π-StepNFT: Wider Space Needs Finer Steps in Online RL for Flow-based VLAs

Anonymous authors
Paper under double-blind review

Abstract

Flow-based vision-language-action (VLA) models have emerged as a powerful paradigm for embodied control, but multi-step denoising makes policy-gradient online reinforcement learning (RL) difficult because the final-action likelihood induced by the sampler is not directly tractable. We identify an exploration–supervision mismatch within the denoising trajectory, where stochastic differential equation (SDE) sampling explores a wider local stochastic transition space while supervision derived from the final denoised output does not directly use the adjacent transitions observed during sampling. We propose ***π-StepNFT*** (Step-wise Negative-aware Fine-Tuning), a flow-native online RL framework that couples wider stochastic exploration with finer step-wise supervision without computing the final-action likelihood. π-StepNFT treats observed SDE transitions as supervision units, using variance-normalized transition errors to rank mirrored positive and negative velocity branches and map rollout feedback to local velocity updates. On LIBERO, starting from shared few-shot SFT initializations, π-StepNFT raises average success from 57.6% to 92.0% for π_0 and from 77.1% to 94.0% for $\pi_{0.5}$. On ManiSkill, it achieves the highest reported mean success among the evaluated RL baselines on five test-only visual perturbations.

1 Introduction

Vision-language-action (VLA) models have recently shown increasingly general capabilities, enabling robots to follow open-ended natural language instructions and perform complex tasks across diverse environments. Early approaches either represented actions as discrete tokens (Zitkovich et al., 2023; Kim et al., 2024) or mapped observations to continuous regression targets (Kim et al., 2025b). Recent work increasingly adopts flow-matching policies (Black et al., 2026; 2025; Intelligence et al., 2025; Ye et al., 2025; Bjorck et al., 2025), which combine large-scale vision-language pretraining with generative action prediction and have become a strong paradigm for complex manipulation. Yet the multi-step generative mechanism that gives flow-based VLAs flexibility also makes online policy optimization substantially harder than for explicit action-distribution policies.

After supervised fine-tuning (SFT) converges, robustness remains constrained by the behavioral manifold covered by expert demonstrations. Online reinforcement learning (RL) can use execution-time feedback to adapt the policy beyond these demonstrations. A distinctive challenge for flow-based VLAs is translating rollout feedback into generative-policy updates. The policy outputs a velocity field, and the executed action is obtained by numerically integrating an ordinary differential equation (ODE) over multiple denoising steps rather than by sampling once from an explicit action distribution. Consequently, exact likelihood-based updates require costly Jacobian-trace estimation, while direct optimization through the sampler entails solver backpropagation.

Online RL for flow-based VLAs involves two distinct temporal levels. At the environment level, temporal credit determines how feedback is assigned across states and executed actions according to their contribution to future returns. At the denoising level, rollout feedback must be translated into updates of the sequence of velocity predictions used to generate each action. Existing methods translate rollout feedback into policy improvement through objectives that either bypass or approximate the action likelihoods of generative policies. Some bypass explicit likelihoods through value-guided latent-space policy optimization (Li et al., 2025b; Wagenmaker et al., 2025) or by conditioning action generation on an auxiliary value estimate (Intelligence et al., 2025). Policy-gradient approaches

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

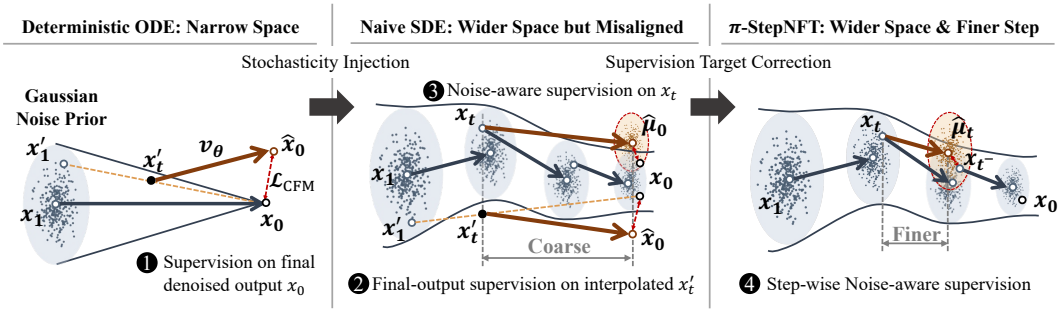


Figure 1: **Illustration of the design of π-StepNFT.** ODE ❶ and naive SDE ❷ retain supervision from the final denoised output x_0 at reconstructed interpolation states x'_t. Noise-aware supervision ❸ moves to sampled solver states x_t with local covariance normalization. π-StepNFT ❹ further targets the observed next state x_{t-}. Wider space denotes the local stochastic transition neighborhood.

instead approximate step-wise likelihoods through Gaussian parameterization along the denoising process (Chen et al., 2025), allowing reward or advantage signals to be propagated through the sampler. In contrast, Diffusion-NFT (Zheng et al., 2026) provides a likelihood-free alternative from image generation, directly optimizing the flow field on the forward process and defining an implicit policy improvement direction through mirrored positive and negative branches.

Diffusion-NFT deliberately decouples optimization from the rollout sampler (Zheng et al., 2026). This sampler-agnostic formulation is natural for image generation, where optimization focuses on a static final sample. In embodied control, however, perturbations along the denoising path can alter the executed action and propagate through closed-loop interaction. With the initial noise fixed, an ODE rollout follows a deterministic denoising path, whereas an equivalent stochastic differential equation (SDE) preserves the same marginals while exploring a local stochastic next-state distribution at each solver state x_t, yielding observed transitions $x_t \to x_{t-}$ (Liu et al., 2025). Diffusion-NFT still constructs intermediate flow-matching supervision from the final denoised output x_0 through the forward process, rather than directly using these observed transitions. This creates an exploration–supervision mismatch within the denoising trajectory.

Figure 1 illustrates this mismatch and the design progression toward π-StepNFT. Under ODE rollouts, supervision derived from the final denoised output x_0 is applied at reconstructed interpolation states x'_t ❶. SDE stochasticity widens the local transition space ❷ and introduces timestep-dependent transition noise. We move supervision to sampled solver states x_t with noise-aware normalization ❸, but targets derived from x_0 remain coarse. π-StepNFT instead uses the observed next state x_{t-} ❹, yielding finer step-wise guidance over the wider stochastic transition space.

To address this mismatch, we propose ***π-StepNFT***, a flow-native online RL framework for flow-based VLAs coupling wider stochastic exploration with finer step-wise supervision without computing final-action likelihoods. It uses variance-normalized errors on each observed SDE transition and replaces regression weighting with contrastive ranking over mirrored positive and negative velocity branches to map rollout feedback to local velocity updates. Our contributions are as follows:

- We identify an exploration–supervision mismatch in embodied generative control. SDE sampling explores a wider local stochastic transition space, while supervision derived from the final denoised output does not directly use the adjacent transitions observed during SDE sampling.
- We propose a flow-native online RL objective that maps rollout feedback to local velocity updates on observed SDE transitions, using variance-normalized transition errors to rank mirrored positive and negative velocity branches.
- We show that the expected ranking gradient follows the outcome-conditioned transition-mean difference at the rollout policy. Relative branch comparison removes the fixed quadratic displacement penalty in weighted regression and, under binary feedback, explicitly optimizes the relative error between both mirrored branches.
- We validate π-StepNFT on LIBERO (Liu et al., 2023) and ManiSkill (Mu et al., 2021). On LIBERO, starting from shared few-shot SFT initializations, π-StepNFT raises average success from 57.6% to 92.0% for π_0 and from 77.1% to 94.0% for $\pi_{0.5}$. On ManiSkill, it obtains the highest reported mean success among evaluated RL baselines on five test-only visual shifts.

2 PRELIMINARIES

We use two time scales: environment steps and denoising time. An episode trajectory is indexed by environment steps $i = 0, \ldots, H-1$, yielding $\tau = \{(s_i, a_i)\}_{i=0}^{H-1}$. Separately, a flow policy generates each action through denoising over continuous time $t \in [0, 1]$ or its discretization. When discretized, we use K solver steps with schedule $1 = t_0 > t_1 > \cdots > t_K = 0$, and denote sampler states by $\{x_{t_j}\}_{j=0}^{K}$. Unless noted, i indexes environment interaction and t denotes denoising time.

2.1 FLOW MATCHING FOR VLA MODELS

We consider a flow-based VLA policy that maps context c and Gaussian noise $x_1 \sim \mathcal{N}(0, I)$ to an executed action $x_0 \in \mathbb{R}^d$. Flow matching (Lipman et al., 2022) learns a velocity field $v_\theta(x_t, t, c)$ over interpolants $x'_t = tx_1 + (1-t)x_0$, with target velocity $u_t = x_1 - x_0$. Since $x_0 = x'_t - tu_t$, this velocity regression can also be interpreted as supervision on x_0 (Figure 1 (Left)). The standard conditional flow-matching (CFM) objective is

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t, x_0, x_1}\left[\left\| v_\theta(x'_t, t, c) - u_t \right\|_2^2\right]. \tag{1}$$

ODE-to-SDE Sampling. Deterministic flow sampling integrates the ODE from $t = 1$ to $t = 0$, so each x_t maps to a single next state x_{t^-}. The reverse-time SDE (Liu et al., 2025) preserves the same marginals but adds Wiener noise, turning this update into a Gaussian transition $q_{\theta,t}(x_{t^-} \mid x_t, c) = \mathcal{N}(\mu_{\theta,t}(x_t, c), \Sigma_t)$ with $\Sigma_t = \sigma_t^2 \delta_t I$, where $\sigma_t = \sigma_0 \sqrt{\frac{t}{1-t^-}}$, thereby widening the local transition neighborhood. Its mean remains affine in the predicted velocity:

$$\mu_{\theta,t}(x_t, c) = U_t(x_t, t) + B_t(t)\, v_\theta(x_t, t, c), \tag{2}$$

where U_t and B_t depend on the noise schedule (Appendix B.2).

2.2 NEGATIVE-AWARE FINE-TUNING

Negative-aware Fine-tuning (NFT) was introduced for verification-driven math reasoning as a supervised-style alternative to policy-gradient RL (Chen et al., 2026). Given online generations from a rollout policy π^{old}, a binary verifier partitions them into positive and negative sets $\mathcal{D}^+$ and $\mathcal{D}^-$. NFT treats them as samples from two latent policies π^+ and π^-, using their gap to define an implicit improvement direction instead of discarding failures as in rejection-sampling fine-tuning.

