LiFT: Likelihood-Free Tree-Structured Policy Optimization for Flow-Based VLAs

Junjie Gao* **Siyuan Song*** **Zixuan Zhang** **Feiyang Song**
Yongzhou Pan **Nuan Wen** **Yaosheng Deng** **Xue Tian** **Mir Feroskhan†**

School of Mechanical and Aerospace Engineering
Nanyang Technological University
Singapore

Abstract

Flow-based vision-language-action (VLA) models provide expressive action generation for embodied control, but supervised fine-tuning (SFT) often confines them to narrow expert behaviors. Online reinforcement learning (RL) can improve beyond demonstrations through environment interaction, yet flow-based VLAs struggle with sparse long-horizon rewards and intractable action likelihoods. We propose ***LiFT***, a likelihood-free tree policy optimization framework for flow-based VLAs. LiFT first addresses sparse-reward credit assignment by creating sibling continuations from shared rollout histories and using their final outcomes to compute branch level relative advantages. LiFT further uses local action-flow instability as an expansion criterion, focusing rollout budget on ambiguous action-generation regions that are most likely to reveal meaningful outcome differences. Finally, LiFT applies branch-level advantages through a decision-level surrogate ratio on executed action chunks, matching policy updates to the granularity of tree-based credit assignment. This yields a likelihood-free policy update without attributing chunk-level credit to individual denoising steps. Evaluations on in-distribution and out-of-distribution benchmarks show that LiFT improves task success and robustness to distribution shifts.

1 Introduction

Vision-language-action (VLA) models have become a central paradigm for generalist robotic manipulation, bridging high-level multimodal reasoning and low-level physical control [Din et al., 2025, Lv et al., 2025, 2026]. Early systems use discrete action tokens or direct continuous regression [Zitkovich et al., 2023, Kim et al., 2024], while recent models adopt generative policies to produce temporally coherent actions for manipulation and navigation [Black et al., 2024, 2025, Bjorck et al., 2025, Chen et al., 2026, Sridhar et al., 2024]. Built on pretrained Vision-Language Models (VLMs), modern VLAs acquire visuomotor skills from large-scale heterogeneous robot demonstrations [O'Neill et al., 2024, Khazatsky et al., 2024] and are further adapted by target-task SFT. However, this demonstration-driven pipeline is constrained by trajectory coverage [Zhang et al., 2024a]. SFT fits expert behaviors within the demonstrated manifold, but often lacks recovery ability once closed-loop execution drifts under perturbations or distribution shifts [Wang et al., 2026, Fei et al., 2025b, Zhang et al., 2025b]. Online reinforcement learning (RL) therefore offers a natural post-training stage for VLAs, using reward-guided interaction to improve task success, data efficiency, and out-of-distribution generalization beyond SFT [Lu et al., 2025, Liu et al., 2025b, Li et al., 2025a, Zang et al., 2025].

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

However, flow-based VLAs are difficult to optimize with RL because their action chunks are generated by integrating a learned velocity field, typically with an ordinary differential equation (ODE) sampler. This supports high-frequency dexterous control, but lacks tractable executed-action likelihoods and provides limited stochasticity for online exploration. Recent methods tackle this challenge by either modifying the sampler or reshaping the supervision. πRL converts the ODE sampler into an SDE and optimizes a two-layer MDP [Chen et al., 2025a, Zhang et al., 2025a], which couples updates to the sampler and lengthens the denoising horizon. π-StepNFT uses step-wise contrastive supervision [Wang et al., 2026], but its shared trajectory-level success/failure label still yields coarse supervision.

Sparse outcome rewards further create a credit-assignment bottleneck in long-horizon manipulation. Since a final success or failure label cannot identify the causal action chunks, trajectory-level updates may reinforce correct behaviors and critical errors indiscriminately. Prior work makes sparse feedback denser using learned reward and progress models [Liang et al., 2026, Tan et al., 2025a], as well as objectives based on task stages or subgoals [Xu et al., 2025, Zhang et al., 2024b]. These methods improve feedback granularity, but rely on auxiliary evaluators, progress labels, or task-specific decomposition.

To address these challenges, we propose ***LiFT***, a tree-structured policy optimization framework for flow-based VLAs. First, LiFT extracts informative credit signals from sparse outcome rewards through tree-structured rollouts. Shared prefixes enable efficient generation of multiple leaf trajectories, while sibling outcome comparisons yield fine-grained advantages rather than uniform credit across entire trajectories. To make tree expansion informative rather than uniform, LiFT further introduces uncertainty-guided branching. It selects branching nodes by probing the local stability of the action-flow field, where high velocity variance under re-noising perturbations reveals ambiguous action-generation regions likely to yield divergent outcomes. Second, LiFT constructs a surrogate ratio from the flow-matching loss difference, enabling branch-level advantages to optimize executed action chunks without computing explicit likelihoods over the multi-step flow sampler.

Our contributions are summarized as follows:

- We propose LiFT, a tree-structured policy optimization framework for sparse-reward VLA reinforcement learning. LiFT branches at the level of action chunks and allocates expansion budget to uncertain decision states, deriving branch-level advantages by comparing sibling continuations under shared prefixes.
- We formulate a likelihood-free policy update for flow-based VLA policies. Tree-propagated advantages directly weight the conditional flow-matching objective on executed action chunks, avoiding likelihood ratio estimation over the full stochastic sampling path.
- We conduct extensive experiments on standard robotic manipulation benchmarks, including LIBERO [Liu et al., 2023] and ManiSkill [Mu et al., 2021], to evaluate LiFT across flow-based VLA backbones, task suites, and sparse-reward settings. We further conduct real-world experiments to validate LiFT beyond simulation.

2 Related Works

2.1 Reinforcement Learning for Vision-Language-Action Policies

Recent VLA-RL methods extend supervised fine-tuning with online reward-based optimization to improve data efficiency, long-horizon execution, and closed-loop generalization [Li et al., 2025a, Liu et al., 2025b, Zang et al., 2025, Lu et al., 2025, Tan et al., 2025b, Chen et al., 2025c]. Existing work has explored PPO-style actor-critic updates [Liu et al., 2025b, Chen et al., 2025a, Lu et al., 2025], DPO-style preference optimization [Liu et al., 2025b], self-referential or process-reward construction [Fei et al., 2025a, Tan et al., 2025b], and GRPO-style group-relative optimization [Li et al., 2025a, Zang et al., 2025].

Flow-based VLAs introduce an additional challenge: action chunks are generated through multi-step flow-matching dynamics, making the likelihood of the executed action difficult to compute and limiting the direct application of standard policy-gradient objectives [Black et al., 2024, 2025, Chen et al., 2025a, Zhang et al., 2025a, Wang et al., 2026]. Existing flow-based VLA-RL and generative-policy optimization methods mainly address this likelihood gap through two routes. The

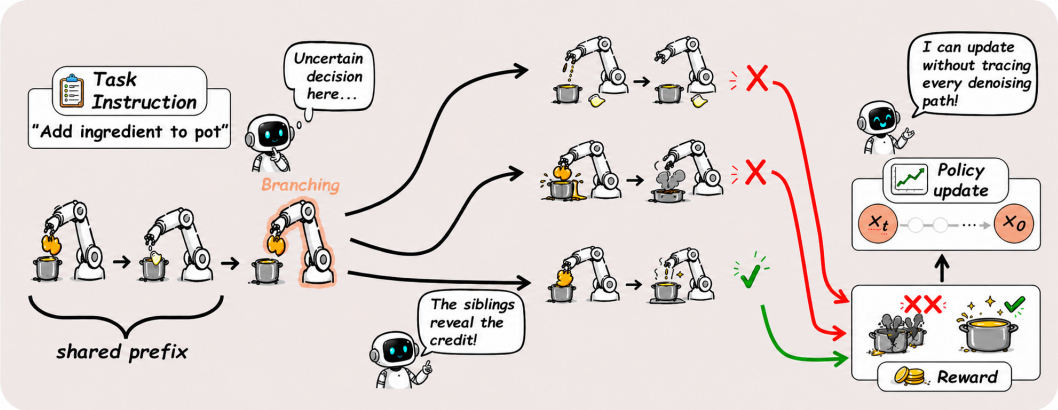

Figure 1: Conceptual overview of LiFT. LiFT branches from shared rollout prefixes at uncertain action-generation states, compares sibling outcomes to assign branch-level credit under sparse rewards, and updates the flow-based action policy with a surrogate ratio without tracing every denoising path.

first restores tractable transition probabilities with ODE-to-SDE conversion, Flow-SDE sampling, or learnable noise injection [Liu et al., 2025a, Chen et al., 2025a, Zhang et al., 2025a, Li et al., 2025b, He et al., 2025]. The second bypasses exact executed-action likelihoods with surrogate supervision, including value or preference feedback, reward-weighted flow-matching objectives, and likelihood-free contrastive updates [Li et al., 2025d, Intelligence et al., 2025, McAllister et al., 2025, Zheng et al., 2025, Wang et al., 2026].

