).

Model	LIBERO					
	Spatial	Object	Goal	Long	Avg.	Δ Avg.
<i># Full Dataset SFT</i>						
Octo	78.9	85.7	84.6	51.1	75.1	—
OpenVLA	84.7	88.4	79.2	53.7	76.5	—
π_{fast}	96.4	96.8	88.6	60.2	85.5	—
OpenVLA-OFT	91.6	95.3	90.6	86.5	91.0	—
π_0	96.8	98.8	95.8	85.2	94.2	—
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9	—
<i># Few-shot SFT + RL</i>						
π_0	SFT	65.3	64.4	49.8	57.6	—
	Flow-SDE	98.4	99.4	96.2	90.2	+38.5
	Flow-Noise	99.0	99.2	98.2	93.8	+40.0
<i># Few-shot SFT + RL</i>						
$\pi_{0.5}$	SFT	84.6	95.4	84.6	43.9	—
	Flow-SDE	99.6	100	98.8	93.0	+20.8
	Flow-Noise	99.6	100	99.6	94.0	+21.2

2025), where we adopt a shared, co-located GPU allocation strategy that places the environment, rollout model, and actor model on the same GPU and executes them serially.

For the model configurations, we adhere to the official setting provided by openpi. In these settings, π_0 utilizes image, language, and proprioceptive states as input, whereas $\pi_{0.5}$ notably omits state information for the LIBERO benchmark². Following this precedent, we consistently omit the state input for $\pi_{0.5}$ during both SFT and RL phases on LIBERO and ManiSkill. Our experiments are conducted on 8 NVIDIA H100 80GB GPUs, and detailed training hyperparameters are available in Appendix Tabs. 6 and 7.

5.2 MAIN RESULTS

5.2.1 LIBERO

SFT Procedure. The LIBERO benchmark comprises four task suites, each consisting of 10 distinct subtasks. To facilitate few-shot SFT on LIBERO, a minimum of 40 expert demonstration trajectories is necessary to ensure a positive success rate for each subtask across four task suites, thereby guaranteeing a positive optimization signal for the subsequent RL phase.

For the π_0 model, we utilized a subset of 58 trajectories, sampled from the total of 1,692 trajectories spanning the four task suites in the official LIBERO SFT dataset³, to perform SFT, which served as the initial checkpoint⁴ for subsequent RL training. Additionally, a larger pool of 208 trajectories was employed for the LIBERO-Long few-shot SFT⁵ due to the long-horizon and more challenging nature of these tasks. For the $\pi_{0.5}$ model, given its stronger pretrained checkpoint, we only leveraged 40 trajectories for few-shot SFT, providing a unified checkpoint⁶ across task suites.

RL Procedure. In RL, the VLA model receives a multi-modal input state comprising: an agent-view and a wrist-view (both 224×224 RGB images), natural language guidance, the robot arm’s 6-DOF joint angles, and the gripper state. The model outputs an action to interact with the LIBERO environment, which provides a binary reward of 1 for successful task completion and 0 otherwise.

²<https://github.com/Physical-Intelligence/openpi/issues/687>

³<https://huggingface.co/datasets/physical-intelligence/libero>

⁴<https://huggingface.co/RLinf/RLinf-Pi0-SFT-Spatial-Object-Goal>

⁵<https://huggingface.co/RLinf/RLinf-Pi0-SFT-Long>

⁶<https://huggingface.co/RLinf/RLinf-Pi05-SFT>

Experiments. We benchmark the performance of π_{RL} , which fine-tunes the few-shot SFT π_0 and $\pi_{0.5}$ models with our proposed Flow-Noise and Flow-SDE, against several state-of-the-art VLAs trained on the entire LIBERO dataset, including Octo, OpenVLA, OpenVLA-OFT, π_{fast} (Pertsch et al., 2025), π_0 , and $\pi_{0.5}$. We conduct experiments on four LIBERO task suites and report performance as the success rate across all 500 initial states (10 sub-tasks \times 50 states each).

Analysis. As detailed in Tab. 1, our proposed two solutions, Flow-Noise and Flow-SDE, not only achieve comparable performance but also establish a new state-of-the-art by significantly boosting the performance of the few-shot π_0 and $\pi_{0.5}$ SFT models.

For the few-shot π_0 model, the SFT baseline performs poorly, with an average success rate of only 57.6%, indicating that the model struggles with limited demonstration data. Our proposed π_{RL} substantially boosts performance, with Flow-SDE and Flow-Noise reaching 96.1% and 97.6%, respectively, and surpassing the full-dataset π_0 SFT baseline of 94.2%.

While the $\pi_{0.5}$ few-shot SFT baseline achieves a decent average performance of 77.1%, it struggles with the challenging LIBERO-Long task, scoring only 43.9%. Our proposed π_{RL} framework rectifies this deficiency, boosting the LIBERO-Long success rate from 43.9% to 94.0%, constituting a 50.1% improvement. Notably, despite using only a single trajectory for SFT, π_{RL} reaches 98.3% final performance, surpassing the 96.9% full-dataset SFT model.