Diffusion-NFT (Zheng et al., 2026) extends this idea to flow-based image generation, where the policy is induced by a velocity field. Let v^{old} be the rollout field, and let $v^{\pm}$ be the latent fields associated with positive and negative samples. Since they are unobserved, Diffusion-NFT uses mirrored branches around v^{old},

$$v_\theta^+ = (1-\beta)v^{\mathrm{old}} + \beta v_\theta, \qquad v_\theta^- = (1+\beta)v^{\mathrm{old}} - \beta v_\theta, \tag{3}$$

and optimizes the weighted flow-matching loss

$$\mathcal{L}_{\mathrm{NFT}}(\theta) = r\, \mathcal{L}_{\mathrm{CFM}}^+(\theta) + (1-r)\, \mathcal{L}_{\mathrm{CFM}}^-(\theta), \tag{4}$$

where $\mathcal{L}_{\mathrm{CFM}}^{\pm}$ replaces the predictor in Eq. (1) with $v_\theta^{\pm}$. This preserves the supervised regression structure of flow matching while linking the optimum to the oracle improvement direction: under unlimited data and model capacity, $v_\theta^\star = v^{\mathrm{old}} + \frac{2}{\beta}\Delta$, where Δ denotes the latent positive direction.

3 METHOD

We introduce π-StepNFT (Step-wise Negative-aware Fine-Tuning), a flow-native online RL framework for flow-based VLAs that avoids computing the final-action likelihood induced by the full sampler. As illustrated in Figure 1, rather than reconstructing flow-matching supervision from the final output x_0, we use each sampled SDE transition $x_t \to x_{t^-}$ as a local supervision unit and normalize its transition errors by the local SDE covariance.

We further reformulate how optimality feedback enters negative-aware fine-tuning. Eq. (4) uses the optimality score as a regression weight, causing the two mirrored branches to contribute in different

Algorithm 1 π-StepNFT: Step-wise Negative-aware Fine-Tuning

Require: SFT policy π^{old} induced by v^{old}, simulator $\mathcal{E}$, schedule $\{t_j\}_{j=0}^{K}$, hyperparameters $\beta, \lambda_{\text{TR}}$.
Initialize: trainable velocity $v_\theta \leftarrow v^{\text{old}}$ and buffer $\mathcal{D} \leftarrow \emptyset$; use shorthand $x_{i,j} := x_{t_j}^{(i)}$.
for each iteration m **do**
  // Rollout Phase
  for each task (s_0, l) **do**
    Rollout π^{old} in $\mathcal{E}$ to obtain trajectory τ and SDE paths $\{\{x_{t_j}^{(i)}\}_{j=0}^{K}\}_{i=0}^{H-1}$.
    Assign an optimality score $r_i \in [0, 1]$ to each environment step i from rollout feedback.
    Store $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x_{t_j}^{(i)}, x_{t_{j+1}}^{(i)}, v^{\text{old}}(x_{t_j}^{(i)}, t_j, c_i), t_j, c_i, r_i)$ where $j \sim \mathcal{U}\{0, \dots, K-1\}\}_{i=0}^{H-1}$.
  end for
  // Optimization Phase
  for each mini-batch $(x_t, x_{t^-}, v_t^{\text{old}}, t, c, r) \sim \mathcal{D}$ **do**
    $\Delta v_{\theta,t} \leftarrow v_\theta(x_t, t, c) - v_t^{\text{old}}, \quad v_{\theta,t}^{\pm} \leftarrow v_t^{\text{old}} \pm \beta \Delta v_{\theta,t}$.
    $(\mu_{\theta,t}^{\pm}, \Sigma_t) \leftarrow \textsc{MeanCov}(x_t, v_{\theta,t}^{\pm}, t), \quad E_{\theta,t}^{\pm} \leftarrow \|x_{t^-} - \mu^{\pm}\|_{\Sigma_t^{-1}}^2$. ▷ MeanCov: Eq. (2)
    Update policy: $\theta \leftarrow \theta - \eta \nabla_\theta \left[\text{softplus}\left(\frac{1}{2}(2r-1)(E_{\theta,t}^{+} - E_{\theta,t}^{-})\right) + \lambda_{\text{TR}} \|\Delta v_{\theta,t}\|_2^2\right]$.
  end for
  EMA update rollout policy: $\theta^{\text{old}} \leftarrow \alpha_m \theta^{\text{old}} + (1 - \alpha_m)\theta, \quad \mathcal{D} \leftarrow \emptyset$.
end for
Output: optimized flow policy π_θ induced by v_θ.

proportions. While this weighting is natural for graded final-sample quality in image generation, it does not explicitly compare the two mirrored branch errors. We instead use the optimality score to define a branch preference and optimize a relative ranking objective, encouraging the outcome-favored branch to better explain the same observed transition.

3.1 Step-wise Transitions and Mirror Errors

We collect rollouts with an EMA policy π^{old}, where π_θ is the action policy induced by velocity field v_θ. At environment step i, the SDE solver applies v^{old} over denoising steps to produce $\{x_{t_j}^{(i)}\}_{j=0}^{K}$ (omitting (i) below), then executes chunked x_{t_K}, equivalently x_0, as the action. We uniformly sample one adjacent transition $(x_t, x_{t^-}, t) = (x_{t_j}, x_{t_{j+1}}, t_j)$ with $j \sim \mathcal{U}\{0, \dots, K-1\}$ and cache $v_t^{\text{old}} = v^{\text{old}}(x_t, t, c)$.

Given current prediction $v_{\theta,t} = v_\theta(x_t, t, c)$, define its local deviation from the rollout policy: $\Delta v_{\theta,t} = v_{\theta,t} - v_t^{\text{old}}$. Analogous to Eq. (3), we form two local velocity candidates around the rollout velocity: $v_{\theta,t}^{\pm} = v_t^{\text{old}} \pm \beta \Delta v_{\theta,t}$, where $\beta > 0$ controls branch displacement. This is equivalent to Eq. (3), but applied to one observed solver transition. By Eq. (2), candidate $v_{\theta,t}^{\pm}$ yields a one-step Gaussian transition with shared covariance Σ_t and mean $\mu_{\theta,t}^{\pm} = \mu_t(v_{\theta,t}^{\pm})$. We measure how well each branch explains x_{t^-} by variance-normalized transition errors

$$E_{\theta,t}^{\pm} = \left\| x_{t^-} - \mu_{\theta,t}^{\pm} \right\|_{\Sigma_t^{-1}}^2 . \tag{5}$$

The local SDE covariance Σ_t normalizes transition errors for comparison across denoising steps.

3.2 Step-wise Contrastive Objective

Algorithm 1 assigns each sampled SDE transition the environment-step optimality score $r_i \in [0, 1]$, denoted r below. By default, $r_i = R(\tau) \in \{0, 1\}$ broadcasts the trajectory outcome. Critic-derived feedback can provide step-dependent scores. The signed label $y = 2r - 1$ identifies the preferred branch, while $|y|$ controls preference strength. Ideally, we compare whether x_{t^-} is more likely under $p(x_{t^-} \mid x_t, c, o = 1)$ or $p(x_{t^-} \mid x_t, c, o = 0)$. Here, $o \in \{0, 1\}$ denotes the latent outcome underlying the positive and negative transition splits. Since these oracle outcome-conditioned distributions are unobservable, we use the mirrored branches from Section 3.1.

Definition 3.1 (π-StepNFT Step-wise Ranking Loss)**.** For a sampled transition tuple (x_t, x_{t^-}, t, c) with signed label $y \in [-1, 1]$, let $E_{\theta,t}^{+}$ and $E_{\theta,t}^{-}$ be the variance-normalized errors from Section 3.1.

Since $q^{\pm}_{\theta,t}(x_{t^-} \mid x_t, c) = \mathcal{N}(\mu^{\pm}_{\theta,t}, \Sigma_t)$ share covariance, the step-wise contrastive loss is

$$\ell_t(\theta) = \mathrm{softplus}\left(\frac{1}{2} y \left(E^{+}_{\theta,t} - E^{-}_{\theta,t}\right)\right) = \mathrm{softplus}\left(-y \log \frac{q^{+}_{\theta,t}(x_{t^-} \mid x_t, c)}{q^{-}_{\theta,t}(x_{t^-} \mid x_t, c)}\right). \tag{6}$$

Minimizing Eq. (6) ranks the two mirrored branches on the same solver transition. For $y > 0$, the loss favors a larger positive-to-negative transition log-ratio, or a smaller relative error $E^{+}_{\theta,t} - E^{-}_{\theta,t}$; $y < 0$ reverses this preference. Unlike DPO-style preference learning over two sampled outputs, our objective ranks local perturbations around the rollout policy on one transition. Since y supplies preference direction, optimization uses only the one-step transition ratio, not the final-action likelihood induced by the full sampler.

In practice, Algorithm 1 adds $\lambda_{\mathrm{TR}} \|\Delta v_{\theta,t}\|_2^2$ to control update magnitude. Since this term stabilizes optimization but does not define branch preference, Section 3.4 analyzes the ranking loss $\ell_t(\theta)$.

3.3 Objective Comparison with Weighted Regression

To isolate the effect of the learning objective, we compare step-wise ranking with a weighted-regression baseline using the same observed SDE transition and the same variance-normalized mirrored errors $E^{\pm}_{\theta,t}$. Under our step-wise formulation, the regression baseline replaces the conditional flow-matching losses in Eq. (4) with these transition errors:

$$\ell_t^{\mathrm{wMSE}}(\theta) = r E^{+}_{\theta,t} + (1 - r) E^{-}_{\theta,t}. \tag{7}$$

Although both objectives use the same mirrored branches and transition errors, they differ in how they treat branch displacement and combine the two branch errors.

Lemma 3.2 (Weighted regression and step-wise ranking decomposition)**.** *For a sampled solver transition (x_t, x_{t^-}, t, c), let $\mu_t^{\mathrm{old}} = \mu_t(v_t^{\mathrm{old}})$ be the rollout transition mean, $e_t = x_{t^-} - \mu_t^{\mathrm{old}}$ be the residual of the rollout mean, $d_t = \mu^{+}_{\theta,t} - \mu_t^{\mathrm{old}}$ be the displacement of the positive branch in mean space, and B_t be the affine coefficient from Eq.* (2)*. Under the mirrored construction in Section 3.1, we have $d_t = \beta B_t \Delta v_{\theta,t}$, $\mu^{\pm}_{\theta,t} = \mu_t^{\mathrm{old}} \pm d_t$. Therefore:*

(a) *The step-wise weighted-regression objective in Eq.* (7) *satisfies*

$$\ell_t^{\mathrm{wMSE}}(\theta) = \mathrm{const} - 2y \left\langle \Sigma_t^{-1} e_t, d_t \right\rangle + \|d_t\|^2_{\Sigma_t^{-1}}. \tag{8}$$

(b) *The step-wise ranking objective in Eq.* (6) *satisfies*

$$\ell_t(\theta) = \mathrm{softplus}\left(-2y \left\langle \Sigma_t^{-1} e_t, d_t \right\rangle\right). \tag{9}$$

Fixed displacement penalty in weighted regression. Eq. (8) decomposes weighted regression into the outcome-signed alignment term $-2y\langle \Sigma_t^{-1} e_t, d_t \rangle$ and a fixed quadratic displacement penalty $\|d_t\|^2_{\Sigma_t^{-1}}$. The latter is independent of the label y and therefore always penalizes displacement from the rollout transition mean μ_t^{old}. In contrast, this common quadratic term cancels when the two mirrored branch errors are compared relatively in Eq. (9).