2.2 Sparse Reward Reinforcement Learning for Long-Horizon Tasks

Sparse terminal rewards obscure which action chunks drive success or failure in long-horizon manipulation. Existing methods mitigate this sparsity by augmenting outcome rewards with intermediate feedback, including learned process reward models [Lu et al., 2025, Liang et al., 2026, Tan et al., 2025a], stage-aware or self-referential feedback [Chen et al., 2025b, Fei et al., 2025a, Xu et al., 2025], and structured rewards guided by preferences, keypoints, keyframes, or task-specific objectives [Zhang et al., 2024b, Qiu et al., 2026, Zhang et al., 2026, Luo et al., 2025]. These methods expose intermediate task progress and improve optimization efficiency, but often depend on additional reward construction, such as learned reward models, predefined stages, keyframes, or task-specific progress estimators.

Beyond reward densification, tree-structured policy optimization offers a complementary route for credit assignment from sparse outcomes. Recent methods branch multiple continuations from shared prefixes and compare sibling outcomes to derive segment-level advantages, improving rollout efficiency and credit assignment without relying on separately trained process reward models [Ji et al., 2025, Hou et al., 2025, Yang et al., 2025, Cao et al., 2026, Li et al., 2025c]. However, these methods are mainly developed for language reasoning, agentic tasks, or generative model post-training, and do not address action-chunk credit assignment for flow-based VLA policies.

3 Methodology

We defer the full MDP formulation and flow-based VLA preliminaries to Appendix A. Building on this formulation, we present LiFT, a tree-structured policy optimization method for flow-based VLAs.

3.1 Tree-Structured Rollout Collection

Tree rollout construction. LiFT collects online experience as tree-structured rollouts rather than independent trajectories, so that continuations sharing the same prefix can be compared under sparse terminal rewards. For each task condition c, LiFT constructs M action-chunk trees $\{T_i\}_{i=1}^{M}$. Each tree starts from the initial environment state and is expanded online as the policy interacts with the

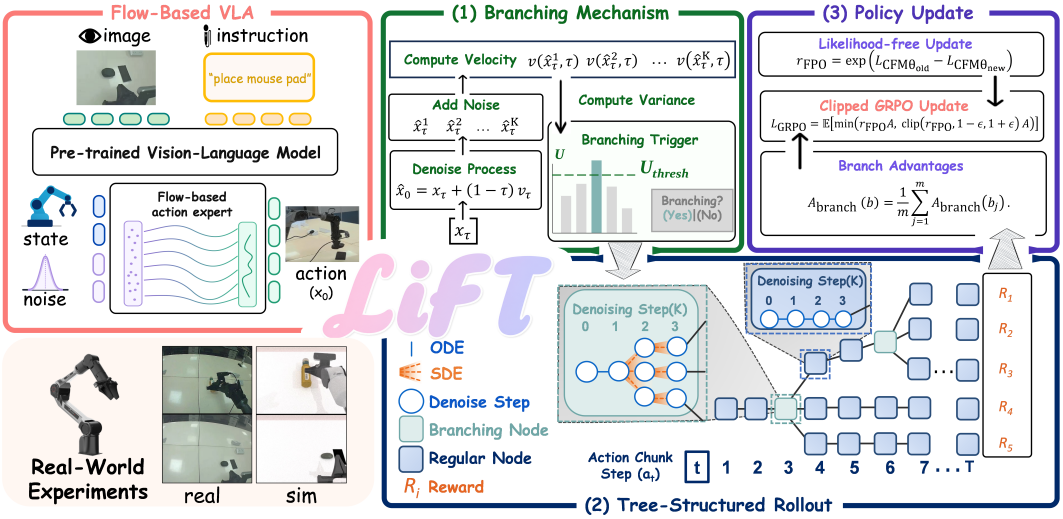

Figure 2: System overview of LiFT. LiFT uses velocity-variance probing as a branching criterion for shared-prefix tree rollouts. Sparse terminal rewards are propagated backward to compute branch-level advantages, which are used in a clipped GRPO objective with a likelihood-free surrogate ratio. This enables online optimization without exact flow-path likelihoods.

environment. A node in T_i represents a policy-generated action chunk $\mathbf{a}_{i,t} \sim \pi_\theta(\cdot \mid s_{i,t}, c)$, where $s_{i,t}$ is the rollout state before executing the chunk.

During the generation of a candidate node, LiFT computes its uncertainty score from the corresponding action chunk generation process. If the candidate node satisfies the branching criterion, LiFT treats the original chunk as a branching proposal and replaces it with N candidate action chunks sampled from its prefix state using the Flow-SDE sampler. Each candidate induces a distinct branch, which is rolled out until it reaches the next branching point or terminates the episode. We denote all branches in T_i by $\mathcal{B}(T_i)$.

A continuation that terminates forms a leaf $\ell \in \mathcal{L}(T_i)$, corresponding to a completed root-to-leaf robot trajectory with sparse outcome reward $R_{i,\ell}$. Once all continuations terminate, the leaf rewards $\{R_{i,\ell}\}_{\ell=1}^{L_i}$ are used to construct tree-structured advantages.

Uncertainty-guided branching. LiFT uses an uncertainty score as its branching criterion. This score identifies states where the flow policy is locally unstable, allowing the branching budget to focus on continuations that are more likely to reveal informative outcome differences.

For an action chunk $\mathbf{a}_{i,t}$, let $\mathbf{a}_{i,t}^\tau$ denote its intermediate state at denoising time τ, and let

$$\mathbf{v}_{i,t}^\tau = v_{\theta_{\text{old}}}\left(\mathbf{a}_{i,t}^\tau, \tau, o_{i,t}, c\right)$$

be the velocity predicted by the rollout policy. We first estimate the clean action chunk induced by the current flow state:

$$\hat{\mathbf{a}}_{i,t} = \mathbf{a}_{i,t}^\tau + (1-\tau)\mathbf{v}_{i,t}^\tau. \tag{1}$$

We then probe local flow stability by re-noising this estimated clean action chunk with K independent Gaussian noises:

$$\tilde{\mathbf{a}}_{i,t}^{\tau,k} = \tau\hat{\mathbf{a}}_{i,t} + (1-\tau)\boldsymbol{\epsilon}^k, \quad \boldsymbol{\epsilon}^k \sim \mathcal{N}(0, I), \quad k = 1, \ldots, K. \tag{2}$$

The rollout policy predicts velocities for these perturbed intermediate states:

$$\tilde{\mathbf{v}}_{i,t}^{\tau,k} = v_{\theta_{\text{old}}}\left(\tilde{\mathbf{a}}_{i,t}^{\tau,k}, \tau, o_{i,t}, c\right). \tag{3}$$

The uncertainty score is defined as the average coordinate-wise variance of the predicted velocities:

$$U_{i,t} = \frac{1}{D}\sum_{d=1}^{D} \mathrm{Var}_{k=1}^{K}\left(\tilde{v}_{i,t,d}^{\tau,k}\right), \tag{4}$$

where D is the action-chunk dimension. LiFT uses a window-constrained threshold rule to keep a fixed tree shape: the horizon is divided into W windows, and each window expands the first candidate with $U_{i,t} > U_{\text{thresh}}$, or the last candidate if no threshold crossing occurs.