Discussion on two methods. Flow-SDE and Flow-Noise differ primarily in their noise injection strategy and MDP formulation, with experiments indicating that Flow-Noise marginally outperforms Flow-SDE, a result we attribute to two factors:

- *Noise Injection:* Flow-Noise employs a noise network for exploration, complemented by a relative entropy bonus for noise magnitude adaptation, which affords the model finer control during convergence, thus achieving better performance.
- *MDP Formulation:* Flow-Noise adopts a one-layer MDP formulation where the log-probability of the executed action is derived from the joint log-probability of the denoised sequence. This formulation endows Flow-Noise with higher data utilization efficiency, leading to faster convergence, as demonstrated in Fig. 8.

Despite this, the performance discrepancy is still marginal (e.g., 1.5% in π_0 and 0.4% in $\pi_{0.5}$). Additionally, Flow-Noise requires recomputing the entire denoising trajectory for log-likelihood computation. Consequently, the update time per RL training step scales with the number of denoising steps, whereas it remains constant for Flow-SDE due to its mixed ODE-SDE rollout strategy.

5.2.2 MANISKILL

SFT Procedure. Since the SFT dataset provided by RL4VLA lacks the state information required for π_0 , we re-synthesized trajectories following their setting using the MPLib motion planning suite (Guo et al., 2025a), with the final 15 additional frames appended to reinforce motion completion.

RL Procedure. In RL, the VLA model receives an input comprising a single 480 x 640 RGB third-person view, a short language instruction, and the current joint pose. The model also receives a structured reward signal from the environment: 1.0 for correct object placement and 0.1 for successful attachment of the gripper to the object, mitigating unwanted throwing behaviors.

Experiments. Based on π_0 and $\pi_{0.5}$ models, we empirically validate the performance of Flow-Noise against SFT baselines on SIMPLER and MultiTask two benchmarks.

In SIMPLER, the experimental setup comprises an 8-DoF WidowX-250S arm evaluated on four standard tasks: (1) *Spoon*: placing a spoon on a cloth, (2) *Carrot*: placing a carrot on a plate, (3) *Eggplant*: placing an eggplant in a basket, and (4) *Cube*: stacking a cube. For the SFT stage, we employ a curated dataset in which each task is trained with 144 demonstration episodes.

For MultiTask, the policy is prompted to picking from 16 different object types and placing them onto 17 different receptacles, distributed across 16 unique table scenes, yielding a total of 4,352 unique task combinations. Given the high complexity of this MultiTask setting, the SFT dataset was prepared with 16,384 episodes, a scale substantially larger than those for SIMPLER tasks.

Table 2: Evaluation results on the SIMPLER benchmark for π_0 and $\pi_{0.5}$.

Model		SIMPLER				
		Carrot	Eggplant	Spoon	Cube	Avg.
π_0	SFT	82.7	87.5	61.7	37.1	67.2
	Flow-Noise	95.7	96.7	91.6	63.0	86.7
	Δ	+13.0	+9.2	+29.9	+25.9	+19.5
$\pi_{0.5}$	SFT	70.6	91.9	43.5	31.0	59.2
	RL	82.0	98.2	82.8	53.3	79.1
	Δ	+11.4	+6.3	+39.3	+22.3	+19.9

Table 3: Evaluation results on the MultiTask of ManiSkill.

Model		IND	OOD			Avg.
			Visual	Semantic	Action	
π_0	SFT	41.6	43.4	4.8	10.2	19.5
	Flow-Noise	85.7	72.9	6.6	17.9	32.5
	Δ	+44.1	+29.5	+1.8	+7.7	+13.0
$\pi_{0.5}$	SFT	40.1	38.8	16.6	22.3	25.9
	Flow-Noise	81.1	59.0	25.4	39.1	41.2
	Δ	+41.0	+20.2	+8.8	+16.8	+15.3

Analysis. As detailed in Tab. 2 and Tab. 3, π_{RL} achieves substantial performance improvements across both the SIMPLER and MultiTask environments. In the SIMPLER environment, π_{RL} increases the average success rate of the π_0 model from 67.2% to 86.7%, with three tasks (carrot, eggplant, and spoon) exceeding 90% success. In the challenging, large-scale MultiTask environment, which comprises 4352 task compositions, the performance of π_0 increases from 41.6% to 85.7%, while the $\pi_{0.5}$ model improves from 40.1% to 84.8%. These results demonstrate that π_{RL} not only enhances performance significantly but also scales effectively to complex, large-scale multi-task settings.