Relative branch comparison under binary feedback. For $r = 1$, weighted regression reduces to $E^{+}_{\theta,t}$, while for $r = 0$ it reduces to $E^{-}_{\theta,t}$. Thus, each binary-labeled sample optimizes one outcome-selected absolute branch error. Ranking instead directly optimizes the relative error $E^{+}_{\theta,t} - E^{-}_{\theta,t}$, placing both mirrored branch errors in every sample objective. Both objectives remain negative-aware by learning from positive and negative samples, but ranking makes the branch comparison explicit at the per-sample level.

Together, ranking and weighted regression use the same mirrored branches and outcome feedback but differ in objective structure. Ranking removes the fixed quadratic displacement penalty in weighted regression and replaces outcome-selected absolute-error fitting with explicit relative branch comparison under binary feedback.

3.4 Outcome-Conditioned Gradient Analysis

We examine whether ranking the constructed mirrored branches yields an update related to outcome-conditioned differences in local denoising transitions. Fix context c, solver state x_t and denoising time t. With latent outcome $o \in \{0, 1\}$ (Section 3.2), define outcome-conditioned transition means

$$\mu_t^+(x_t, c) = \mathbb{E}[x_{t^-} \mid x_t, c, o = 1], \qquad \mu_t^-(x_t, c) = \mathbb{E}[x_{t^-} \mid x_t, c, o = 0]. \tag{10}$$

We denote their difference by $\Delta\mu_t^\star = \mu_t^+ - \mu_t^-$, the oracle outcome-conditioned transition-mean difference. These means are unavailable during training. π-StepNFT instead constructs mirrored transition means $\mu_{\theta,t}^\pm = \mu_t^{\text{old}} \pm d_t$, where $d_t = \beta B_t \Delta v_{\theta,t}$. They are local perturbations around the rollout transition mean rather than estimates of $\mu_t^\pm$. We therefore ask whether ranking these constructed branches yields an expected update related to $\Delta\mu_t^\star$. The following proposition establishes this relation when the actor matches the rollout policy and accounts for soft-score mismatch.

Proposition 3.3 (Outcome-conditioned gradient at the rollout policy)**.** *Fix the rollout transition distribution with $\Sigma_t \succ 0$, and hold cached rollout quantities and scores constant with respect to θ. Assume the conditional expectations below exist. Let θ_0 satisfy $v_{\theta_0,t} = v_t^{\text{old}}$, and set $J_0 = \partial v_{\theta,t}/\partial\theta|_{\theta=\theta_0}$. For $o \in \{0, 1\}$, $r \in [0, 1]$, and $y = 2r - 1$, define $\alpha(x_t, c) = \mathbb{P}(o = 1 \mid x_t, c) \in (0, 1)$, and $\kappa_t(x_t, c) = 2\alpha(x_t, c)\big(1 - \alpha(x_t, c)\big)$. Let $\varepsilon_t^r(x_t, c) = 2\mathbb{E}[(r - o)e_t \mid x_t, c]$ be the score-mismatch bias, with e_t from Lemma 3.2. Then*

$$\mathbb{E}[-\nabla_\theta \ell_t(\theta) \mid x_t, c]|_{\theta=\theta_0} = \beta J_0^\top B_t^\top \Sigma_t^{-1} \left[\kappa_t(x_t, c)\Delta\mu_t^\star(x_t, c) + \varepsilon_t^r(x_t, c)\right], \tag{11}$$

When $\varepsilon_t^r(x_t, c) = 0$ and $0 < \alpha(x_t, c) < 1$, this becomes

$$\mathbb{E}[-\nabla_\theta \ell_t(\theta) \mid x_t, c]|_{\theta=\theta_0} = \beta\kappa_t J_0^\top B_t^\top \Sigma_t^{-1} \Delta\mu_t^\star(x_t, c). \tag{12}$$

For binary feedback $r = o$, the score-mismatch bias ε_t^r vanishes, and the outcome-conditioned transition-mean difference enters the expected ranking gradient through covariance weighting, $B_t^\top$, and the velocity Jacobian. Thus, the ranking objective yields an outcome-conditioned local update without estimating the oracle means. For soft feedback, ε_t^r measures how score mismatch correlates with the transition residual. Posterior-like scores $r(x_0, c) = \mathbb{P}(o = 1 \mid x_0, c)$ give zero bias when the intermediate transition provides no additional outcome information beyond x_0 and c, i.e., when $\mathbb{P}(o = 1 \mid x_0, c, x_t, x_{t^-}) = \mathbb{P}(o = 1 \mid x_0, c)$.

For finite deviations from the rollout policy under binary feedback, Appendix B.5 derives an additional finite-update correction $-\zeta_t$. It vanishes when the actor matches the rollout policy and remains bounded. The analysis also gives a sufficient condition for positive alignment with the oracle transition-mean difference in variance-normalized transition-mean space. Its magnitude depends on the actor–rollout gap, which EMA rollout updates and trust-region regularization help limit.

4 Experiments

We evaluate π-StepNFT through gains over few-shot SFT, OOD robustness under visual shifts (Section 4.2), and mechanistic analysis of its design (Section 4.3).

4.1 Experiment Setup

Benchmarks & Baselines. We evaluate π-StepNFT on LIBERO (Spatial, Object, Goal, Long) and ManiSkill (PutOnPlateInScene). We additionally use CALVIN (Mees et al., 2022) for long-horizon diagnostics. Using π_0 and $\pi_{0.5}$ from OpenPi as base policies, we initialize from π_{RL} (Chen et al., 2025) SFT checkpoints. On LIBERO, all online RL methods start from the same controlled low-demonstration SFT checkpoints, allowing us to measure policy improvement from a shared initialization; ManiSkill uses full multitask SFT.

Training Protocol. During RL fine-tuning with RLinf (Yu et al., 2025), we freeze the VLM backbone and update the flow-matching action expert. Unless otherwise specified, π-StepNFT uses trajectory-level binary success on LIBERO and ManiSkill. GRPO- and GAE-based soft feedback are mapped to $r \in [0, 1]$ using Diffusion-NFT's normalization (Zheng et al., 2026) with fixed scale $Z = 3$. π-StepNFT + Critic uses a PPO critic to compute GAE while retaining the same ranking objective for actor updates. We report success rates. Appendix C gives implementation details.

Table 1: Success rates (%) on LIBERO. π-StepNFT: three-seed means. π_{RL}: reported results from Chen et al. (2025).

Model		Spatial	Object	Goal	Long	Avg.	Δ Avg.
# Full SFT							
Octo (Ghosh 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., 2025b)		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	—
# 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	95.4	98.0	87.9	86.7	92.0	+34.4
	π-StepNFT + Critic	97.6	98.0	89.8	90.0	93.9	+36.3
# 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	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	98.2	79.8	94.0	+16.9
	π-StepNFT + Critic	98.0	99.3	98.2	88.4	96.0	+18.9

Table 2: Success rates (%) on ManiSkill for $\pi_{0.5}$. π-StepNFT: three-seed means. π_{RL}: reported results from Chen et al. (2025).

Environment	Variation	$\pi_{0.5}$-SFT	$\pi_{0.5}$-RL Flow-SDE	$\pi_{0.5}$-RL Flow-Noise	$\pi_{0.5}$-StepNFT
In distribution	Main-v3-train	40.1	**90.9**	89.7	84.3
Visual-Language Variations	Instruct-v1-test	46.6	77.0	**85.7**	82.9
	VisionImage-v1-test	46.2	78.8	**83.1**	80.5
	VisionTexture03-v1-test	36.7	69.6	75.0	**80.5**
	VisionTexture05-v1-test	32.7	58.0	62.2	**71.2**
	VisionWhole03-v1-test	40.1	69.6	71.6	**82.0**
	VisionWhole05-v1-test	30.7	55.0	57.0	**62.5**
	Visual Avg. (5)	37.3	66.2	69.8	**75.3**
	Visual-Language Avg. (6)	38.8	68.0	72.4	**76.6**
Semantic Reasoning (object/receptacle)	MultiCarrot-v1-test	16.7	36.8	38.2	**39.1**
	MultiPlate-v1-test	11.8	**29.4**	28.3	21.9
	Avg.	14.3	33.1	**33.3**	30.5
Action Execution	PositionChangeTo-v1-test	13.5	36.2	**54.7**	28.1
	Position-v1-test	31.2	54.5	55.0	**56.3**
	Avg.	22.4	45.4	**54.9**	42.2

4.2 Main Results: Online Improvement and Robustness under Visual Shifts

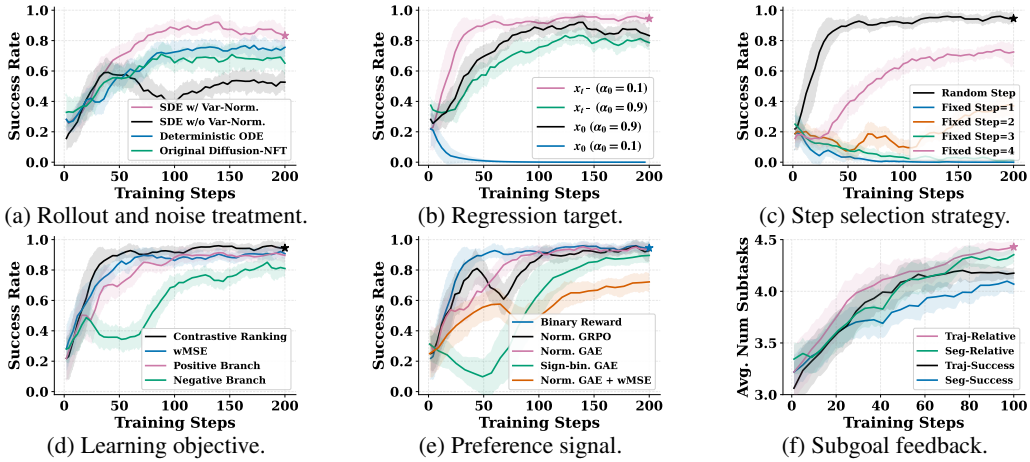
Online Improvement from Shared Few-shot SFT Initializations. Starting from the same few-shot SFT initialization for all compared online RL methods within each backbone, π-StepNFT raises average success from 57.6% to 92.0% for π_0 and from 77.1% to 94.0% for $\pi_{0.5}$, with means 2.0 and 2.5 percentage points higher than Flow-SDE + GRPO, respectively (Table 1). Incorporating critic-derived temporal-credit signals into the same ranking objective further improves the averages to 93.9% and 96.0%. The largest gains from critic supervision occur on Long, where success increases from 86.7% to 90.0% for π_0 and from 79.8% to 88.4% for $\pi_{0.5}$, narrowing the corresponding gaps to PPO to 0.2 and 4.6 percentage points. The ranking interface supports both binary outcomes and critic-derived temporal credit, while PPO retains the highest overall means.