Flow-SDE expansion. Following the ODE-to-SDE conversion for flow-based VLAs [Chen et al., 2025a], the deterministic flow trajectory is converted into an equivalent stochastic process that injects exploration noise while preserving the marginal action distribution. Let $\mathbf{a}_{i,t}^{\tau}$ denote the intermediate action state for chunk step t in tree T_i at denoising time τ, and let $\mathbf{v}_{i,t}^{\tau} = v_\theta(\mathbf{a}_{i,t}^{\tau}, \tau, o_t)$. The Flow-SDE expansion is given by

$$d\mathbf{a}_{i,t}^{\tau} = \left[\mathbf{v}_{i,t}^{\tau} + \frac{\sigma_\tau^2}{2(1-\tau)}\left(\tau\mathbf{v}_{i,t}^{\tau} - \mathbf{a}_{i,t}^{\tau}\right)\right] d\tau + \sigma_\tau dw_\tau. \tag{5}$$

After discretization, the one-step transition is

$$p(\mathbf{a}_{i,t}^{\tau+\delta} \mid \mathbf{a}_{i,t}^{\tau}) = \mathcal{N}\left(\mathbf{a}_{i,t}^{\tau} + \left[\mathbf{v}_{i,t}^{\tau} + \frac{\sigma_\tau^2}{2(1-\tau)}\left(\tau\mathbf{v}_{i,t}^{\tau} - \mathbf{a}_{i,t}^{\tau}\right)\right]\delta, \sigma_\tau^2\delta I\right). \tag{6}$$

By repeatedly sampling this stochastic transition from the prefix state associated with the branching proposal, LiFT obtains N candidate action chunks $\{\mathbf{a}^{(j)}\}_{j=1}^{N}$. Each candidate chunk induces an outgoing branch, which is rolled out as a distinct continuation from the same prefix.

3.2 Tree-Structured GRPO

Given the completed action-chunk trees from Sec. 3.1, LiFT optimizes the policy with a tree-structured group-relative objective. For each tree T_i, terminal rewards are normalized into leaf advantages and then propagated backward through the branch structure. This produces branch-level advantages for policy updates, avoiding the coarse assignment of a single trajectory reward to all action chunks.

For each completed tree T_i, we normalize the terminal rewards across its leaves. The tree-level reward statistics are computed as

$$\mu_i = \frac{1}{L_i}\sum_{\ell\in\mathcal{L}(T_i)} R_{i,\ell}, \qquad \sigma_i = \sqrt{\frac{1}{L_i}\sum_{\ell\in\mathcal{L}(T_i)}(R_{i,\ell} - \mu_i)^2}. \tag{7}$$

The normalized leaf advantage is

$$\hat{A}_{\text{leaf},i}(\ell) = \frac{R_{i,\ell} - \mu_i}{\sigma_i}. \tag{8}$$

We then propagate leaf advantages backward through the tree to obtain branch-level advantages. For a branch $b \in \mathcal{B}(T_i)$, let $\text{child}(b)$ denote the set of branches that continue from the endpoint of b. If b directly terminates at a leaf ℓ, its advantage is initialized as

$$\hat{A}_{\text{branch},i}(b) = \hat{A}_{\text{leaf},i}(\ell). \tag{9}$$

Otherwise, if b reaches another branching point with outgoing branches $\text{child}(b) = \{b_j\}_{j=1}^{m}$, its advantage is computed by averaging the already computed child branch advantages:

$$\hat{A}_{\text{branch},i}(b) = \frac{1}{m}\sum_{j=1}^{m}\hat{A}_{\text{branch},i}(b_j). \tag{10}$$

This recursion is applied in leaf-to-root order, so each branch receives the average normalized outcome of all downstream continuations. The averaging reduces rollout noise from the shared prefix and yields a lower-variance branch advantage. We provide the corresponding variance analysis in Appendix B.

The resulting branch advantage is used as the shared optimization signal for the policy-generated action chunks along that branch. For each branch b, let $\mathcal{K}(b)$ denote the set of such action chunks

that are included in the policy update. For each $\mathbf{a}_{i,t} \in \mathcal{K}(b)$, we use $\hat{A}_{\mathrm{branch},i}(b)$ as its advantage. Following the clipped GRPO objective, the tree-level loss is

$$\mathcal{L}^{(i)}_{\mathrm{GRPO}}(\theta) = -\sum_{b\in\mathcal{B}(T_i)}\sum_{\mathbf{a}_{i,t}\in\mathcal{K}(b)} \min\left(\rho_{i,t}(\theta)\hat{A}_{\mathrm{branch},i}(b), \mathrm{clip}\left(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_{\mathrm{branch},i}(b)\right). \tag{11}$$

Here, $\rho_{i,t}(\theta)$ is the policy ratio for the executed action chunk $\mathbf{a}_{i,t}$:

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(\mathbf{a}_{i,t} \mid o_{i,t}, c)}{\pi_{\theta_{\mathrm{old}}}(\mathbf{a}_{i,t} \mid o_{i,t}, c)}. \tag{12}$$

For flow-based VLA policies, the exact likelihood ratio $\rho_{i,t}(\theta)$ is intractable. We therefore replace it with a flow-matching surrogate ratio, as described next.

3.3 Flow-Matching Policy Ratio

The tree-structured objective in Sec. 3.2 produces advantages at the level of rollout branches, and each branch is composed of executed action chunks. Therefore, the policy ratio should also be evaluated at the same decision level. Directly applying the policy ratio over the full flow sampling path would require likelihoods over all denoising transitions, which is both intractable for flow-based VLAs and misaligned with our branch-level credit assignment. LiFT instead evaluates the ratio on the executed clean action chunk, using a flow-matching surrogate following Flow Policy Optimization (FPO) [McAllister et al., 2025]. Although rollout branches are generated with Flow-SDE, the ODE-to-SDE construction preserves the terminal distribution of the sampled action chunk. Therefore, the FPO-style ratio can still be evaluated on the action chunk sampled by Flow-SDE; see Appendix C.

Starting from the exact policy ratio in Eq. (12), we follow Flow Policy Optimization (FPO) [McAllister et al., 2025] and replace the intractable likelihood ratio with an ELBO-based surrogate:

$$\rho^{\mathrm{ELBO}}_{i,t}(\theta) = \frac{\exp\left(\mathrm{ELBO}_\theta(\mathbf{a}_{i,t} \mid o_{i,t}, c)\right)}{\exp\left(\mathrm{ELBO}_{\theta_{\mathrm{old}}}(\mathbf{a}_{i,t} \mid o_{i,t}, c)\right)}. \tag{13}$$

To estimate this ELBO ratio, we follow FPO [McAllister et al., 2025] and use the conditional flow-matching objective as a likelihood-free surrogate. For a fixed executed action chunk and conditioning context, this objective is parameter-equivalent to the negative ELBO up to a term independent of θ:

$$\mathcal{L}_{\mathrm{FM},\theta}\left(\mathbf{a}_{i,t} \mid o_{i,t}, c\right) = -\mathrm{ELBO}_\theta\left(\mathbf{a}_{i,t} \mid o_{i,t}, c\right) + C_{i,t}, \tag{14}$$

where $C_{i,t}$ does not depend on the policy parameters. Therefore, the ELBO ratio can be estimated by the difference between the old and current flow-matching losses.

For each optimized action chunk $\mathbf{a}_{i,t}$, we sample N_{mc} flow timestep-noise pairs $\{(\tau_n, \boldsymbol{\epsilon}_n)\}_{n=1}^{N_{\mathrm{mc}}}$, where $\tau_n \sim \mathcal{U}(0,1)$ and $\boldsymbol{\epsilon}_n \sim \mathcal{N}(0, I)$. Using the same linear interpolation path as the action-flow process, we construct

$$\mathbf{a}^{\tau_n}_{i,t} = \tau_n \mathbf{a}_{i,t} + (1-\tau_n)\boldsymbol{\epsilon}_n, \quad \mathbf{u}^{(n)}_{i,t} = \mathbf{a}_{i,t} - \boldsymbol{\epsilon}_n. \tag{15}$$

The Monte Carlo flow-matching loss is

$$\widehat{\mathcal{L}}_{\mathrm{FM},\theta}\left(\mathbf{a}_{i,t} \mid o_{i,t}, c\right) = \frac{1}{N_{\mathrm{mc}}}\sum_{n=1}^{N_{\mathrm{mc}}}\left\| v_\theta\left(\mathbf{a}^{\tau_n}_{i,t}, \tau_n, o_{i,t}, c\right) - \mathbf{u}^{(n)}_{i,t}\right\|_2^2. \tag{16}$$

We use the same timestep-noise pairs for the old and current policies. The resulting surrogate ratio is

$$\widehat{\rho}^{\mathrm{FPO}}_{i,t}(\theta) = \exp\left(\widehat{\mathcal{L}}_{\mathrm{FM},\theta_{\mathrm{old}}}\left(\mathbf{a}_{i,t} \mid o_{i,t}, c\right) - \widehat{\mathcal{L}}_{\mathrm{FM},\theta}\left(\mathbf{a}_{i,t} \mid o_{i,t}, c\right)\right). \tag{17}$$

We substitute $\rho_{i,t}(\theta)$ in Eq. (11) with $\widehat{\rho}^{\mathrm{FPO}}_{i,t}(\theta)$. The resulting objective applies each branch-level advantage to the executed clean action chunks in that branch, without evaluating likelihoods along the denoising path.