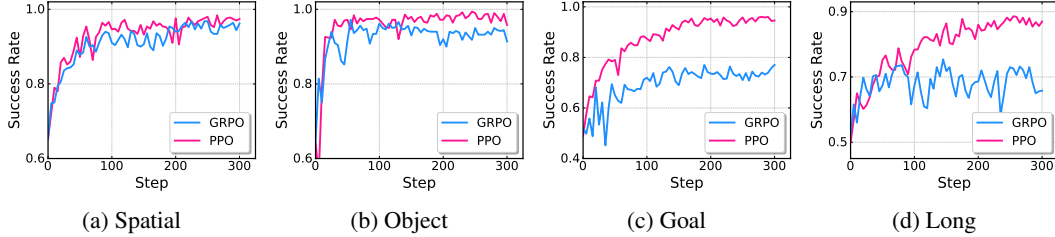
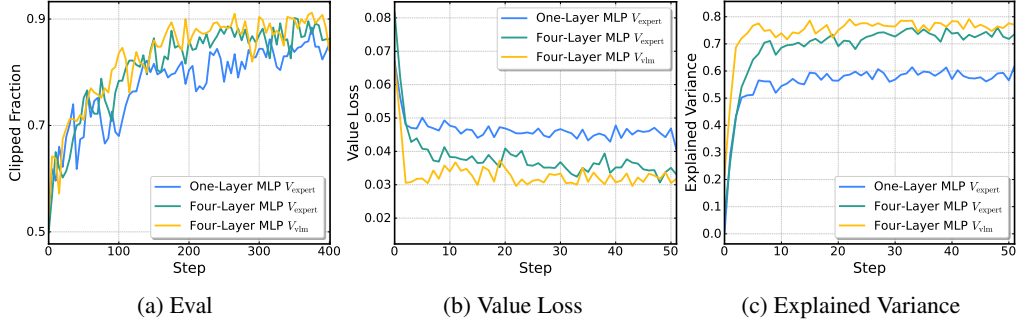
OOD Tests. Following RL4VLA, we further evaluate the model’s generalization across three challenging OOD scenarios: (1) *Vision*, challenging the model with novel backgrounds and textures; (2) *Semantics*, probing comprehension with unseen objects, varied instructions, and confounding elements like extra objects or receptacles; and (3) *Execution*, assessing robustness against varied initial states, unseen robot poses, and dynamic disturbances, such as the target object being moved during execution. In the OOD scenarios detailed in Tab. 3, we observe that the π_0 -SFT model demonstrates strong generalization for visual information. This can be attributed to the robust foundation of its VLM, which allows it to better handle visual disturbances.

However, the semantic performance of π_0 drops dramatically. This degradation is less pronounced when switching to the $\pi_{0.5}$ baseline, a benefit likely stemming from the knowledge generalization of the pre-trained $\pi_{0.5}$ model. Regarding action execution, π_0 exhibits a larger performance drop than $\pi_{0.5}$. We hypothesize that this discrepancy arises from the inclusion of joint angle states as input in π_0 , leading to severe overfitting on the control task. In contrast, $\pi_{0.5}$ omits these inputs, thereby avoiding this same degree of performance degradation.

Furthermore, while RL yields significant improvements on in-distribution tasks, we observe its gains are limited in OOD scenarios. We attribute this discrepancy to two factors we aim to address in future work. First, the SFT baseline model itself exhibits substantial performance degradation in OOD settings, which inherently caps the generalization potential achievable by the subsequent RL finetuning. Second, freezing the VLM during the RL stage for training efficiency prevents the model from adapting its visual features to the environment, consequently hindering its visual generalization capabilities.

Table 4: Comparison of the PPO and GRPO with Flow-SDE on the LIBERO.

Model		LIBERO					
		Spatial	Object	Goal	Long	Avg.	Δ Avg.
π_0	SFT	65.3	64.4	49.8	51.2	57.6	—
	+GRPO	97.8	97.8	83.2	81.4	90.0	+32.4
	+PPO	98.4	99.4	96.2	90.2	96.0	+38.4
$\pi_{0.5}$	SFT	84.6	95.4	84.6	43.9	77.1	—
	+GRPO	97.4	99.8	91.2	77.6	91.5	+14.4
	+PPO	99.6	100	98.8	93.0	97.9	+20.8

Figure 5: Visual comparison of PPO and GRPO with Flow-SDE π_0 on the LIBERO, demonstrating that PPO outperforms GRPO in terms of convergence performance and training speed.Figure 6: Ablation on the critic structure and placement within Flow-SDE π_0 on the LIBERO-Long, indicating that the critic V_{vlm} attached after the VLM exhibits superior performance. Furthermore, a four-layer MLP demonstrates stronger regression capability than a one-layer MLP in V_{expert} .

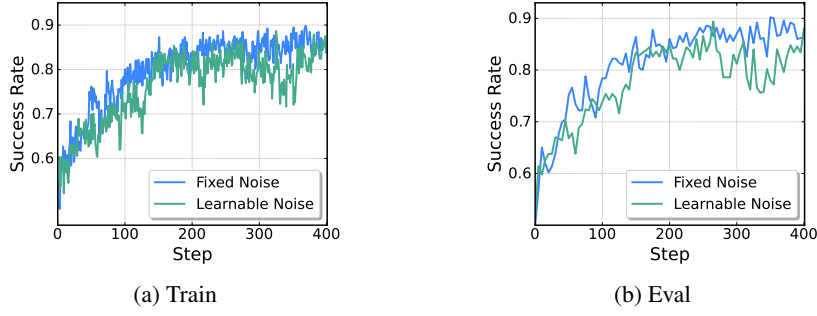
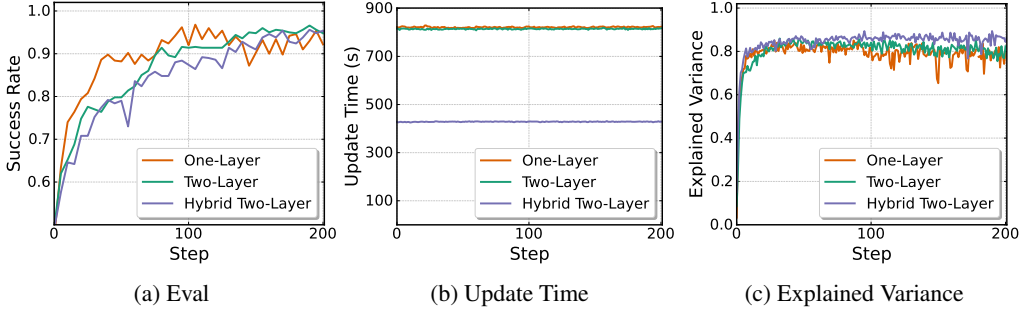
5.3 ABLATION STUDY

Given that Flow-SDE achieves performance comparable to Flow-Noise while offering higher computational efficiency, we conduct ablation studies with the Flow-SDE method on the LIBERO benchmark to investigate the impact of the RL algorithm, critic design, stochasticity injection strategy, MDP formulation, and various hyperparameters.