Higher Reported Mean Performance under Visual Shifts in ManiSkill. For $\pi_{0.5}$, across the five test-only visual perturbations, π-StepNFT obtains a mean success rate of 75.3%, compared with 66.2% for Flow-SDE and 69.8% for Flow-Noise, and achieves the highest reported result on four of the five variants (Table 2). Relative to its in-distribution performance, mean success decreases by 9.0 percentage points under these perturbations, compared with drops of 24.7 and 19.9 percentage points for Flow-SDE and Flow-Noise, respectively. This robustness advantage is concentrated in the tested visual appearance shifts. The PPO baselines remain stronger on the in-distribution setting and achieve higher category averages under semantic and execution shifts.

4.3 Mechanistic Analysis: Step-wise Guidance and Outcome-based Ranking

Figure 2 traces the design choices from Original Diffusion-NFT to π-StepNFT: rollout weighting and noise treatment, transition targets, step selection, and the ranking objective. The final panels examine feedback choice on short- and long-horizon tasks.

Wider space needs finer steps. Figure 2a holds the final x_0 target fixed under weighted regression and a conservative EMA schedule (α: $0.9 \rightarrow 0.995$). Original Diffusion-NFT is a direct transfer baseline that combines ODE rollouts with soft GRPO weighting and stays below 0.7 success. Deterministic ODE keeps the same endpoint-derived regression but replaces soft GRPO scores with binary outcomes, improving performance before plateauing ❶. Introducing SDE rollouts without variance normalization gives limited, unstable gains ❷; adding noise-aware normalization improves training ❸. Figure 2b then compares targets under stochastic rollouts: x_0 regression is only partly stabilized by conservative EMA ($\alpha_0 = 0.9$) and collapses at $\alpha_0 = 0.1$. While final-output regression trains on intermediate states, it does not directly fit the observed adjacent SDE transition. Targeting the immediate x_{t-} instead provides local, noise-aware supervision and improves stability and convergence ❹. Its stability at $\alpha_0 = 0.1$, where x_0 regression collapses, shows that the gain is not solely due to conservative EMA updates. Together, these comparisons support pairing wider SDE exploration with noise-aware supervision of the observed adjacent transition, rather than relying on exploration alone.



(a) Rollout and noise treatment. (b) Regression target. (c) Step selection strategy.

(d) Learning objective. (e) Preference signal. (f) Subgoal feedback.

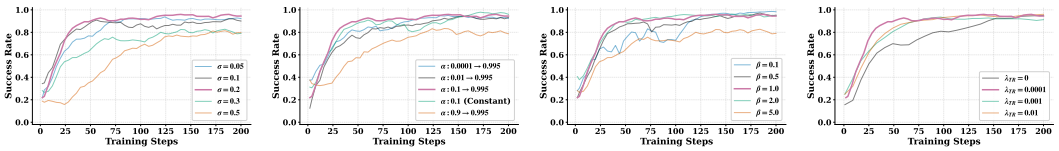
Figure 2: **Component-wise ablations of π-StepNFT.** (a) With x_0 targets, Original Diffusion-NFT uses ODE rollouts and GRPO-weighted MSE; Deterministic ODE uses binary-outcome weighting instead. SDE w/o and w/ Var-Norm then introduce stochastic rollouts without and with noise-aware normalization. (b) Immediate x_{t^-} targets stabilize stochastic training. (c) Random step selection improves stability over fixed steps. (d) Ranking outperforms weighted-MSE and the separate Positive/Negative Branch ablations. (e) Binary outcomes perform best on the short-horizon ablation; ranking also supports soft scores. (f) Relative subgoal feedback helps in a controlled long-horizon diagnostic. Curves show five-seed means with ± 1 standard deviation.

Random step selection improves stability. Figure 2c shows non-monotonic fixed-step performance, with some choices collapsing. Random selection covers different denoising stages and achieves the best, most stable performance with single-transition updates.

Contrastive ranking beats weighted regression. Under the same step-wise transition formulation, Figure 2d compares objectives. The Positive Branch and Negative Branch ablations use all samples but retain only the respective branch loss. Both remain below weighted-MSE: positive-only improves slowly, while negative-only drops mid-training. Weighted-MSE converges more slowly and reaches lower final success than ranking. With binary feedback, weighted-MSE explicitly fits an outcome-selected absolute error; ranking compares both mirrored errors on each transition. This ordering favors outcome-selected regression over isolated-branch objectives, and relative comparison over weighted regression (Section 3.3).

Binary outcomes are strongest on the short-horizon ablation. Figure 2e next examines feedback: terminal binary outcomes, normalized GRPO and GAE advantages, and sign-binarized GAE on the default LIBERO-Object ablation task. Binary outcomes converge fastest and achieve the highest final success. Sign-binarized GAE remains weaker, showing that binarization alone does not explain the difference. With the same normalized scores r from PPO-critic GAE, ranking outperforms weighted-MSE. Holding this feedback fixed isolates the actor objective from the feedback source. The gain therefore extends beyond binary feedback. Together with the main LIBERO results, this favors binary outcomes on the short-horizon Object task, while critic-derived signals yield their largest gains on Long.

Relative subgoal feedback helps in a controlled long-horizon diagnostic. For long-horizon feedback, Figure 2f uses a fixed five-subgoal CALVIN sequence, which provides subgoal rewards unavailable in LIBERO-Long, to compare four label constructions from subtask completion. Traj-Success broadcasts final success to all segments, whereas Seg-Success labels each segment by immediate subgoal completion. Traj-Relative compares each rollout's completion count with the mean across the current rollout batch, while Seg-Relative applies the same relative comparison to each segment's subsequent completion count. Direct segment labels underperform trajectory-success labels, suggesting that isolated subgoal completion need not support subsequent subtasks. Among the relative variants, Traj-Relative performs best, while Seg-Relative is close but less stable, supporting relative long-horizon preferences in this diagnostic.



(a) Noise level σ. (b) Decay α. (c) Branch scale β. (d) Trust reg. weight λ_{TR}.

Figure 3: **Hyperparameter sensitivity analysis.** Curves show five-seed means with ± 1 standard deviation; **bold pink** marks the main-experiment configuration.

4.4 Hyperparameter Sensitivity Analysis

Figure 3 shows that excessive SDE noise impedes convergence, while insufficient noise limits local stochastic exploration. Dynamic EMA decay balances rollout-policy lag against aggressive-update instability. Branch displacement scale $\beta \in [1, 2]$ performs best; this scale controls normalized branch displacement and ranking strength. Trust-region weight $\lambda_{\text{TR}} = 10^{-4}$ regularizes updates without slowing convergence. Appendix C.5 provides further analysis.

5 Related Works

5.1 Online RL for VLAs

VLA policies span action tokenization (Zitkovich et al., 2023; Kim et al., 2024), continuous regression (Kim et al., 2025b), and flow-based action generation (Ghosh et al., 2024; Black et al., 2026; 2025; Bjorck et al., 2025). Tokenized policies expose categorical probabilities and continuous policies often use Gaussian heads (Lu et al., 2025; Liu et al., 2026; Li et al., 2025a), whereas flow-based VLAs use multi-step ODE sampling whose final-action likelihoods are not directly tractable. Existing methods bypass this bottleneck through latent-space value learning (Li et al., 2025b; Wagenmaker et al., 2025), advantage-conditioned action generation (Intelligence et al., 2025), and Q-learning-based actor distillation (Park et al., 2025), or approximate likelihood-based optimization via denoising Gaussian parameterization (Chen et al., 2025) or conditional-flow-matching importance surrogates (Lyu et al., 2025). In contrast, π-StepNFT avoids computing final-action likelihoods and ranks variance-normalized one-step SDE transition errors across negative-aware mirrored branches.

5.2 Policy Optimization for Generative Models

Policy optimization for diffusion and flow models treats denoising as a sequential decision process (Black et al., 2023; Fan et al., 2023; Liu et al., 2025; Zhang et al., 2025b), requiring tractable denoising likelihoods or sampler backpropagation. SAC Flow stabilizes sampler backpropagation through velocity-reparameterized off-policy SAC (Zhang et al., 2026). Other methods avoid sampler gradients through value-based diffusion-policy objectives (Ma et al., 2025), adjoint-matched Q-learning (Li & Levine, 2026), reward-weighted flow-matching surrogates (Pfrommer et al., 2025; Fan et al., 2025; McAllister et al., 2025), or preference-ranking losses over final outputs or intermediate samples (Wallace et al., 2024; Zhang et al., 2025a; Ren et al., 2025). π-StepNFT instead ranks mirrored local transition models on the same observed SDE transition. Diffusion-NFT (Zheng et al., 2026) instead constructs likelihood-free mirrored branches in the forward process and uses regression targets derived from the final denoised output. π-StepNFT shares its negative-aware motivation but replaces endpoint-derived regression with observed-transition supervision.

6 Conclusion

We introduced π-StepNFT to address the exploration–supervision mismatch in online RL for flow-based VLAs. It maps binary or critic-derived feedback to local velocity updates by ranking variance-normalized errors on observed SDE transitions, without computing final-action likelihoods. Our analysis characterizes the outcome-conditioned gradient at the rollout policy and its finite-update correction under binary feedback. Experiments show gains over few-shot SFT on LIBERO and the highest reported mean success among evaluated online RL baselines across five test-only ManiSkill visual shifts.

AI USE STATEMENT

OpenAI Codex was used to assist with language editing, presentation and organization, literature search, LaTeX formatting, and revisions to the description and interpretation of results. It was not used to generate experimental data.

REPRODUCIBILITY STATEMENT

Section 3.1–3.2 and Algorithm 1 specify the transition construction and optimization objective, with derivations and proofs in Appendix B. Section 4.1 describes the evaluation and training protocol. Appendix C provides benchmark definitions, SFT initializations, rollout and update budgets, compute, hyperparameters, and ablation protocols; Appendix D reports additional analyses, system profiling, and full ManiSkill OOD results.

REFERENCES

Jiamu Bai, Xin Yu, Meilong Xu, Weitao Lu, Xin Pan, Kiwan Maeng, Daniel Kifer, Jian Wang, and Yu Wang. Towards better optimization for listwise preference in diffusion models. *arXiv preprint arXiv:2510.01540*, 2025.