Table 1: Success rates on the LIBERO benchmark. We report results on Spatial, Object, Goal, and Long tasks, together with the average success rate. Δ Avg. denotes the improvement over the corresponding SFT baseline.

Model		Spatial	Object	Goal	Long	Avg.	Δ Avg.
# Full Dataset SFT							
Octo [Team et al., 2024]		78.9	85.7	84.6	51.1	75.1	–
OpenVLA [Kim et al., 2024]		84.7	88.4	79.2	53.7	76.5	–
π_{fast} [Pertsch et al., 2025]		96.4	96.8	88.6	60.2	85.5	–
OpenVLA-OFT [Kim et al., 2025]		91.6	95.3	90.6	86.5	91.0	–
π_0 [Black et al., 2024]		96.8	98.8	95.8	85.2	94.2	–
$\pi_{0.5}$ [Black et al., 2025]		98.8	98.2	98.0	92.4	96.9	–
# Few-shot SFT + RL							
π_0	SFT	65.3	64.4	49.8	51.2	57.6	–
	π_{RL} (Flow-SDE + PPO)	98.4	99.4	96.2	90.2	96.0	+38.4
	π_{RL} (Flow-SDE + GRPO)	97.8	97.8	83.2	81.4	90.0	+32.4
	π-StepNFT	93.5	98.0	83.7	86.7	90.5	+32.9
	LiFT	98.0	98.4	92.7	89.5	94.7	+37.1
# Few-shot SFT + RL							
$\pi_{0.5}$	SFT	84.6	95.4	84.6	43.9	77.1	–
	π_{RL} (Flow-SDE + PPO)	99.6	100.0	98.8	93.0	97.9	+20.8
	π_{RL} (Flow-SDE + GRPO)	97.4	99.8	91.2	77.6	91.5	+14.4
	π-StepNFT	97.8	100.0	98.2	79.8	94.0	+16.9
	LiFT	98.7	99.8	98.5	88.4	96.4	+19.3

4 Experiments

4.1 Experimental Setup

Benchmarks and evaluation. We evaluate LiFT on LIBERO [Liu et al., 2023] and ManiSkill [Mu et al., 2021]. For LIBERO, we report success rates on the Spatial, Object, Goal, and Long suites, together with the average success rate. For ManiSkill, we evaluate both in-distribution and out-of-distribution generalization, with OOD results grouped into Vision, Semantic, and Execution shifts. All results are task success rates, and Δ denotes the absolute improvement over the corresponding SFT initialization.

SFT initialization. Following π-StepNFT [Wang et al., 2026], we use few-shot SFT initializations to avoid performance saturation on LIBERO: π_0 is trained on 58 demonstrations for Spatial, Object, and Goal, and 208 demonstrations for Long, while $\pi_{0.5}$ uses a unified few-shot set of 40 demonstrations, one per subtask. On ManiSkill, we use the full SFT initialization with 16,384 demonstrations.

Implementation details. LiFT is instantiated on two flow-based VLA backbones, π_0 and $\pi_{0.5}$. We freeze the VLM backbone and fine-tune only the action expert in all experiments. Experiments are run on 8 NVIDIA A100 80GB GPUs, with additional hyperparameters reported in Appendix D.

4.2 Main Results on LIBERO

Table 1 reports LIBERO success rates under full-dataset SFT and few-shot SFT + RL settings. Starting from few-shot SFT, LiFT improves the average success rate from 57.6% to 94.7% on π_0, and from 77.1% to 96.4% on $\pi_{0.5}$, corresponding to absolute gains of $+37.1$ and $+19.3$ percentage points. These results show that LiFT can effectively improve flow-based VLA policies beyond their supervised initialization.

LiFT also outperforms critic-free online RL baselines. On π_0, it improves over π-StepNFT by $+4.2$ percentage points on average and by $+9.0$ percentage points on Goal, and improves over Flow-SDE + GRPO by $+4.7$ percentage points on average. The gains are especially pronounced on LIBERO-Long, where LiFT reaches 89.5% with π_0 and 88.4% with $\pi_{0.5}$, outperforming Flow-SDE + GRPO by $+8.1$ and $+10.8$ percentage points, respectively. Since LIBERO-Long requires long-horizon

Table 2: Comparison on in-distribution (IND) and out-of-distribution (OOD) evaluations. Δ denotes the improvement over the corresponding SFT baseline.

Model		IND		OOD				
		Avg.	**Δ**	**Vision**	**Semantic**	**Execution**	**Avg.**	**Δ**
π_0	SFT	38.4	–	32.6	8.4	13.2	18.1	–
	π_{RL} (Flow-SDE + PPO)	78.8	+40.4	61.1	25.4	31.5	39.3	+21.2
	π-StepNFT	79.2	+40.8	69.1	49.1	33.1	50.4	+32.3
	LiFT	**82.6**	**+44.2**	**74.4**	**51.0**	**37.8**	**54.4**	**+36.3**
$\pi_{0.5}$	SFT	40.1	–	40.2	16.6	22.4	26.4	–
	π_{RL} (Flow-SDE + PPO)	**90.9**	**+50.8**	68.0	34.5	45.4	49.3	+22.9
	π-StepNFT	85.4	+45.3	76.9	56.6	45.1	59.5	+33.1
	LiFT	89.3	+49.2	**83.4**	**59.5**	**48.7**	**63.9**	**+37.5**

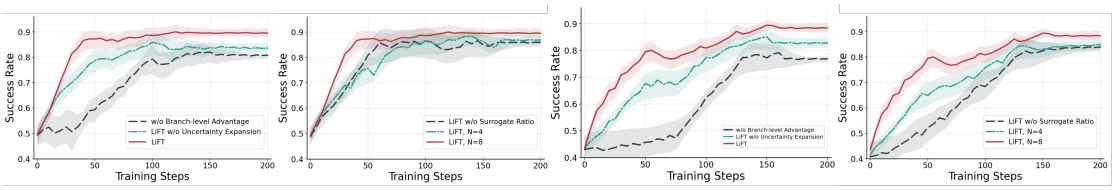

Figure 3: Ablation curves on LIBERO. The two left plots use π_0, and the two right plots use $\pi_{0.5}$. LiFT is compared with variants removing branch-level advantages, uncertainty-guided branching, or the surrogate ratio, as well as different Monte Carlo sample numbers N_{mc}.

execution under sparse terminal rewards, this improvement supports the benefit of propagating terminal outcomes through shared-prefix branches for more precise credit assignment.

Compared with PPO-based optimization, LiFT achieves competitive average performance without training an explicit critic and without evaluating exact likelihoods along the flow sampling path. This demonstrates that LiFT is effective for online policy optimization of flow-based VLA action chunks without requiring an explicit critic or exact flow-path likelihoods.

4.3 Main Results on ManiSkill

Table 2 evaluates whether LiFT generalizes beyond the training distribution under visual, semantic, and execution shifts. LiFT improves both in-distribution and out-of-distribution performance across the two flow-based VLA backbones. With π_0, LiFT achieves 82.6% IND success and 54.4% OOD average success, outperforming SFT by $+44.2$ and $+36.3$ percentage points, respectively. It also exceeds π-StepNFT by $+4.0$ percentage points on OOD average, with strong gains under Vision and Execution shifts.

With $\pi_{0.5}$, Flow-SDE + PPO obtains the highest IND success rate, but its OOD performance drops substantially. In contrast, LiFT achieves the best OOD average success rate of 63.9%, outperforming Flow-SDE + PPO by $+14.6$ percentage points and π-StepNFT by $+4.4$ percentage points. This suggests that LiFT does not simply overfit to the training distribution, but produces updates that transfer more robustly across shifted evaluation settings.