5.3.1 RL ALGORITHMS

Given the significant performance gains from PPO on the LIBERO benchmark, we also investigated the effectiveness of GRPO (Shao et al., 2024) (see Appendix A for a detailed description), another widely used policy gradient method applied in VLA+RL training. We compare the performance of PPO and GRPO on both the π_0 and $\pi_{0.5}$ models, with results summarized in Tab. 4.

We further visualize training curves of PPO and GRPO in Fig. 5, demonstrating that PPO outperforms GRPO in both final convergence performance and training stability across all four LIBERO task suites.

Figure 7: Ablation on the injection strategy within Flow-SDE of π_0 on the LIBERO-Long.Figure 8: Ablation on the MDP formulation within Flow-SDE of π_0 on the LIBERO-Long.

5.3.2 CRITIC DESIGN

Placement. We compare two critic placement strategies, one positioned after the action expert (V_{expert}) and the other after the VLM (V_{vlm}), with π_0 model on the LIBERO-Long task suite. As illustrated in Fig. 6, both placements yield comparable performance. However, we observe that V_{vlm} exhibits slightly superior performance, lower value loss, and higher explained variance, despite not receiving the proprioceptive state as input. This advantage can be attributed to a key difference in their input: V_{vlm} learns a direct mapping from observation to value, while V_{expert} must contend with optimization challenges arising from coupled state and noisy action inputs.

Nevertheless, to align with the design of the value function, we maintain the V_{expert} architecture for the π_0 , ensuring that state information is incorporated to calculate the value.

Structure. We investigate a four-layer MLP versus a one-layer MLP, which mirrors the action-projection structure in the action expert. Results in Fig. 6 indicate that the four-layer MLP leads to a more accurate value approximation, resulting in enhanced performance and training stability.

5.3.3 STOCHASTICITY INJECTION

Flow-Noise and Flow-SDE provide two distinct approaches for injecting stochasticity. Specifically, Flow-Noise employs a learnable noise network, while Flow-SDE uses a fixed noise level strategy as illustrated in Fig. 3. To isolate the impact of the injection strategy, we evaluate these two strategies on the LIBERO-Long task suite, with the same Flow-SDE MDP formulation. Since the fixed noise approach does not incorporate an entropy coefficient, we set the entropy bonus for learned noise to 0 to ensure a fair comparison.

We set the fixed noise level to $a = 0.5$, and the lower and upper bounds for the learnable noise log-variance to 0.08 and 0.16, respectively. As depicted in Fig. 7, two noise strategies exhibit similar train and eval performance at step 0, which indicates comparable noise magnitudes. Furthermore, the converged performance affirms the efficiency of both injection methods.

Table 5: Ablation study of hyperparameters for Flow-SDE on the LIBERO-Spatial. Performance is reported as task success rate (%). “Train” refers to policy performance during the stochastic rollout phase, whereas “Eval” refers to performance during the deterministic evaluation phase.

Models	Stage	Hyperparameters									
		Noise Level			Denoise Step				Action Chunk		
		0.2	0.5	0.8	1	2	4	8	5	10	20
SFT	Train	62.3	56.0	46.6	9.4	28.3	56.1	62.6	56.0	60.7	70.3
	Eval	65.2	65.2	65.2	63.8	64.9	65.2	63.2	65.2	70.5	72.6
RL	Train	59.5	93.5	95.3	73.8	90.8	93.5	84.3	93.5	93.3	87.5
	Eval	73.1	94.5	98.1	88.5	97.0	94.5	86.7	94.5	95.5	89.2

5.3.4 FLOW POLICY MDP

Flow-Noise and Flow-SDE also differ in their MDP formulation, as shown in Fig. 2. Built on the standard one-layer MDP, Flow-Noise directly calculates the log-likelihood of the denoised sequence for the policy update. In contrast, Flow-SDE constructs a two-layer MDP by integrating the denoising process with the environment, and further employs a hybrid ODE-SDE sampling technique for acceleration. With the same Flow-SDE noise injection strategy, we evaluate these different frameworks on the LIBERO-Long task suite, as illustrated in Fig. 8.

While the one-layer formulation converges fastest, all three frameworks achieve similar final performance. In terms of computational cost, the hybrid two-layer paradigm reduces training time by half compared to the standard two-layer approach, thanks to a shorter effective MDP chain that lowers the computational cost per RL update. Moreover, we observe that the one-layer MDP shows no significant speed advantage over the standard two-layer model, as its update stage necessitates re-computing the entire denoising trajectory to calculate the log-likelihood, resulting in comparable computational overhead.