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

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. 2023.

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.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. NFT: Bridging supervised learning and reinforcement learning in math reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ujBrsQm6Zu.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online rl fine-tuning for flow-based vision-language-action models, 2025. URL https://arxiv.org/abs/2510.25889.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Jiajun Fan, Shuaike Shen, Chaoran Cheng, Yuxin Chen, Chumeng Liang, and Ge Liu. Online reward-weighted fine-tuning of flow matching with wasserstein regularization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Reinforcement learning for fine-tuning text-to-image diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=8OTPepXzeh.

Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

Jiaqi Han, Austin Wang, Minkai Xu, Wenda Chu, Meihua Dang, Yisong Yue, and Stefano Ermon. Discrete diffusion trajectory alignment via stepwise decomposition. *arXiv preprint arXiv:2507.04832*, 2025.

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

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.

Dohyun Kim, Seungwoo Lyu, Seung Wook Kim, and Paul Hongsuck Seo. Direct diffusion score preference optimization via stepwise contrastive policy-pair supervision. *arXiv preprint arXiv:2512.23426*, 2025a.

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*, 2025b.

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.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vd4eNAdtO6.

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*, 2025b.

Xinyao Liao, Wei Wei, Xiaoye Qu, and Yu Cheng. Step-level reward for free in rl-based t2i diffusion model fine-tuning. *arXiv preprint arXiv:2505.19196*, 2025.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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*, 2025.

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, 2026. URL https://arxiv.org/abs/2505.19789.

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.

Mingyang Lyu, Yinqian Sun, Erliang Lin, Huangrui Li, Ruolin Chen, Feifei Zhao, and Yi Zeng. Reinforcement fine-tuning of flow-matching policies for vision-language-action models, 2025. URL https://arxiv.org/abs/2510.09976.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6Anv3KB9lz.

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.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

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.

NVIDIA. Nvidia h100 tensor core gpu datasheet. Technical report, NVIDIA, 2022. URL https://resources.nvidia.com/en-us-gpu-resources/h100-datasheet-24306. Technical Report.

NVIDIA. Nvidia ampere gpu architecture whitepaper. Technical report, NVIDIA, 2023. URL https://images.nvidia.com/aem-dam/Solutions/geforce/ada/nvidia-ada-gpu-architecture.pdf. Technical Report.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KVf2SFL1pi.

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.

Samuel Pfrommer, Yixiao Huang, and Somayeh Sojoudi. Reinforcement learning for flow-matching policies. *arXiv preprint arXiv:2507.15073*, 2025.

Jie Ren, Yuhang Zhang, Dongrui Liu, Xiaopeng Zhang, and Qi Tian. Refining alignment framework for diffusion models with intermediate-step preference ranking, 2025. URL https://arxiv.org/abs/2502.01667.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

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.

Mingyang Sun, Pengxiang Ding, Weinan Zhang, and Donglin Wang. Iterative refinement of flow policies in probability space for online reinforcement learning. *arXiv preprint arXiv:2510.15388*, 2025.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Andrew Wagenmaker, Mitsuhiko 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.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8228–8238, 2024.

Feng Wang and Zihao Yu. Coefficients-preserving sampling for reinforcement learning with flow matching. *arXiv preprint arXiv:2509.05952*, 2025.

Yi-Lun Wu, Bo-Kai Ruan, Chiang Tseng, and Hong-Han Shuai. Ranking-based preference optimization for diffusion models from implicit user feedback. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Haoyun Li, Jie Li, Jiagang Zhu, Lv Feng, et al. Gigabrain-0: A world model-powered vision-language-action model. *arXiv preprint arXiv:2510.19430*, 2025.

Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, et al. Rlinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. *arXiv preprint arXiv:2509.15965*, 2025.

Tao Zhang, Cheng Da, Kun Ding, Huan Yang, Kun Jin, Yan Li, Tingting Gao, Di Zhang, Shiming Xiang, and Chunhong Pan. Diffusion model as a noise-aware latent reward model for step-level preference optimization. *arXiv preprint arXiv:2502.01051*, 2025a.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning, 2025b. URL https://arxiv.org/abs/2505.22094.

Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. Sac flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zZvWj4JrYj.

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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VJZ477R89F.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

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*, pp. 2165–2183. PMLR, 2023.

A Scope of Compared Methods

A.1 Image-generation Alignment versus Embodied Flow-policy RL

Online RL methods for flow or diffusion models have recently appeared in two related but distinct settings. The first is image-generation alignment, where the model produces a static sample and reward is evaluated on the final generated output. The second is embodied control, where generated actions are executed in a closed-loop MDP and small action errors can affect future states. Therefore, methods developed for image generation are conceptually relevant but not directly comparable as embodied-control baselines unless they are instantiated with a VLA policy, robotic environment rollouts, and matched interaction budgets.

Table 3: Relationship between online RL routes in image generation and their embodied flow-based VLA instantiations.

Conceptual route	**Method**	**Embodied instantiation**	**Compared in this paper**
SDE policy-gradient route	Flow-GRPO (Liu et al., 2025)	Flow-SDE + GRPO	Yes; reported as π_{RL} (Flow-SDE + GRPO)
Flow-noise policy-gradient route	ReinFlow (Zhang et al., 2025b)	Flow-Noise + PPO	Yes; reported as π_{RL} (Flow-Noise + PPO) or Flow-Noise baseline
NFT likelihood-free route	Diffusion-NFT (Zheng et al., 2026)	Direct NFT transfer; π-StepNFT	Yes; direct NFT transfer is included as a diagnostic baseline, and π-StepNFT is our proposed repair

A.2 Comparison Criteria for Online RL Methods for VLAs

For controlled primary numerical comparisons, we prioritize methods that satisfy the following criteria: (i) same policy class: they use a flow-based VLA backbone or can be instantiated on the same $\pi_0/\pi_{0.5}$ backbone; (ii) same online interaction budget: they can be trained under a matched online interaction budget, including rollout trajectories, environment steps, parallel environments, and denoising steps; (iii) same supervision source: they do not require extra supervision beyond the online rollout feedback used in our setting, such as additional demonstrations, human interventions, or large heterogeneous robot data; and (iv) open implementation: their implementations are publicly available or sufficiently specified to avoid uncontrolled reimplementation choices. Table 4 summarizes these criteria for recent online RL methods for VLAs and flow-based policies.

Table 4: Comparability of recent online RL methods for VLAs and flow-based policies. Primary comparisons prioritize a matched flow-based VLA policy class, matched online rollout budget, no extra supervision or heterogeneous training data, and public or sufficiently specified implementation details.

Method	**Key mechanism**	**Flow Backbone**	**Reproducible Recipe**	**Matched Budget**	**Online Feedback**	**Primary Comparison**
π_{RL} Flow-SDE + GRPO	Flow-SDE + GRPO surrogate	✓	✓	✓	✓	✓
π_{RL} Flow-SDE + PPO	Flow-SDE + critic-based PPO	✓	✓	✓	✓	✓
π_{RL} Flow-Noise + PPO	Flow-Noise + PPO likelihood	✓	✓	✓	✓	✓
FPO-VLA (Lyu et al., 2025)	CFM importance surrogate + Q-ensemble	✓	△	△	✓	×
FQL (Park et al., 2025)	Q-learning + actor distillation	△	✓	△	✓	×
DSRL (Wagenmaker et al., 2025)	Latent-noise steering policy	△	✓	△	✓	×
GR-RL (Li et al., 2025b)	Progress value + latent noise prediction	✓/△	△	△	×	×
RECAP / $\pi^*_{0.6}$ (Intelligence et al., 2025)	Advantage-conditioned VLA	✓	×/△	×	×	×
VLA-RL (Lu et al., 2025)	Autoregressive VLA + process reward	×	✓	△	×	×
RL4VLA (Liu et al., 2026)	Explicit-action VLA + PPO-style RL	×	✓	△	✓	×
SimpleVLA-RL (Li et al., 2025a)	OpenVLA-OFT + scalable RL system	×	✓	△	✓	×

Here, ✓ means that the criterion is clearly satisfied under the same experimental setting, × means that it is clearly violated, and △ means that the method is relevant but the criterion cannot be certified

as satisfied for a controlled comparison. This usually occurs when the method is evaluated with a different backbone or policy interface, requires nontrivial additional modules such as critics, Q-ensembles, latent controllers, actor distillation, or reward models, uses a different rollout/update protocol, or lacks public implementation details needed to match these factors.

Scope of compared methods. Our primary comparisons focus on methods that follow the standard two-stage online RL loop: first collecting rollouts with the current policy, and then optimizing the same flow-based VLA policy using the collected online feedback. This rollout–optimization structure makes it possible to control the main experimental budget, including the number of rollout trajectories, environment steps, parallel environments, and denoising steps. Under this criterion, the most controlled baselines are the π_{RL} Flow-SDE and Flow-Noise variants (Chen et al., 2025): they can be instantiated on the same $\pi_0/\pi_{0.5}$ backbone, trained with the same SFT initialization, and evaluated under the same simulator and rollout protocol. In particular, Flow-SDE + GRPO corresponds to the Flow-GRPO-style SDE policy-optimization route, while Flow-Noise + PPO corresponds to the ReinFlow-style noise-based policy-gradient route. These baselines therefore allow us to isolate whether our proposed method provides benefits under a matched online interaction budget.

Why not all related methods are primary baselines. Several recent online RL methods for VLAs or flow-based policies are highly relevant, but they do not provide clean primary baselines under the controlled rollout–optimization setting above.

- RECAP and $\pi^*_{0.6}$ train VLAs from heterogeneous experience, including demonstrations, on-policy collection, and expert teleoperated interventions, and they start from offline RL pretraining before downstream on-robot specialization. Their performance therefore depends on additional data sources and correction channels beyond simulator-only online rollout feedback. Moreover, without open-source code, its full pipeline cannot be reproduced under our matched budget.
- FQL addresses flow-policy RL by training an expressive one-step policy with Q-learning rather than directly optimizing the iterative flow policy itself. This changes the trained policy object and introduces actor distillation and one-step policy extraction.
- DSRL is also not a direct objective-level baseline. It adapts a pretrained diffusion or flow policy by training a small latent-noise policy, without modifying the base policy weights. Its pipeline treats noise as the RL action and can include warmup rollouts or offline-data usage through noise-aliasing value distillation, so its training object and rollout accounting differ from ours.
- FPO-VLA is algorithmically closer because it targets online fine-tuning of flow-matching VLAs through a conditional-flow-matching importance surrogate. However, it also uses structure-aware credit assignment, clipped surrogate objectives, multi-step latent exploration, and Q-ensemble value estimation. It also lacks public code and sufficient hyperparameter details for reliable reproduction under the same backbone, solver schedule, rollout budget, and update protocol.