4.4 Ablation Studies

We ablate the main components of LiFT on LIBERO with both π_0 and $\pi_{0.5}$. Table 3 reports final success rates, while Figure 3 shows the corresponding training dynamics. The results show that each component contributes to both final performance and optimization efficiency.

Removing branch-level advantage propagation leads to the largest degradation. The average success rate drops from 94.7% to 89.6% on π_0, and from 96.4% to 90.6% on $\pi_{0.5}$. The drop is especially clear on LIBERO-Long, where performance decreases by 8.7 and 11.5 percentage points, respectively. This indicates that, under LiFT's tree-structured rollout setting, propagating terminal outcomes through shared-prefix branches is crucial for more precise action-chunk credit assignment under sparse rewards.

Table 3: Ablation study of LiFT with π_0 and $\pi_{0.5}$ on LIBERO. We report success rates on Spatial, Object, Goal, and Long tasks, together with the average success rate.

Model	Ablation	Spatial	Object	Goal	Long	Avg.
π_0	w/o Uncertainty-guided Expansion	96.8	97.2	86.9	83.6	91.1
	w/o Branch-level Advantage	95.9	96.5	85.3	80.8	89.6
	w/o FPO Ratio	97.5	97.9	88.0	85.9	92.3
	LiFT	**98.0**	**98.4**	**92.7**	**89.5**	**94.7**
$\pi_{0.5}$	w/o Uncertainty-guided Expansion	97.8	98.4	96.7	82.8	93.9
	w/o Branch-level Advantage	97.1	97.8	90.5	76.9	90.6
	w/o FPO Ratio	98.3	98.8	96.4	84.0	94.4
	LiFT	**98.7**	**99.8**	**98.5**	**88.4**	**96.4**

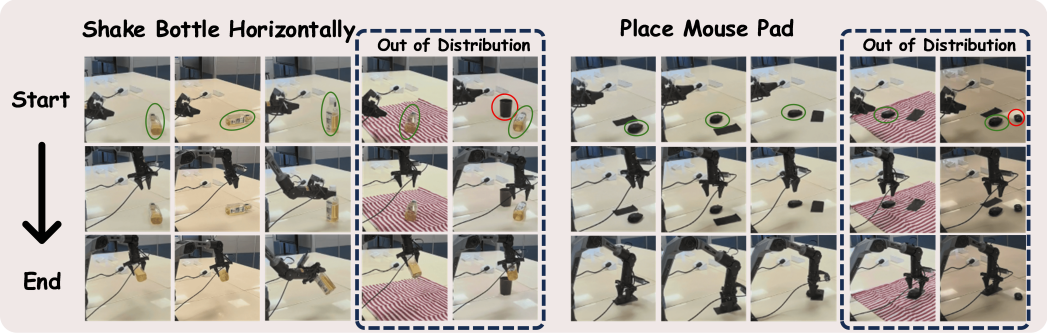

Figure 4: Real-world manipulation examples under standard and out-of-distribution settings. Green circles mark target objects, and red circles mark distractors. Each column shows a representative execution sequence from start to end.

Uncertainty-guided expansion also provides consistent gains. Replacing it with uniform expansion reduces the average success rate by 3.6 percentage points on π_0 and 2.5 percentage points on $\pi_{0.5}$. In Figure 3, this variant follows the full method in the early stage but reaches a lower plateau, suggesting that uniform expansion can still increase rollout diversity but is less effective at selecting informative branching states.

The FPO surrogate ratio is also necessary for stable policy improvement. Removing it reduces the average success rate from 94.7% to 92.3% on π_0, and from 96.4% to 94.4% on $\pi_{0.5}$. The learning curves show that the variant without the surrogate ratio is slower and more unstable, especially in the early and middle stages of training. This supports the need to align branch-level advantages with a decision-level policy ratio on executed action chunks, rather than applying the sparse-reward signal without a proper likelihood-free ratio estimate. We also vary the Monte Carlo sample number N_{mc} used for FPO ratio estimation. As shown in Figure 3, $N_{\text{mc}} = 8$ learns faster and converges higher than $N_{\text{mc}} = 4$, suggesting that more samples yield a more stable surrogate ratio estimate.

4.5 Real-World Experiments

We also conduct real-world experiments to qualitatively validate LiFT beyond simulation. As shown in Figure 4, LiFT can be deployed on a physical robot arm to perform language-conditioned manipulation tasks in real environments. We leave the experimental details to Appendix E.

5 Conclusion, Limitation and Future Works

We presented LiFT, a likelihood-free tree-structured policy optimization framework for flow-based VLA models. LiFT improves sparse-reward online RL by constructing shared-prefix tree roll-

outs, propagating terminal outcomes into branch-level advantages, and applying these advantages through a flow-matching surrogate ratio without computing exact action likelihoods. Experiments on LIBERO and ManiSkill show that LiFT improves few-shot fine-tuning, sample efficiency, and out-of-distribution robustness across two flow-based VLA backbones. We further conduct real-world experiments to qualitatively validate LiFT beyond simulation.

LiFT still has several limitations. Its uncertainty score is only an approximate proxy for downstream outcome diversity, and may not always identify the most informative branching states. In addition, although shared prefixes amortize rollout cost, tree expansion and uncertainty probing still require extra model forward passes and simulator state management. Future work will explore more calibrated branching criteria, lower-overhead expansion, and larger-scale real-world deployment.

Acknowledgments and Disclosure of Funding

The authors would like to thank the Intelligent Cybernetics Lab (ICLab) at Nanyang Technological University for its research support. The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg). The authors declare no competing interests.

References

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.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: A Vision-Language-Action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

Lang Cao, Hui Ruan, Yongqian Li, Peng Chao, Wu Ning, Haonan Song, Renhong Chen, and Yitong Li. Treeadv: Tree-structured advantage redistribution for group-based rl. *arXiv preprint arXiv:2601.03703*, 2026.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2025a.

Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. Sarm: Stage-aware reward modeling for long horizon robot manipulation. *arXiv preprint arXiv:2509.25358*, 2025b.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. *arXiv preprint arXiv:2502.05450*, 2025c.

Yuqi Chen, Junjie Gao, Yongzhou Pan, Siyuan Song, Zixuan Zhang, Jiaping Xiao, and Mir Feroskhan. Geninav: Generative model driven image-goal navigation via imagination-guided consistency flow matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 996–1005, 2026.

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, Li Ji, Ao Li, Shiduo Zhang, Liming Liu, Jinlong Hou, Jingjing Gong, Xianzhong Zhao, and Xipeng Qiu. Srpo: Self-referential policy optimization for vision-language-action models. *arXiv preprint arXiv:2511.15605*, 2025a.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 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.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. Treerl: Llm reinforcement learning with on-policy tree search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12355–12369, 2025.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $pi^*_{0.6}$: A VLA That Learns From Experience. *arXiv preprint arXiv:2511.14759*, 2025.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 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.

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, Yiming Cheng, Miles Yang, Zhao Zhong, and Liefeng Bo. Mixgrpo: Unlocking flow-based grpo efficiency with mixed ode-sde. *arXiv preprint arXiv:2507.21802*, 2025b.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, et al. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint arXiv:2508.17445*, 2025c.

Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, et al. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025d.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

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.

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.

Yutong Luo, Jianing Xu, Yongzhou Chen, Yinuo Yang, Tianyao Zheng, and Hao Liu. Optimal hovering control strategy for uavs using ppo. In *2025 7th International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 247–251. IEEE, 2025.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025.

Qi Lv, Jianming Xing, Zhao Yang, Mingyuan Yao, Yinan Shi, Yawei Jueluo, Mike Zheng Shou, and Xiang Deng. Hermite curves as trajectory priors for vision-language-action models. *arXiv preprint arXiv:2608.01265*, 2026.

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.

Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. *arXiv preprint arXiv:2107.14483*, 2021.

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)*, pages 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.

Yibo Qiu, Shu'ang Sun, Haoliang Ye, Ronald X Xu, and Mingzhai Sun. Keyframe-guided structured rewards for reinforcement learning in long-horizon laboratory robotics. *arXiv preprint arXiv:2603.00719*, 2026.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Yuheng Ji, Cheng Chi, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, et al. Robo-dopamine: General process reward modeling for high-precision robotic manipulation. *arXiv preprint arXiv:2512.23703*, 2025a.

Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025b.

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.