5.3.5 HYPER-PARAMETERS

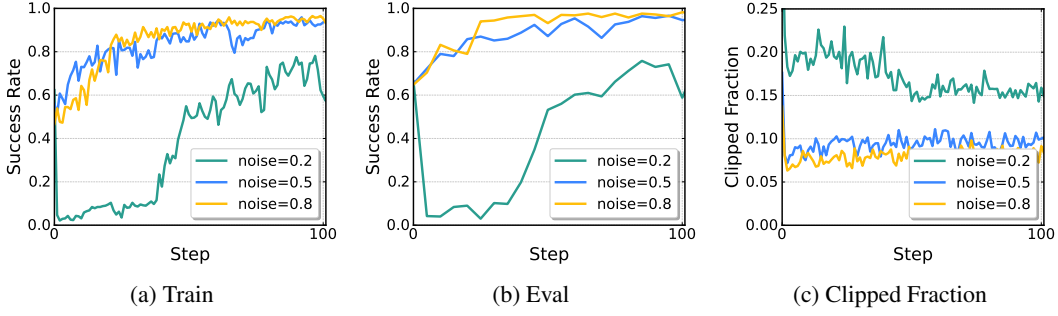
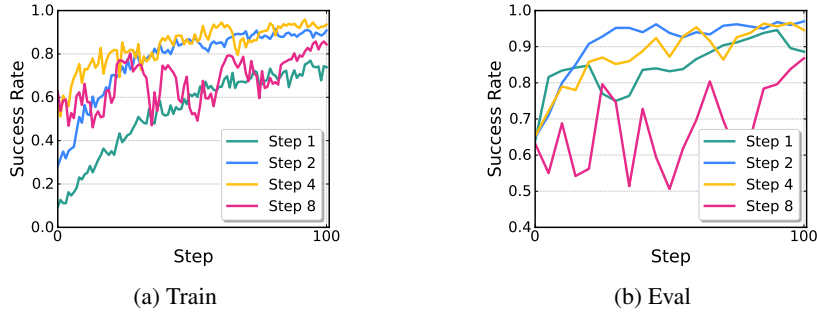
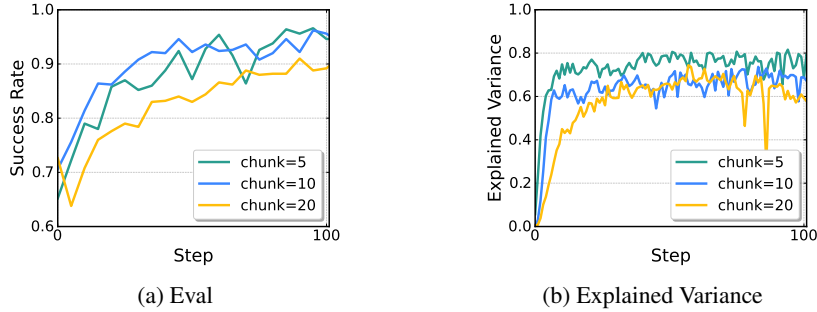
Building on the Flow-SDE with π_0 , we investigate the influence of the noise level, denoise step, and action chunk on the LIBERO-Spatial benchmark. We denote the train stage as the phase where the policy generates stochastic actions for exploration, whereas the evaluation stage involves generating deterministic actions. The train and eval success rates for the SFT baseline and the RL fine-tuned model after 100 training steps are presented in Tab. 5.

Noise Level. The noise level a in the Flow-SDE is defined in Eq. (8), which governs the noise injection magnitude during the denoising process. From Tab. 5, we observe that the SFT baseline’s eval performance is identical across all noise levels as it relies on deterministic ODE sampling. Its training performance, however, degrades as the noise level increases, which is intuitive as higher noise can disrupt the flow path and lead to an inaccurate marginal action distribution.

Extending this analysis to the RL fine-tuning stage reveals a key trade-off: while lower noise levels mitigate performance degradation induced by policy exploration, the capacity for RL refinement is correspondingly constrained. This trade-off is empirically validated in Fig. 9, which indicates that training with the minimal noise level $a = 0.2$ exhibits instability, manifesting as a significantly higher clip fraction. We attribute this instability to the substantially larger gradient magnitudes induced by the low noise level.

Denoise Step. The denoise step K defines the number of discretization steps for action generation and is critical for controlling the fidelity of the ODE-to-SDE transition in Eq. (8). In Tab. 5, we observe that while all configurations start with similar eval performance, the train success rate plummets at $K = 1$, indicating a significant ODE-to-SDE discretization error.

However, as in our noise-level analysis, a larger K is not necessarily optimal. As shown in Fig. 10, a larger K introduces a clear trade-off: it yields higher rollout performance but complicates the training process due to an increased number of denoising steps.

Figure 9: Ablation on the noise level a , conducted with the Flow-SDE π_0 on the LIBERO-Spatial.Figure 10: Ablation on the denoise step, conducted with the Flow-SDE π_0 on the LIBERO-Spatial.Figure 11: Ablation on the chunk size, conducted with the Flow-SDE π_0 on the LIBERO-Spatial.

Action chunk. The action chunk refers to the number of consecutive actions the policy executes within a single observation. We ablate the action chunk size across 5, 10, and 20, with results presented in Tab. 5 and further visualized in Fig. 11.