For these reasons, we discuss RECAP, $\pi^*_{0.6}$, FQL, DSRL, and FPO-VLA as important related routes, while using the matched π_{RL} Flow-SDE and Flow-Noise variants as the primary numerical baselines.

B Theoretical Analysis and Proofs

This section provides the detailed derivations used by Proposition 3.3 and Lemma 3.2.

B.1 Notation Summary

Table 5 summarizes the main symbols used in the method description and theoretical analysis.

Table 5: Terminology and notation summary.

Term / Merged Expressions	**Notation**
Environment step index / episode horizon	i, H
Denoising time / solver schedule / solver steps	t, t_j, K
Environment trajectory / episode trajectory	τ, s_i, a_i
Final denoised output	x_0
Gaussian prior / initial noise	x_1
Target velocity	u_t
SDE transition state / step-wise transition	x_t, x_{t_j}, x_{t^-}, δ_t, $x_t \to x_{t^-}$
Stochastic noise terms	σ_t, ϵ
Covariance / variance-normalization term	Σ_t
Affine transition mean / affine coefficients	μ_t, U_t, B_t
Positive / negative rollout sets	$\mathcal{D}^+$, $\mathcal{D}^-$
Positive / negative latent policies	π^+, π^-
Latent outcome / oracle outcome	o
Optimality score	r
Signed preference label	y
Flow policy / rollout policy / old policy	π_θ, π^{old}
Velocity field / old velocity field	v_θ, v^{old}
Positive / negative mirrored velocity branches	v_θ^+, v_θ^-
Time-indexed velocity values / time-indexed mirrored branches	$v_{\theta,t}$, v_t^{old}, $v_{\theta,t}^+$, $v_{\theta,t}^-$
Local velocity deviation	$\Delta v_{\theta,t}$
Variance-normalized transition errors / mirror errors	$E_{\theta,t}^+$, $E_{\theta,t}^-$
Transition density / branch transition density	$q_{\theta,t}$, $q_{\theta,t}^+$, $q_{\theta,t}^-$
Transition mean / branch transition means	$\mu_{\theta,t}$, $\mu_{\theta,t}^+$, $\mu_{\theta,t}^-$
Rollout transition mean	μ_t^{old}
Outcome-conditioned transition means	μ_t^+, μ_t^-
Oracle improvement direction	Δ
Oracle outcome-conditioned transition-mean difference	$\Delta\mu_t^\star$
Branch displacement scale	β
Trust-region regularizer	λ_{TR}
Rollout residual	e_t
Mean-space branch displacement	d_t
Normalized branch-displacement magnitude	ρ_t
Finite-update correction	$-\zeta_t$
Outcome probability	α
Alignment coefficient	κ_t
Score-mismatch bias	ε_t^r
Conditional flow matching objective	$\mathcal{L}_{\text{CFM}}$
Diffusion-NFT objective	$\mathcal{L}_{\text{NFT}}$
Contrastive ranking / logistic ranking loss	ℓ_t
Weighted regression / wMSE	ℓ_t^{wMSE}

B.2 Affine Mean of the SDE Transition

We first recall the one-step ODE and SDE updates underlying Eq. (2). With step size $\delta_t > 0$ and $t^- = t - \delta_t$, deterministic ODE sampling uses the Euler update

$$x_{t^-} = x_t - \delta_t \, v_\theta(x_t, t, c). \tag{13}$$

The reverse-time SDE sampler uses the Euler–Maruyama update

$$x_{t^-} = x_t - \delta_t \left[v_\theta(x_t, t, c) + \frac{\sigma_t^2}{2t}\Big(x_t + (1-t) v_\theta(x_t, t, c) \Big) \right] + \sigma_t \sqrt{\delta_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{14}$$

Hence the SDE transition is Gaussian, $q_{\theta,t}(x_{t^-} \mid x_t, c) = \mathcal{N}(\mu_{\theta,t}(x_t, c), \Sigma_t)$ with $\Sigma_t = \sigma_t^2 \delta_t I$, and its mean is the deterministic part of the SDE update.

We now derive the affine mean in Eq. (2). Fix (x_t, t, c) and write $v = v_\theta(x_t, t, c)$. The flow-SDE solver writes the deterministic transition mean through the rectified-flow endpoint estimates

$$x_0^{\text{pred}} = x_t - tv, \qquad x_1^{\text{pred}} = x_t + (1-t)v, \tag{15}$$

with Euler-Maruyama weights

$$w_0 = 1 - t + \delta_t, \qquad w_1 = t - \delta_t - \frac{\sigma_t^2 \delta_t}{2t}. \tag{16}$$

Thus

$$\begin{aligned} \mu_t(v) &= w_0 x_0^{\text{pred}} + w_1 x_1^{\text{pred}} \\ &= w_0 (x_t - tv) + w_1 (x_t + (1-t)v) \\ &= (w_0 + w_1) x_t + \big(-t w_0 + (1-t) w_1 \big) v. \end{aligned} \tag{17}$$

The coefficient of x_t is

$$w_0 + w_1 = (1 - t + \delta_t) + \left(t - \delta_t - \frac{\sigma_t^2 \delta_t}{2t} \right) = 1 - \frac{\sigma_t^2 \delta_t}{2t}, \tag{18}$$

and the coefficient of v is

$$\begin{aligned} -t w_0 + (1-t) w_1 &= -t(1 - t + \delta_t) + (1-t) \left(t - \delta_t - \frac{\sigma_t^2 \delta_t}{2t} \right) \\ &= (-t + t^2 - t\delta_t) + \left(t - t^2 - \delta_t + t\delta_t - \frac{(1-t)\sigma_t^2 \delta_t}{2t} \right) \\ &= -\delta_t - \frac{(1-t)\sigma_t^2 \delta_t}{2t}. \end{aligned} \tag{19}$$

Therefore

$$\mu_t(v) = U_t(x_t, t) + B_t(t) v, \qquad U_t(x_t, t) = \left(1 - \frac{\sigma_t^2 \delta_t}{2t} \right) x_t, \qquad B_t(t) = -\delta_t - \frac{(1-t)\sigma_t^2 \delta_t}{2t}, \tag{20}$$

which is the affine form used in Eq. (2). □

B.3 Log-ratio Form of the Ranking Loss

We prove the log-ratio identity in Eq. (6). For the mirrored branches, $q^{\pm}_{\theta,t}(x_{t^-} \mid x_t, c) = \mathcal{N}(\mu^{\pm}_{\theta,t}, \Sigma_t)$ with shared covariance Σ_t. For a Gaussian transition $q(x) = \mathcal{N}(\mu, \Sigma)$,

$$\log q(x) = -\frac{1}{2}(x - \mu)^\top \Sigma^{-1} (x - \mu) - \frac{1}{2} \log \det(2\pi\Sigma). \tag{21}$$

Subtracting the two branch log-densities cancels the normalization constant:

$$\begin{aligned} \log \frac{q^+_{\theta,t}(x_{t^-} \mid x_t, c)}{q^-_{\theta,t}(x_{t^-} \mid x_t, c)} &= -\frac{1}{2} \left[(x_{t^-} - \mu^+_{\theta,t})^\top \Sigma_t^{-1} (x_{t^-} - \mu^+_{\theta,t}) - (x_{t^-} - \mu^-_{\theta,t})^\top \Sigma_t^{-1} (x_{t^-} - \mu^-_{\theta,t}) \right] \\ &= -\frac{1}{2} \left[\left\| x_{t^-} - \mu^+_{\theta,t} \right\|^2_{\Sigma_t^{-1}} - \left\| x_{t^-} - \mu^-_{\theta,t} \right\|^2_{\Sigma_t^{-1}} \right] \\ &= -\frac{1}{2} \left(E^+_{\theta,t} - E^-_{\theta,t} \right). \end{aligned} \tag{22}$$

The Gaussian normalizers cancel because the two transition distributions have the same covariance. Substituting this identity gives

$$\ell_t(\theta) = \mathrm{softplus}\left(\frac{1}{2}y(E^+_{\theta,t} - E^-_{\theta,t})\right) = \mathrm{softplus}\left(-y\log\frac{q^+_{\theta,t}(x_{t^-} \mid x_t, c)}{q^-_{\theta,t}(x_{t^-} \mid x_t, c)}\right). \tag{23}$$

□

B.4 Proof of Proposition 3.3

Fix (x_t, t, c) and the rollout distribution $x_{t^-} \mid x_t, t, c \sim \mathcal{N}(\mu_t^{\mathrm{old}}, \Sigma_t)$, with $\Sigma_t \succ 0$. Conditioning below suppresses t. Cached states, old velocities, transition coefficients, and scores are held fixed with respect to θ; all required conditional expectations are assumed finite. Let θ_0 satisfy $v_{\theta_0,t} = v_t^{\mathrm{old}}$, and define $J_0 = \partial v_{\theta,t}/\partial\theta|_{\theta=\theta_0}$.

Step 1: Constructed branch means. Let $\mu_t^{\mathrm{old}} = \mu_t(v_t^{\mathrm{old}})$ be the rollout transition mean, $e_t = x_{t^-} - \mu_t^{\mathrm{old}}$ be its residual, and $d_t = \mu^+_{\theta,t} - \mu_t^{\mathrm{old}}$ be the positive-branch displacement. Using $v^\pm_{\theta,t} = v_t^{\mathrm{old}} \pm \beta\Delta v_{\theta,t}$ and the affine mean $\mu_t(v) = U_t + B_t v$ from Eq. (2),

$$d_t = \beta B_t \Delta v_{\theta,t}, \qquad \mu^\pm_{\theta,t} = \mu_t^{\mathrm{old}} \pm d_t. \tag{24}$$

Step 2: Error difference. The variance-normalized transition errors satisfy

$$\begin{aligned} E^+_{\theta,t} - E^-_{\theta,t} &= \|e_t - d_t\|^2_{\Sigma_t^{-1}} - \|e_t + d_t\|^2_{\Sigma_t^{-1}} \\ &= -4\langle\Sigma_t^{-1}e_t, d_t\rangle. \end{aligned} \tag{25}$$

Step 3: Gradient form. Let $z_t = \frac{1}{2}y(E^+_{\theta,t} - E^-_{\theta,t})$. Since $\ell_t(\theta) = \mathrm{softplus}(z_t)$, write $s(z_t) = 1/(1+\exp(-z_t))$ for the sigmoid function, avoiding confusion with the SDE noise scale σ_t. From Eq. (25),

$$z_t = -2y\langle\Sigma_t^{-1}e_t, d_t\rangle. \tag{26}$$

Because the rollout transition mean μ_t^{old} and residual e_t are fixed in the current batch, only $d_t = \beta B_t\Delta v_{\theta,t}$ depends on θ. Thus

$$-\nabla_\theta\ell_t(\theta) = 2\beta\, s(z_t)y\left(\frac{\partial v_{\theta,t}}{\partial\theta}\right)^\top B_t^\top\Sigma_t^{-1}e_t. \tag{27}$$

For a d-dimensional velocity field and p trainable parameters, $J_0 \in \mathbb{R}^{d\times p}$ and $B_t, \Sigma_t \in \mathbb{R}^{d\times d}$; a scalar B_t acts as a multiple of the identity. At $\theta = \theta_0$, $d_t = 0$, so $z_t = 0$ and $s(z_t) = 1/2$ exactly. Hence

$$-\nabla_\theta\ell_t(\theta)|_{\theta=\theta_0} = \beta y J_0^\top B_t^\top\Sigma_t^{-1}e_t. \tag{28}$$

Step 4: Conditional expectation, outcome-conditioned transition-mean difference, and soft-score bias. Let $o \in \{0,1\}$ denote the latent outcome that defines the oracle positive and negative transition splits, and let $r \in [0,1]$ be the optimality score used in the ranking loss, with $y = 2r - 1$. Let

$$\alpha(x_t, c) = \mathbb{P}(o = 1 \mid x_t, c), \qquad \kappa_t(x_t, c) = 2\alpha(x_t, c)\big(1 - \alpha(x_t, c)\big). \tag{29}$$

At θ_0, J_0 and the transition coefficients are fixed given (x_t, t, c). Equation (28) therefore reduces the expected gradient to the conditional moment $\mathbb{E}[ye_t \mid x_t, c]$.