Siting Wang, Xiaofeng Wang, Zheng Zhu, Minnan Pei, Xinyu Cui, Cheng Deng, Jian Zhao, Guan Huang, Haifeng Zhang, and Jun Wang. π-StepNFT: Wider space needs finer steps in online rl for flow-based VLAs. *arXiv preprint arXiv:2603.02083*, 2026.

Feng Xu, Guangyao Zhai, Xin Kong, Tingzhong Fu, Daniel FN Gordon, Xueli An, and Benjamin Busam. Stare-vla: Progressive stage-aware reinforcement for fine-tuning vision-language-action models. *arXiv preprint arXiv:2512.05107*, 2025.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. Treerpo: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183*, 2025.

Hongzhi Zang, Mingjie Wei, Si Xu, Yongji Wu, Zhen Guo, Yuanqing Wang, Hao Lin, Liangzhi Shi, Yuqing Xie, Zhexuan Xu, et al. Rlinf-vla: A unified and efficient framework for vla+ rl training. *arXiv preprint arXiv:2510.06710*, 2025.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025a.

Yuhang Zhang, Chao Yan, Jiaping Xiao, and Mir Feroskhan. Npe-drl: Enhancing perception constrained obstacle avoidance with nonexpert policy guided reinforcement learning. *IEEE Transactions on Artificial Intelligence*, 6(1):184–198, 2024a.

Yuhang Zhang, Jiaping Xiao, and Mir Feroskhan. Learning cross-modal visuomotor policies for autonomous drone navigation. *IEEE Robotics and Automation Letters*, 10(6):5425–5432, 2025b.

Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. Grape: Generalizing robot policy via preference alignment. *arXiv preprint arXiv:2411.19309*, 2024b.

Zixuan Zhang, Yuqi Chen, Junjie Gao, Siyuan Song, Yongzhou Pan, Beichen Wang, and Mir Feroskhan. Roamflow: Reinforcement-aligned one-step action meanflow policy for image-goal navigation. *arXiv preprint arXiv:2606.29934*, 2026.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

A Preliminaries

A.1 Problem Formulation

We formulate each language-conditioned manipulation task, specified by a task condition c, as a finite-horizon MDP, defined by a tuple $\mathcal{M}_c = (\mathcal{S}, \mathcal{A}, P_0^c, P_{\text{env}}^c, R_{\text{env}}^c, \gamma)$. Here, $\mathcal{S}$ denotes the decision-state space, $\mathcal{A}$ denotes the action-chunk space, P_0^c is the initial-state distribution, P_{env}^c is the environment transition dynamics, R_{env}^c is the task reward function, and γ is the discount factor.

At decision step t, the state $s_t \in \mathcal{S}$ consists of the robot observation and proprioceptive information. Conditioned on s_t and the task condition c, the VLA policy predicts an action chunk

$$\mathbf{a}_t = [a_{t,0}, \ldots, a_{t,H-1}] \sim \pi_\theta(\cdot \mid s_t, c) \in \mathcal{A},$$

where H is the chunk horizon. The environment executes the action chunk as one macro-step, transitions to $s_{t+1} \sim P_{\text{env}}^c(\cdot \mid s_t, \mathbf{a}_t)$, and returns a chunk-level reward $R_{\text{env}}^c(s_t, \mathbf{a}_t)$. The objective is to learn a policy π_θ that maximizes the expected γ-discounted return over a horizon of $T+1$:

$$J(\pi_\theta; c) = \mathbb{E}_{\pi_\theta, P_0^c, P_{\text{env}}^c}\left[\sum_{t=0}^{T} \gamma^t R_{\text{env}}^c(s_t, \mathbf{a}_t)\right]. \tag{18}$$

With the policy-gradient surrogate, the gradient of the return objective can be estimated from sampled trajectories as

$$\nabla_\theta J(\pi_\theta; c) = \mathbb{E}_{\pi_\theta, P_0^c, P_{\text{env}}^c}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t \mid s_t, c) A_c^\pi(s_t, \mathbf{a}_t)\right]. \tag{19}$$

The advantage function, $A_c^\pi(s_t, \mathbf{a}_t) = Q_c^\pi(s_t, \mathbf{a}_t) - V_c^\pi(s_t)$, measures the relative merit of executing action chunk $\mathbf{a}_t$ at state s_t under task condition c, and provides a variance-reduced signal for policy-gradient updates.

A.2 Flow-Based VLA Policies

A flow-based VLA policy π_θ maps the current decision state and task condition to a sequence of H future actions, i.e.,

$$\mathbf{a}_t = [a_{t,0}, \ldots, a_{t,H-1}] \sim \pi_\theta(\cdot \mid s_t, c),$$

which is interpreted as an action chunk. The policy represents the conditional action-chunk distribution implicitly through a learned conditional velocity field v_θ. In practice, the visual observations, language instruction, and proprioceptive state are encoded as the conditioning input, while the flow matching action head generates the action chunk.

Given an action chunk $\mathbf{a}_t$, random noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, and flow time $\tau \in [0, 1]$, the conditional probability path used in flow matching constructs a noisy action chunk

$$\mathbf{a}_t^\tau = \tau \mathbf{a}_t + (1 - \tau)\boldsymbol{\epsilon}. \tag{20}$$

For this linear path, the corresponding ground-truth vector field is

$$\mathbf{u}(\mathbf{a}_t^\tau \mid \mathbf{a}_t) = \mathbf{a}_t - \boldsymbol{\epsilon}. \tag{21}$$

The conditional flow-matching objective trains v_θ to match this ground-truth vector field:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{\tau, \boldsymbol{\epsilon}, (\mathbf{a}_t, s_t, c)}\left[\left\| v_\theta(\mathbf{a}_t^\tau, \tau, s_t, c) - \mathbf{u}(\mathbf{a}_t^\tau \mid \mathbf{a}_t)\right\|_2^2\right]. \tag{22}$$

During inference, the policy first samples an initial noise vector $\mathbf{a}_t^0 \sim \mathcal{N}(0, I)$, and then iteratively refines it by integrating the learned velocity field. With a forward Euler discretization and step size δ, the update is

$$\mathbf{a}_t^{\tau+\delta} = \mathbf{a}_t^\tau + v_\theta(\mathbf{a}_t^\tau, \tau, s_t, c) \cdot \delta. \tag{23}$$

After integrating from $\tau = 0$ to $\tau = 1$, the final sample $\mathbf{a}_t^1$ is used as the executable action chunk.

B Variance Reduction of Tree-Structured Branch Advantages

This section provides a variance-reduction argument for LiFT's tree-structured branch advantage estimator. Averaging multiple continuations from the same branching node reduces the conditional variance of the backed-up branch advantage compared with using a single continuation.

Consider a branch b that terminates at a branching node u_b in tree T_i. Suppose u_b has m outgoing child branches, denoted by $b_1, \ldots, b_m$. Following the backup rule in Sec. 3.2, LiFT assigns branch b the uniformly backed-up advantage

$$\hat{A}_{\text{branch},i}(b) = \frac{1}{m} \sum_{j=1}^{m} \hat{A}_{\text{branch},i}(b_j). \tag{24}$$

We analyze the conditional variance of this estimator given the shared prefix ending at u_b. Let

$$\hat{A}_{\text{branch},i}(b_j) = A_j^\star + \varepsilon_j, \qquad \mathbb{E}[\varepsilon_j \mid u_b] = 0, \qquad \text{Var}(\varepsilon_j \mid u_b) = \sigma_j^2. \tag{25}$$

where $A_j^\star$ is the conditional expected backed-up value of child branch b_j, and ε_j captures downstream rollout noise and terminal reward noise. We assume that, conditioned on the shared prefix at u_b, the residual noises $\{\varepsilon_j\}_{j=1}^m$ are independent. This assumption is applied to the post-normalized branch-return estimates, or equivalently conditional on the tree-level normalization statistics.