While a larger chunk size yields a marginal performance improvement, it also reduces the frequency of policy-environment interactions and hinders accurate reward credit assignment. These factors contribute to less reliable advantage estimation, as reflected in the explained variance metric. Consequently, while a large chunk size may provide a stronger SFT baseline, it ultimately constrains the potential gains from subsequent RL fine-tuning. In conclusion, our analysis reveals a consistent trade-off:

Caveat: Hyperparameters optimized for rollout might induce training instability, impeding potential performance gains from RL.

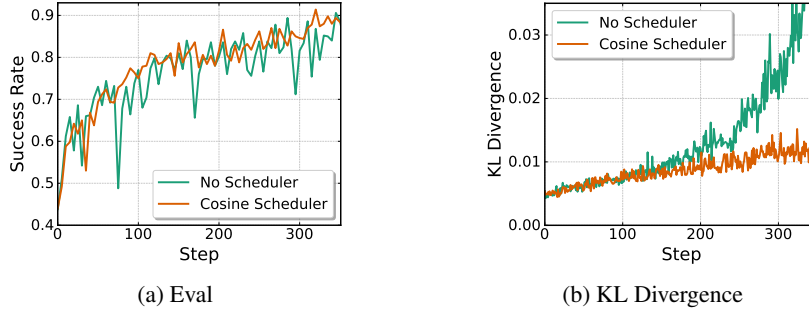


Figure 12: Ablation study on the learning rate scheduler. The experiment is conducted with Flow-SDE $\pi_{0.5}$ on the LIBERO-Long benchmark, demonstrating that the scheduler alleviates over-optimization and stabilizes the training process.

Therefore, a careful selection of these parameters is essential to achieve a suitable balance between train performance and a stable training process.

5.4 INSIGHTS FROM LARGE-SCALE TRAINING

In this subsection, we elaborate on some empirical insights we gained during RL training.

Hyperparameters. According to the hyperparameters ablation detailed in Sec. 5.3.5, the performance disparity between the train and eval performance of the initial SFT checkpoint warrants close attention. If this disparity is significant, we recommend either reducing the noise magnitude or increasing the number of denoising steps to mitigate the performance degradation caused by the discrepancy between deterministic and stochastic action generation. Furthermore, as previously established, lower noise levels yield larger gradients, requiring a smaller learning rate to maintain training stability.

We also observed that when train performance improves steadily while eval performance oscillates, increasing the number of denoising steps can help alleviate this, benefiting from reduced divergence in the action distributions between the deterministic and stochastic action generation processes. Regarding the action chunk, we empirically found that long-horizon tasks benefit from larger chunk sizes. For instance, we set the chunk size to 10 for LIBERO-Long and 5 for the other sub-tasks.

Training. In our $\pi_{0.5}$ experiments on the LIBERO-Long benchmark, we observed that the Kullback–Leibler (KL) divergence metric increased steadily throughout training, potentially leading to instability. We mitigated this issue by implementing a learning rate scheduler with cosine annealing. As demonstrated in Fig. 12, this scheduler effectively prevents the KL divergence from escalating, thereby stabilizing the training process.

Critic. In our ManiSkill experiments, we observe that policy evaluation performance exhibits an initial dip before improving for both π_0 and $\pi_{0.5}$ models, as shown in Fig. 13. We attribute this transient degradation to the critic providing inaccurate signals during its warm-up phase. The subsequent eval improvement correlates directly with the critic’s value estimations stabilizing, as evidenced by the rising explained variance.

5.5 EXTENSION: FINE-TUNE VLM AND ACTION EXPERT SIMULTANEOUSLY

In our previous experiments, the VLM is frozen, and the optimization is confined exclusively to the action expert during RL. In this subsection, we aim to investigate the role of the VLM during RL. Specifically, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022) for the VLM, facilitating its joint optimization with the action expert. We set the LoRA rank to $r = 32$ and the scaling parameter to $\alpha = 32$, while the action expert remains fully trainable.

We conduct experiments with the π_0 model with Flow-SDE on the LIBERO-Long benchmark, comparing three distinct configurations: 1) VLM frozen baseline (5×10^{-6} learning rate, 4 updates per epoch), 2) VLM LoRA-I (5×10^{-6} learning rate, 4 updates per epoch), and 3) VLM LoRA-II with

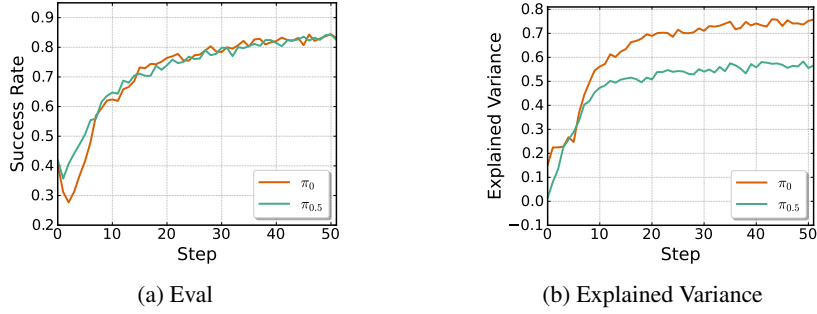


Figure 13: Training curve on the MultiTask of ManiSkill with Flow-Noise on the π_0 and $\pi_{0.5}$ models.