Under the rollout SDE transition, $\mu_t^{\mathrm{old}} = \mathbb{E}[x_{t^-} \mid x_t, c]$, so

$$\mathbb{E}[e_t \mid x_t, c] = 0. \tag{30}$$

Moreover, with the outcome-conditioned transition means $\mu_t^+(x_t, c) = \mathbb{E}[x_{t^-} \mid x_t, c, o = 1]$ and $\mu_t^-(x_t, c) = \mathbb{E}[x_{t^-} \mid x_t, c, o = 0]$ from Section 3.4, the law of total expectation gives

$$\mu_t^{\mathrm{old}} = \alpha(x_t, c)\mu_t^+(x_t, c) + \big(1 - \alpha(x_t, c)\big)\mu_t^-(x_t, c). \tag{31}$$

We first compute the signed residual induced by the latent oracle split. Using Eq. (30),

$$\begin{aligned}\mathbb{E}[(2o-1)e_t \mid x_t, c] &= 2\,\mathbb{E}[oe_t \mid x_t, c] \\ &= 2\alpha(x_t,c)\big(\mu_t^+(x_t,c) - \mu_t^{\text{old}}\big) \\ &= 2\alpha(x_t,c)\big(1-\alpha(x_t,c)\big)\Delta\mu_t^\star(x_t,c) \\ &= \kappa_t(x_t,c)\Delta\mu_t^\star(x_t,c), \end{aligned} \tag{32}$$

where $\Delta\mu_t^\star(x_t,c) = \mu_t^+(x_t,c) - \mu_t^-(x_t,c)$. For non-degenerate states with $\alpha(x_t,c) \in (0,1)$, the scalar $\kappa_t(x_t,c)$ is positive; when α is 0 or 1, the conditional direction degenerates.

For a soft score $r \in [0,1]$, the signed label is $y = 2r-1$. Decomposing it around the latent label gives

$$2r - 1 = (2o-1) + 2(r-o). \tag{33}$$

Therefore,

$$\begin{aligned}\mathbb{E}[ye_t \mid x_t, c] &= \mathbb{E}[(2r-1)e_t \mid x_t, c] \\ &= \mathbb{E}[(2o-1)e_t \mid x_t, c] + 2\mathbb{E}[(r-o)e_t \mid x_t, c] \\ &= \kappa_t(x_t,c)\Delta\mu_t^\star(x_t,c) + \varepsilon_t^r(x_t,c), \end{aligned} \tag{34}$$

where

$$\varepsilon_t^r(x_t,c) = 2\mathbb{E}[(r-o)e_t \mid x_t, c] \tag{35}$$

is the score-mismatch bias. Thus, the oracle direction is still determined by the latent positive and negative transition means, while the soft score contributes an additional term measuring whether the score error $r-o$ is systematically correlated with the local transition residual e_t.

Taking conditional expectation of the exact gradient at θ_0 in Eq. (28) gives

$$\mathbb{E}[-\nabla_\theta \ell_t(\theta) \mid x_t, c]|_{\theta=\theta_0} = \beta J_0^\top B_t^\top \Sigma_t^{-1} \mathbb{E}[ye_t \mid x_t, c]. \tag{36}$$

Substituting Eq. (34) yields

$$\mathbb{E}[-\nabla_\theta \ell_t(\theta) \mid x_t, c]|_{\theta=\theta_0} = \beta J_0^\top B_t^\top \Sigma_t^{-1} \left[\kappa_t(x_t,c)\Delta\mu_t^\star(x_t,c) + \varepsilon_t^r(x_t,c)\right], \tag{37}$$

which proves the soft-score relation in Proposition 3.3.

In the hard-label case $r = o$, we have $\varepsilon_t^r(x_t,c) = 0$, and Eq. (37) reduces to

$$\mathbb{E}[-\nabla_\theta \ell_t(\theta) \mid x_t, c]|_{\theta=\theta_0} = \beta\kappa_t J_0^\top B_t^\top \Sigma_t^{-1} \Delta\mu_t^\star(x_t,c). \tag{38}$$

For $0 < \alpha(x_t,c) < 1$, $\beta\kappa_t > 0$, giving the interpretation in terms of the outcome-conditioned transition-mean difference at the rollout policy.

For posterior-like soft scores $r(x_0,c) = \mathbb{P}(o=1 \mid x_0, c)$, the bias term also vanishes when the intermediate transition provides no additional information about the latent positive outcome once x_0 and c are known,

$$\mathbb{P}(o=1 \mid x_0, c, x_t, x_{t^-}) = \mathbb{P}(o=1 \mid x_0, c). \tag{39}$$

Indeed, under Eq. (39),

$$\begin{aligned}\mathbb{E}[(r-o)e_t \mid x_t, c] &= \mathbb{E}[\mathbb{E}[(r-o)e_t \mid x_0, c, x_t, x_{t^-}] \mid x_t, c] \\ &= \mathbb{E}\big[e_t\big(r - \mathbb{E}[o \mid x_0, c, x_t, x_{t^-}]\big) \mid x_t, c\big] \\ &= \mathbb{E}\big[e_t\big(\mathbb{P}(o=1 \mid x_0, c) - \mathbb{P}(o=1 \mid x_0, c, x_t, x_{t^-})\big) \mid x_t, c\big] = 0. \end{aligned} \tag{40}$$

At the rollout policy, hard binary feedback yields the exact interpretation in terms of the outcome-conditioned transition-mean difference, and soft scores recover the same expected gradient when their mismatch from the latent split is uncorrelated with the local transition residual. This completes the proof of Proposition 3.3. □

Remark: connection to Diffusion-NFT splits. Diffusion-NFT motivates positive and negative branches through distribution splits on final denoised output x_0. The proof above does not require explicitly training those splits: Proposition 3.3 only uses the rollout transition, the mirrored branches, and the outcome-conditioned transition means μ_t^+ and μ_t^-. The oracle positive/negative split remains the latent variable $o \in \{0,1\}$; soft scores $r \in [0,1]$ act as approximate scores for this split and enter the expected gradient at the rollout policy through the score-mismatch bias ε_t^r.

B.5 Finite-Update Analysis under Binary Feedback

Proposition 3.3 gives the exact expected gradient at the rollout policy. Here we retain the full logistic coefficient to characterize finite displacement under binary feedback $r = o$, $y \in \{-1, +1\}$, using the same fixed rollout distribution and cached quantities. General soft scores require retaining score-dependent logistic weights as well as score mismatch.

Step 1: Exact finite-update gradient. Recall from Eq. (24) that

$$e_t = x_{t^-} - \mu_t^{\mathrm{old}}, \qquad d_t = \beta B_t \left(v_{\theta,t} - v_t^{\mathrm{old}}\right), \tag{41}$$

and let

$$a_t = \left\langle \Sigma_t^{-1} e_t, d_t \right\rangle. \tag{42}$$

Lemma 3.2 then gives the exact binary-feedback loss

$$\ell_t(\theta) = \mathrm{softplus}(-2ya_t). \tag{43}$$

These relations do not require d_t to be small. Using

$$2y\, s(-2ya_t) = y - \tanh(a_t), \qquad y \in \{-1, +1\}, \tag{44}$$

where $s(\cdot)$ denotes the sigmoid function, the exact single-sample descent direction is

$$-\nabla_\theta \ell_t = \beta \left(\frac{\partial v_{\theta,t}}{\partial \theta}\right)^\top B_t^\top \Sigma_t^{-1} \left[ye_t - e_t \tanh\left(\left\langle \Sigma_t^{-1} e_t, d_t \right\rangle\right)\right]. \tag{45}$$

The first term is the oracle-aligned component from Proposition 3.3; the second is the finite-update correction induced by the sigmoid multiplier's dependence on the rollout residual.

Step 2: Conditional expectation of the correction. Equation (32) gives

$$\mathbb{E}[ye_t \mid x_t, c] = \kappa_t \Delta\mu_t^\star. \tag{46}$$

Under the rollout SDE, the residual conditioned on (x_t, c) satisfies

$$e_t \mid x_t, c \sim \mathcal{N}(0, \Sigma_t). \tag{47}$$

Applying the multivariate Stein identity

$$\mathbb{E}[e_t f(e_t)] = \Sigma_t \mathbb{E}[\nabla_{e_t} f(e_t)] \tag{48}$$

with $f(e_t) = \tanh(e_t^\top \Sigma_t^{-1} d_t)$ yields

$$\begin{aligned} &\mathbb{E}\left[e_t \tanh\left(e_t^\top \Sigma_t^{-1} d_t\right) \middle| x_t, c\right] \\ &\quad = d_t\, \mathbb{E}_{Z\sim\mathcal{N}(0,1)} \left[\mathrm{sech}^2\left(\left\|\Sigma_t^{-1/2} d_t\right\|_2 Z\right)\right], \end{aligned} \tag{49}$$

where Z is the standardized projection of the rollout residual onto the branch-displacement direction. Define

$$\rho_t := \left\|\Sigma_t^{-1/2} d_t\right\|_2, \qquad \zeta_t := d_t\, \mathbb{E}_{Z\sim\mathcal{N}(0,1)} \left[\mathrm{sech}^2(\rho_t Z)\right]. \tag{50}$$

Here ρ_t is the branch-mean displacement measured in units of the local transition standard deviation, and ζ_t points in the same direction as d_t. Substituting Eqs. (46) and (49) into Eq. (45) gives

$$\mathbb{E}[-\nabla_\theta \ell_t \mid x_t, c] = \beta \left(\frac{\partial v_{\theta,t}}{\partial \theta}\right)^\top B_t^\top \Sigma_t^{-1} \left[\kappa_t \Delta\mu_t^\star - \zeta_t\right]. \tag{51}$$

The finite-update correction $-\zeta_t$ opposes the current displacement in transition-mean space. This damping effect arises from logistic weighting and varies with d_t. It is distinct from both the fixed quadratic penalty in weighted regression and the separately added trust-region regularizer.