Define the local target of the uniform backup as the average child-branch value

$$\bar{A}^\star(u_b) = \frac{1}{m} \sum_{j=1}^{m} A_j^\star. \tag{26}$$

Proposition B.1. Under the conditional independence assumption above, the uniformly backed-up LiFT estimator in Eq. (24) is an unbiased estimator of $\bar{A}^\star(u_b)$, and its conditional variance is

$$\text{Var}\left[\hat{A}_{\text{branch},i}(b) \mid u_b\right] = \frac{1}{m^2} \sum_{j=1}^{m} \sigma_j^2. \tag{27}$$

In the equal-variance case $\sigma_j^2 = \sigma_{\text{env}}^2$, this becomes

$$\text{Var}\left[\hat{A}_{\text{branch},i}(b) \mid u_b\right] = \frac{\sigma_{\text{env}}^2}{m}. \tag{28}$$

Therefore, using all m sibling continuations reduces the conditional rollout variance by a factor of m compared with using one child continuation to estimate the same local average target.

Proof. First, by linearity of expectation,

$$\begin{aligned} \mathbb{E}\left[\hat{A}_{\text{branch},i}(b) \mid u_b\right] &= \mathbb{E}\left[\frac{1}{m} \sum_{j=1}^{m} \hat{A}_{\text{branch},i}(b_j) \mid u_b\right] \\ &= \frac{1}{m} \sum_{j=1}^{m} \mathbb{E}\left[A_j^\star + \varepsilon_j \mid u_b\right] \\ &= \frac{1}{m} \sum_{j=1}^{m} A_j^\star = \bar{A}^\star(u_b). \end{aligned} \tag{29}$$

Thus, the uniform tree backup is unbiased for the local average child-branch target.

Next, by conditional independence of the residual noises,

$$\begin{aligned}
\mathrm{Var}\left[\hat{A}_{\mathrm{branch},i}(b) \mid u_b\right] &= \mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{A}_{\mathrm{branch},i}(b_j) \mid u_b\right] \\
&= \mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\varepsilon_j \mid u_b\right] \\
&= \frac{1}{m^2}\sum_{j=1}^{m}\mathrm{Var}\left[\varepsilon_j \mid u_b\right] \\
&= \frac{1}{m^2}\sum_{j=1}^{m}\sigma_j^2.
\end{aligned} \tag{30}$$

If all child branches have the same residual variance $\sigma_j^2 = \sigma_{\mathrm{env}}^2$, then

$$\mathrm{Var}\left[\hat{A}_{\mathrm{branch},i}(b) \mid u_b\right] = \frac{1}{m^2} m\sigma_{\mathrm{env}}^2 = \frac{\sigma_{\mathrm{env}}^2}{m}. \tag{31}$$

For comparison, using a single continuation preserves the full conditional residual variance σ_{env}^2. Therefore, in the equal-variance case, the uniform branch backup reduces the conditional downstream rollout variance by a factor of m compared with using a single continuation. This proves the claim. □

Implication for LiFT. Trajectory-level GRPO assigns the same normalized terminal advantage to every optimized action chunk on a completed trajectory. This makes the update sensitive to accidental downstream successes or failures. LiFT instead branches from a shared prefix, compares sibling continuations, and backs up their normalized outcomes to the branch level. The uniform backup in Eq. (24) averages over downstream rollout noise within the local subtree, producing a lower-variance learning signal for the action chunks inside the incoming branch.

C Applicability of FPO Ratios to Flow-SDE Rollouts

LiFT uses Flow-SDE sampling to generate exploratory rollout branches, while its policy update applies an FPO-style surrogate ratio to the terminal action chunk. This section justifies this choice under the continuous-time marginal preservation property of Flow-SDE.

Fix a decision state s_t, task condition c, and policy parameter θ. Let $\pi_\theta(\mathbf{a}_t \mid s_t, c)$ denote the terminal action-chunk density induced by the original flow ODE sampler. Let $p_{\theta,1}^{\mathrm{SDE}}(\mathbf{a}_t \mid s_t, c)$ denote the terminal density induced by the corresponding Flow-SDE sampler at $\tau = 1$. Under the exact ODE-to-SDE construction, with the correct score–velocity relation and continuous-time integration, Flow-SDE preserves the terminal marginal distribution:

$$p_{\theta,1}^{\mathrm{SDE}}(\mathbf{a}_t \mid s_t, c) = \pi_\theta(\mathbf{a}_t \mid s_t, c). \tag{32}$$

Proposition C.1. Assume that Eq. (32) holds for both θ and θ_{old}, and that the policy update uses an FPO-compatible weighted flow-matching loss whose expectation is the negative ELBO up to a constant independent of θ. Then the FPO-style ratio computed on the terminal action chunk collected with Flow-SDE is a valid ELBO-ratio surrogate for the terminal action-density ratio.

Proof. During rollout collection, the terminal action chunk is sampled from the old Flow-SDE sampler according to

$$\mathbf{a}_t \sim p_{\theta_{\mathrm{old}},1}^{\mathrm{SDE}}(\cdot \mid s_t, c).$$

By the marginal preservation assumption,

$$p_{\theta_{\mathrm{old}},1}^{\mathrm{SDE}}(\mathbf{a}_t \mid s_t, c) = \pi_{\theta_{\mathrm{old}}}(\mathbf{a}_t \mid s_t, c), \qquad p_{\theta,1}^{\mathrm{SDE}}(\mathbf{a}_t \mid s_t, c) = \pi_\theta(\mathbf{a}_t \mid s_t, c). \tag{33}$$

Therefore, the terminal density ratio induced by Flow-SDE equals the original flow-policy terminal action ratio:

$$\begin{aligned}\rho_t^{\mathrm{SDE}}(\theta) &:= \frac{p_{\theta,1}^{\mathrm{SDE}}(\mathbf{a}_t \mid s_t, c)}{p_{\theta_{\mathrm{old}},1}^{\mathrm{SDE}}(\mathbf{a}_t \mid s_t, c)} \\ &= \frac{\pi_\theta(\mathbf{a}_t \mid s_t, c)}{\pi_{\theta_{\mathrm{old}}}(\mathbf{a}_t \mid s_t, c)} = \rho_t(\theta).\end{aligned} \tag{34}$$

The exact density ratio $\rho_t(\theta)$ is intractable for flow-based policies. Following FPO, we replace it with an ELBO-ratio surrogate:

$$\rho_t^{\mathrm{ELBO}}(\theta) := \frac{\exp\left(\mathrm{ELBO}_\theta(\mathbf{a}_t \mid s_t, c)\right)}{\exp\left(\mathrm{ELBO}_{\theta_{\mathrm{old}}}(\mathbf{a}_t \mid s_t, c)\right)}. \tag{35}$$

Equivalently, since

$$\mathrm{ELBO}_\theta(\mathbf{a}_t \mid s_t, c) = \log \pi_\theta(\mathbf{a}_t \mid s_t, c) - D_{\mathrm{KL},\theta}^{\mathrm{gap}}(\mathbf{a}_t, s_t, c), \tag{36}$$

the ELBO ratio can be written as

$$\rho_t^{\mathrm{ELBO}}(\theta) = \rho_t(\theta) \exp\left(D_{\mathrm{KL},\theta_{\mathrm{old}}}^{\mathrm{gap}} - D_{\mathrm{KL},\theta}^{\mathrm{gap}}\right). \tag{37}$$

Thus, the ELBO ratio is a variational surrogate of the terminal action-density ratio, rather than an exact PPO likelihood ratio.

Let $\mathcal{L}_{\mathrm{FM},\theta}^w$ denote the FPO-compatible weighted flow-matching loss. Under the FPO loss–ELBO relation, for a fixed terminal action chunk,

$$\mathcal{L}_{\mathrm{FM},\theta}^w(\mathbf{a}_t \mid s_t, c) = -\mathrm{ELBO}_\theta(\mathbf{a}_t \mid s_t, c) + C_t, \tag{38}$$

where C_t is independent of θ. Substituting Eq. (38) into Eq. (35) gives

$$\begin{aligned}\rho_t^{\mathrm{ELBO}}(\theta) &= \exp\left(\mathrm{ELBO}_\theta(\mathbf{a}_t \mid s_t, c) - \mathrm{ELBO}_{\theta_{\mathrm{old}}}(\mathbf{a}_t \mid s_t, c)\right) \\ &= \exp\left(\mathcal{L}_{\mathrm{FM},\theta_{\mathrm{old}}}^w(\mathbf{a}_t \mid s_t, c) - \mathcal{L}_{\mathrm{FM},\theta}^w(\mathbf{a}_t \mid s_t, c)\right).\end{aligned} \tag{39}$$

In practice, LiFT estimates the weighted flow-matching losses with shared Monte Carlo timestep-noise samples $\{(\tau_i, \boldsymbol{\epsilon}_i)\}_{i=1}^{N_{\mathrm{mc}}}$:

$$\widehat{\rho}_t^{\mathrm{FPO}}(\theta) = \exp\left[\frac{1}{N_{\mathrm{mc}}} \sum_{i=1}^{N_{\mathrm{mc}}} \left(\ell_{\theta_{\mathrm{old}}}^w(\tau_i, \boldsymbol{\epsilon}_i; \mathbf{a}_t, s_t, c) - \ell_\theta^w(\tau_i, \boldsymbol{\epsilon}_i; \mathbf{a}_t, s_t, c)\right)\right], \tag{40}$$

where the same $(\tau_i, \boldsymbol{\epsilon}_i)$ are reused for the old and updated policies. Therefore, under exact terminal marginal preservation and an FPO-compatible weighted flow-matching objective, the FPO-style ratio can be applied to terminal action chunks collected with Flow-SDE.