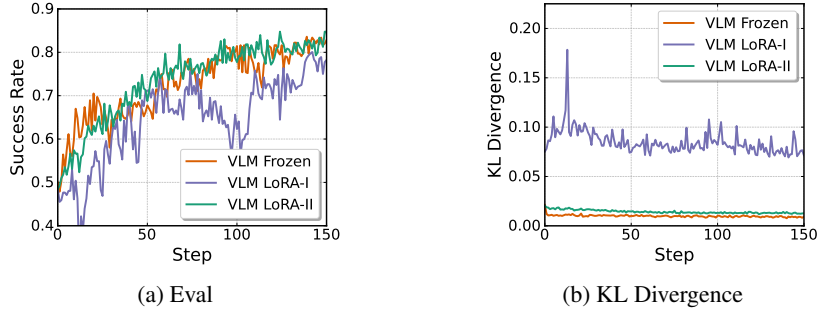


Figure 14: Ablation study on VLM Effectiveness during RL. The experiment is conducted with Flow-SDE π_0 on the LIBERO-Long benchmark. We compare the performance of a frozen VLM baseline (learning rate 5×10^{-6} , 4 updates per epoch) against two LoRA-tuned VLM configurations: LoRA-I (using the same training config) and LoRA-II (a more conservative setting with learning rate 1×10^{-6} and 2 updates per epoch).

conservative update training config (1×10^{-6} learning rate, 2 updates per epoch). As presented in Fig. 14, the VLM LoRA-II configuration achieves a comparable learning trajectory with the VLM frozen baseline. This empirical observation yields two critical inferences: First, the benefit of fine-tuning the VLM on the LIBERO benchmark is not evident; Second, fine-tuning VLM together with the action expert requires a more conservative optimization configuration for training stability. We conjecture the limited performance gain attributable to the limited scene variability within LIBERO, for which the pretrained VLM representations are already sufficiently robust.

6 CONCLUSION

We introduce π_{RL} , the first framework that enables flow-based VLAs, π_0 and $\pi_{0.5}$, to be fine-tuned with PPO. We tackle the fundamental challenge of intractable log-likelihoods in flow matching and propose two technical solutions, Flow-Noise and Flow-SDE, which differ in their stochasticity injection strategies and MDP formulations. Our extensive experiments on the challenging LIBERO and ManiSkill benchmarks demonstrated that π_{RL} achieves significant performance improvements over SFT baselines.

7 LIMITATIONS AND FUTURE WORK

Noise Injection. Our current noise injection strategy exhibits some train performance drop during the ODE-to-SDE conversion. Flow-CPS (Wang & Yu, 2025) attributes this loss to numerical error and proposes an improved coefficients-preserving sampling method. In our experiments, we attempted this configuration. Consistent with our hyperparameter ablation, our experiments showed that while this configuration mitigated the ODE-SDE precision error, it yielded limited RL improvement. Nevertheless, we argue that improving the noise injection strategy holds significant potential,

specifically converting the ODE formulation to an SDE formulation while preserving the action distribution undisturbed.

Training Acceleration. Our current implementation of the mixed ODE-SDE rollout is simplistic in Flow-SDE, *i.e.*, it randomly selects one denoising step as an SDE step, while all other steps remain ODE steps. We posit that future investigations into mixed ODE-SDE rollouts, leveraging advances in accelerating flow-based image generation (Li et al., 2025b; He et al., 2025; Liu et al., 2025a; Li et al., 2025c), could further enhance Flow-SDE, leading to faster training and improved performance.

Generalization. Our experiments in the Maniskill OOD tests indicate that the semantic generalization capabilities of the SFT and RL models remain limited. We aim to investigate and improve this issue in future studies.

Real-world Experiment. Our current experiments are evaluated solely in simulated environments, lacking empirical validation in a physical system. We plan to extend this research by applying our RL methodology to real-world tasks in the future.

REFERENCES

- Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.
- Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0 : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.
- Muhayy Ud Din, Waseem Akram, Lyes Saad Saoud, Jan Rosell, and Irfan Hussain. Vision language action models in robotic manipulation: A systematic review. *arXiv preprint arXiv:2507.10672*, 2025.
- Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhao Ye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.
- Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5): 701–739, 2025.
- Runlin Guo, Xinsong Lin, Minghua Liu, Jiayuan Gu, and Hao Su. Mplib: a lightweight motion planning library. <https://github.com/haosulab/MPLib>, 2025a. URL <https://motion-planning-lib.readthedocs.io/latest/>.
- Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. *arXiv preprint arXiv:2501.16664*, 2025b.
- Xiaoxuan He, Siming Fu, Yuke Zhao, Wanli Li, Jian Yang, Dacheng Yin, Fengyun Rao, and Bo Zhang. Tempflow-grpo: When timing matters for grpo in flow models. *arXiv preprint arXiv:2508.04324*, 2025.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.
- Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.
- Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.
- Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *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*, 2025.
- Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.
- Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025a.
- Junzhe Li, Yutao Cui, Tao Huang, Yinping Ma, Chun Fan, Miles Yang, and Zhao Zhong. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025b.
- Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.
- Yuming Li, Yikai Wang, Yuying Zhu, Zhongyu Zhao, Ming Lu, Qi She, and Shanghang Zhang. Branchgrpo: Stable and efficient grpo with structured branching in diffusion models. *arXiv preprint arXiv:2509.06040*, 2025c.
- Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.
- Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *arXiv preprint arXiv:2505.05470*, 2025a.
- Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can rl bring to vla generalization? an empirical study. *arXiv preprint arXiv:2505.19789*, 2025b.
- Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.
- Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.
- Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.
- David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.
- Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *Advances in Neural Information Processing Systems*, 37:4062–4089, 2024.
- Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.
- Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.
- Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.
- Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Andrew Wagenmaker, Mitsuhiro Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.
- Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.
- Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Xiangyuan Wang, Xu Fu, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation, 2025. URL <https://arxiv.org/abs/2509.15965>.
- Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, Zhihao Liu, Kang Chen, Wenhao Tang, Quanlu Zhang, Weinan Zhang, Chao Yu, and Yu Wang. Rlinf-vla: A unified and efficient framework for vla+rl training, 2025. URL <https://arxiv.org/abs/2510.06710>.
- Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