With finite-step Flow-SDE or hybrid ODE-SDE rollout implementations, the marginal preservation in Eq. (32) is only approximate. Accordingly, Eq. (40) should be understood as an approximate FPO-style surrogate rather than an exact path-level PPO likelihood ratio.

D Hyperparameters for Training

Table 4 summarizes the hyperparameters used in our simulation experiments. For LIBERO, each update collects $64 \times 8 = 512$ completed trajectories, while ManiSkill collects $320 \times 1 = 320$ trajectories per update. For the S/O/G columns, slash-separated values denote the settings for Spatial, Object, and Goal, respectively.

For LiFT, the rollout budget is kept the same as the corresponding baseline. Instead of collecting independent trajectories only, LiFT organizes the same number of completed trajectories as leaves of rollout trees. Each tree context contains $M = 4$ prefix roots and three binary branching points, resulting in $4 \times 2^3 = 32$ leaf trajectories per context. Thus, LIBERO uses 16 tree contexts per update and ManiSkill uses 10 tree contexts per update.

Table 4: Training hyperparameters for simulation experiments. We report the settings used for LIBERO and ManiSkill experiments with π_0 and $\pi_{0.5}$.

Hyperparameter	LIBERO				ManiSkill	
	π_0 S/O/G	π_0 Long	$\pi_{0.5}$ S/O/G	$\pi_{0.5}$ Long	π_0 Multitask	$\pi_{0.5}$ Multitask
# Training setup						
Train epochs / global updates	400	400	400	400	240	240
Parallel environments	64	64	64	64	320	320
Rollout epochs per update	8	8	8	8	1	1
Final trajectories per update	512	512	512	512	320	320
Update epochs	2/2/4	4	1/1/3	4	4	5
# LiFT tree rollout						
Tree leaf budget per update	512	512	512	512	320	320
Tree contexts per update	16	16	16	16	10	10
Leaves per tree context	32	32	32	32	32	32
Prefix roots per context M	4	4	4	4	4	4
Branch width N	2	2	2	2	2	2
Number of branching windows	3	3	3	3	3	3
Branching rule	uncertainty	uncertainty	uncertainty	uncertainty	uncertainty	uncertainty
Uniform ablation rule	evenly spaced	evenly spaced	evenly spaced	evenly spaced	evenly spaced	evenly spaced
Train shared-prefix chunks	yes	yes	yes	yes	yes	yes
Branch advantage backup	subtree mean	subtree mean	subtree mean	subtree mean	subtree mean	subtree mean
Advantage normalization	per tree	per tree	per tree	per tree	per tree	per tree
# Uncertainty-guided branching						
Probe samples K	4	4	4	4	4	4
Probe time τ	0.5	0.5	0.5	0.5	0.5	0.5
Uncertainty score	velocity var.	velocity var.	velocity var.	velocity var.	velocity var.	velocity var.
Expansion selection	windowed threshold	windowed threshold	windowed threshold	windowed threshold	windowed threshold	windowed threshold
# Surrogate ratio						
Policy ratio	surrogate	surrogate	surrogate	surrogate	surrogate	surrogate
MC samples N_{mc}	8	8	8	8	8	8
timestep sampling	uniform	uniform	uniform	uniform	uniform	uniform
loss	MSE	MSE	MSE	MSE	MSE	MSE
Ratio clipping	0.2	0.2	0.2	0.2	0.2	0.2

For uncertainty-guided branching, we use a window-constrained threshold rule. The action-chunk horizon is divided into $W = 3$ temporal windows.

The surrogate policy ratio is estimated with $N_{\text{mc}} = 8$ Monte Carlo flow-matching samples. All LiFT variants use the same rollout budget and optimizer settings as the corresponding baselines. Only the branching rule, branch-level advantage computation, or surrogate ratio is changed in the ablation studies.

E Additional Details on Real-World Evaluation

Setup. We conduct real-world experiments to evaluate the sim-to-real effectiveness of LiFT on physical robot execution. We select two single-arm RoboTwin tasks: *Shake Bottle Horizontally* and *Place Mouse Pad*. For each task, we first use 1000 simulation trajectories for supervised fine-tuning. We then apply LiFT on the SFT model using 1000 simulation scenarios to obtain the RL-enhanced policy. The entire training process uses only simulation data, without any real-world demonstrations, real-world fine-tuning, or real-world RL.

The real-world experiments are conducted with one AgileX PiPER robotic arm equipped with a two-finger gripper. Visual observations are captured by Intel RealSense D435i cameras, and the policy receives RGB observations together with robot proprioceptive states. We manually construct the physical workspace to match the corresponding RoboTwin simulation scenes, including the robot-table configuration, task-relevant objects, target regions, camera viewpoints, and language instructions. The same action interface and language-conditioned policy used in simulation are directly used for real-world deployment.

We evaluate both the SFT and LiFT policies in the real world under three settings. The *Original* setting uses a clean tabletop scene with unseen real-world backgrounds. *Disturbance I* adds irrelevant objects to the tabletop while keeping the task-relevant objects unchanged. *Disturbance II* changes the visual background or tabletop appearance. For each task and setting, each policy is tested with 50 real-world trials, and the success rate is computed as the percentage of successful trials.

Evaluation protocol. For *Shake Bottle Horizontally*, a trial is counted as successful if the robot grasps the bottle and shakes it horizontally according to the task instruction without dropping the object. For *Place Mouse Pad*, a trial is counted as successful if the robot places the mouse pad

Table 5: Real-world success rates on two RoboTwin-style tasks. Disturbance I denotes additional irrelevant objects placed in the scene, while Disturbance II denotes background perturbation.

Model	Method	*Shake Bottle Horizontally*				*Place Mouse Pad*			
		Original	Dist. I	Dist. II	Avg.	Original	Dist. I	Dist. II	Avg.
$\pi_{0.5}$	SFT	40.0	38.0	30.0	36.0	60.0	50.0	44.0	51.3
	LiFT	66.0	54.0	52.0	57.3	72.0	70.0	66.0	69.3
	Δ	+26.0	+16.0	+22.0	+21.3	+12.0	+20.0	+22.0	+18.0

into the target region with the correct final pose. Between trials, the scene is manually reset to the corresponding initial configuration.



Figure 5: Real-world evaluation setup. We build physical tabletop scenes corresponding to selected RoboTwin simulation tasks and evaluate sim-to-real transfer under original and perturbed real-world configurations.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are consistent with the paper's contributions, including the LiFT framework, likelihood-free policy update, and empirical evaluation on robotic manipulation benchmarks.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The analysis of limitations is provided in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The proofs are provided in the supplementary materials.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: These details are provided in the supplementary materials.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will release the code in the future.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: These details are provided in Section 4 and supplementary materials.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We clarify the statistical significance of the experiments in the experimental section of the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We introduce the computational resources used to carry out the experiments

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research complies with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification :The paper primarily focuses on RL algorithm design and theoretical policy optimization for flow-based VLA policies, so we do not identify direct societal impacts beyond general robot learning research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk data or models, so no additional safeguards are required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We acknowledge and cite the original works for the pretrained VLA models and benchmarks used in this study. We respect the applicable license terms, including OpenPI code (Apache-2.0, with additional Gemma terms for relevant model components), LIBERO code and data (MIT and CC BY 4.0), ManiSkill code and assets (Apache-2.0 and CC BY-NC 4.0), and RoboTwin code (MIT).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are released in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper contains no crowdsourcing or research involving human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper contains no research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.