A ALGORITHM DETAILS

GRPO is a critic-free method that estimates the advantage by normalizing rewards within a group of rollouts from the same state. In our robotics MDP task, for each initial state, we use the policy π_θ to sample a group of G trajectories, resulting in G sparse terminal rewards $\{\mathcal{R}^{(j)}\}_{j=1}^G$ denoting the binary success of the task. The advantage for the i -th trajectory, $\hat{A}^{(i)}$, is then calculated based on the group-wise reward normalization:

$$\hat{A}^{(i)} = \frac{\mathcal{R}^{(i)} - \text{mean}(\{\mathcal{R}^{(j)}\}_{j=1}^G)}{\text{std}(\{\mathcal{R}^{(j)}\}_{j=1}^G)} \quad (17)$$

where $\mathcal{R}^{(i)}$ is the terminal reward for the i -th trajectory, and the mean and standard deviation are computed over the group of G trajectories. Since the reward is only granted at the end of an episode, the advantage estimate remains constant across all timesteps within that trajectory.

B EXPERIMENT DETAILS

We record the training hyperparameters used to train both π_0 and $\pi_{0.5}$ on each LIBERO task, and present them in Tabs. 6 and 7.

Table 6: Hyperparameters of Flow-Noise and Flow-SDE with PPO across LIBERO tasks.

Parameters	Algorithms and tasks							
	π_0				$\pi_{0.5}$			
	Spatial	Object	Goal	Long	Spatial	Object	Goal	Long
Train epochs	400	400	400	400	400	400	400	400
Batch size	2048	2048	2048	2048	2048	2048	2048	2048
Update epochs	2	2	4	4	1	1	3	4
Actor lr	1e-5	5e-6	5e-6	5e-6	5e-6	5e-6	5e-6	5e-6
Critic lr	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4	1e-4
Scheduler	False	False	False	False	False	False	False	True
Reward discount rate γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE λ	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Clip ratio ϵ	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Interaction steps	240	320	320	480	240	320	320	480
Parallel environments	64	64	64	64	64	64	64	64
Rollout epochs	8	8	8	8	8	8	8	8
Action chunk H	5	5	5	10	5	5	5	10
Denoise steps	4	4	4	4	3	5	5	5
Noise level σ (Flow-SDE)	0.5	0.5	0.5	0.5	0.5	0.3	0.3	0.5
Max log-var (Flow-Noise)	0.16	0.16	0.16	0.16	0.10	0.10	0.10	0.10
Min log-var (Flow-Noise)	0.08	0.08	0.08	0.08	0.04	0.04	0.04	0.04
Entropy bonus (Flow-Noise)	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005

Table 7: Hyperparameters of Flow-Noise and Flow-SDE with PPO across ManiSkill tasks.

Parameters	Algorithms and tasks									
	π_0					$\pi_{0.5}$				
	Eggplant	Carrot	Spoon	Cube	Multitask	Eggplant	Carrot	Spoon	Cube	Multitask
Train steps	40	40	40	130	400	40	40	40	70	400
Batch size	2560	2560	2560	2560	5120	2560	2560	2560	2560	5120
Update epochs	4	4	4	4	5	4	4	4	4	5
Actor lr	5.6e-6	5.6e-6	5.6e-6	5.6e-6	7.91e-6	5.6e-6	5.6e-6	5.6e-6	5.6e-6	7.91e-6
Critic lr	1.1e-4	1.1e-4	1.1e-4	1.1e-4	1.55e-4	1.1e-4	1.1e-4	1.1e-4	1.1e-4	1.55e-4
Scheduler	False	False	False	False	False	False	False	False	False	False
Reward discount rate γ	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
GAE λ	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Clip ratio ϵ	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Interaction steps	48	48	48	48	48	48	48	48	48	48
Parallel environments	256	256	256	256	320	256	256	256	256	320
Rollout epochs	1	1	1	1	1	1	1	1	1	1
Action chunk H	5	5	5	5	5	5	5	5	5	5
Denoise steps	4	4	4	4	4	4	4	4	4	4
Noise level σ (Flow-SDE)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Max log-var (Flow-Noise)	0.16	0.16	0.16	0.16	0.16	0.10	0.10	0.10	0.10	0.10
Min log-var (Flow-Noise)	0.08	0.08	0.08	0.08	0.08	0.04	0.04	0.04	0.04	0.04
Entropy bonus (Flow-Noise)	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005