Step 3: Local limit and magnitude of the correction. At θ_0, $d_t = 0$ and $\zeta_t = 0$, so Eq. (51) reduces exactly to Proposition 3.3 with $\varepsilon_t^r = 0$. For fixed Σ_t, the correction vanishes as $\rho_t \to 0$, since $\mathbb{E}_Z[\mathrm{sech}^2(\rho_t Z)] \to 1$.

For finite ρ_t, the scalar Stein identity gives

$$\begin{aligned}\left\|\Sigma_t^{-1/2}\zeta_t\right\|_2 &= \rho_t\,\mathbb{E}_Z\left[\mathrm{sech}^2(\rho_t Z)\right] \\ &= \mathbb{E}_Z[Z\tanh(\rho_t Z)]. \end{aligned} \tag{52}$$

Let $g(\rho) = \mathbb{E}_Z[Z\tanh(\rho Z)]$. Since

$$g'(\rho) = \mathbb{E}_Z\left[Z^2\,\mathrm{sech}^2(\rho Z)\right] > 0 \tag{53}$$

for finite $\rho \geq 0$, the correction magnitude increases continuously with ρ_t. Moreover, dominated convergence gives

$$\lim_{\rho\to\infty} g(\rho) = \mathbb{E}_Z|Z| = \sqrt{\frac{2}{\pi}}. \tag{54}$$

Consequently,

$$\left\|\Sigma_t^{-1/2}\zeta_t\right\|_2 \leq \sqrt{\frac{2}{\pi}} \approx 0.798. \tag{55}$$

The deviation from the local result thus grows smoothly and remains bounded.

Step 4: A sufficient alignment condition. Let

$$u_t = \Sigma_t^{-1/2}\Delta\mu_t^\star, \qquad w_t = \Sigma_t^{-1/2}\zeta_t. \tag{56}$$

The inner product between the oracle direction and the bracketed finite-update direction in Eq. (51), both expressed in variance-normalized transition-mean space, satisfies

$$\langle u_t, \kappa_t u_t - w_t\rangle \geq \kappa_t\|u_t\|_2^2 - \|u_t\|_2\|w_t\|_2. \tag{57}$$

It is therefore positive whenever

$$\left\|\Sigma_t^{-1/2}\zeta_t\right\|_2 < \kappa_t\left\|\Sigma_t^{-1/2}\Delta\mu_t^\star\right\|_2. \tag{58}$$

This is a sufficient, not necessary, condition. If it fails, the bound no longer certifies positive alignment, but it does not imply that the direction reverses.

Step 5: Effective update size. Substituting the definition of d_t into Eq. (50) gives

$$\rho_t = \beta\left\|\Sigma_t^{-1/2}B_t\left(v_{\theta,t} - v_t^{\mathrm{old}}\right)\right\|_2. \tag{59}$$

Thus, the accuracy of the local approximation is governed by the normalized displacement ρ_t, which jointly depends on β, the timestep-dependent operator B_t, the SDE covariance, and the actor–rollout-policy gap. The affine relation $\mu_t(v) = U_t + B_t v$ remains exact for any β; increasing ρ_t instead strengthens the bounded finite-update correction in Eq. (51). This characterization is consistent with Figure 3c: $\beta \in \{0.5, 1, 2\}$ maintains strong performance, while $\beta = 5$ degrades performance without complete collapse. The EMA rollout update and trust-region regularization control $v_{\theta,t} - v_t^{\mathrm{old}}$ and therefore help keep ρ_t small.

B.6 Proof of Lemma 3.2

From Eq. (24), we have $\mu_{\theta,t}^{\pm} = \mu_t^{\mathrm{old}} \pm d_t$ and $E_{\theta,t}^{\pm} = \|e_t \mp d_t\|_{\Sigma_t^{-1}}^2$.

For the weighted-regression objective in Eq. (7),

$$\begin{aligned}\ell_t^{\mathrm{wMSE}}(\theta) &= r\|e_t - d_t\|_{\Sigma_t^{-1}}^2 + (1-r)\|e_t + d_t\|_{\Sigma_t^{-1}}^2 \\ &= \|e_t\|_{\Sigma_t^{-1}}^2 + \|d_t\|_{\Sigma_t^{-1}}^2 + 2(1-2r)\langle\Sigma_t^{-1}e_t, d_t\rangle \\ &= \mathrm{const} - 2y\langle\Sigma_t^{-1}e_t, d_t\rangle + \|d_t\|_{\Sigma_t^{-1}}^2, \end{aligned} \tag{60}$$

where $y = 2r - 1$ and $\text{const} = \|e_t\|^2_{\Sigma_t^{-1}}$ is independent of θ. This proves part (a).

For the step-wise ranking loss, Eq. (25) gives

$$E^+_{\theta,t} - E^-_{\theta,t} = -4\langle \Sigma_t^{-1} e_t, d_t \rangle. \tag{61}$$

Substituting into Eq. (6),

$$\ell_t(\theta) = \text{softplus}\left(\frac{1}{2} y (E^+_{\theta,t} - E^-_{\theta,t})\right) = \text{softplus}\left(-2y\langle \Sigma_t^{-1} e_t, d_t \rangle\right), \tag{62}$$

which proves part (b). □

Remark: binary feedback. For $r = 1$, $\ell_t^{\text{wMSE}} = E^+_{\theta,t}$; for $r = 0$, $\ell_t^{\text{wMSE}} = E^-_{\theta,t}$. Thus each binary-labeled sample explicitly optimizes its outcome-selected absolute error, while ranking uses the relative error $E^+_{\theta,t} - E^-_{\theta,t}$. Both objectives use success and failure samples, and both mirrored outputs depend on the same trainable field v_θ, with parameter derivatives $\partial_\theta v^\pm_{\theta,t} = \pm\beta \partial_\theta v_{\theta,t}$. Changes to v_θ therefore affect both recomputed branches.

C EXPERIMENT DETAILS

C.1 DETAILED INTRODUCTION OF BENCHMARKS

We evaluate on two embodied-control benchmark families: LIBERO's four multitask suites and ManiSkill's large compositional pick-and-place task family.

- LIBERO (Liu et al., 2023): We follow the standard protocol across four suites (Spatial, Object, Goal, Long), reporting average success rates over 500 episodes (50 states $\times$ 10 sub-tasks) per suite. The agent receives dual 224$\times$224 RGB inputs, language instructions, and 7-dimensional proprioceptive states (6-DoF joints + gripper). It outputs continuous end-effector actions. The environment provides a sparse binary reward (1 for success, 0 otherwise).
- ManiSkill (Mu et al., 2021): We adopt the PutOnPlateInScene multitask setting from RL4VLA (Liu et al., 2026). This benchmark defines 4,352 compositional tasks derived from 16 objects, 17 receptacles, and 16 tabletop scenes. Observations consist of a 480$\times$640 third-person view, language instructions, and joint poses. Actions are continuous joint-space commands. The environment provides a composite reward to discourage degenerate throwing behaviors. For π-StepNFT, we use only trajectory-level binary success as the optimality signal, without shaped reward components such as the auxiliary 0.1 grasp bonus.

Table 6: OOD task mapping for ManiSkill `PutOnPlateInScene25*` across Vision, Semantics, and Execution categories.

Category	Sub-category (OOD type)	ManiSkill env IDs
Vision	Unseen Table (background)	`VisionImage-v1`
	Dynamic Textures (foreground, weak)	`VisionTexture03-v1`
	Dynamic Textures (foreground, strong)	`VisionTexture05-v1`
	Dynamic Noise (image-level, weak)	`VisionWhole03-v1`
	Dynamic Noise (image-level, strong)	`VisionWhole05-v1`
Semantics	Unseen Objects	`Carrot-v1`
	Unseen Receptacles	`Plate-v1`
	Unseen Instruction Phrasings	`Instruct-v1`
	Multi-Object	`MultiCarrot-v1`
	Distractive Receptacle	`MultiPlate-v1`
Execution	Unseen Position	`Position-v1`
	Unseen Robot Init Pose	`EEPose-v1`
	Mid-Episode Object Reposition	`PositionChangeTo-v1`

To study how long-horizon tasks can benefit when subgoal-level feedback is available, we additionally include a controlled diagnostic.

- CALVIN (Mees et al., 2022): We use CALVIN as a controlled long-horizon diagnostic benchmark with subgoal-level binary feedback. CALVIN evaluates language-conditioned tabletop manipulation, where a single policy must execute multiple natural-language instructions in sequence without resetting between subtasks. Each task contains up to 5 consecutive language-instructed subtasks, with a maximum horizon of 360 environment steps per subtask. For each subtask, the policy keeps attempting the current instruction until it succeeds or reaches the maximum horizon, and then proceeds to the next instruction only after success. This provides a binary completion signal for each subtask, making CALVIN suitable for studying whether intermediate subgoal success/failure feedback can stabilize long-horizon online fine-tuning.

C.2 SFT INITIALIZATION

We initialize each policy from the corresponding π_{RL} SFT checkpoint (Chen et al., 2025). For LIBERO, we use few-shot SFT initializations to prevent performance saturation from masking online RL gains. Specifically, π_0 is initialized from SFT checkpoints trained on pruned subsets of the

total 1,692 LIBERO trajectories: 58 trajectories for Spatial, Object, and Goal, and 208 trajectories for Long. For $\pi_{0.5}$, we use a unified few-shot set of 40 trajectories, corresponding to one demonstration per subtask. For ManiSkill, we use the full 16,384 demonstration trajectories because the PutOnPlateInScene multitask benchmark has higher task complexity.

This initialization protocol keeps LIBERO in a partially solved regime where online RL can expose additional policy improvement, while using a stronger supervised initialization for ManiSkill where sparse multitask generalization is substantially harder.

Table 7: Existing assets used in the experiments. "Not specified" means that the public asset page did not expose separate license metadata.

Asset	Use	Version / ID	License / terms	URL
LIBERO	Benchmark	Spatial/Object/Goal/Long suites	Code: MIT Data: CC BY 4.0	`https://github.com/Lifelong-Robot-Learning/LIBERO`
ManiSkill	Benchmark	ManiSkill 3 release v3.0.1	Apache-2.0	`https://github.com/haosulab/ManiSkill`
CALVIN	Long-horizon diagnostic	Public repository	MIT	`https://github.com/mees/calvin`
OpenPI π_0 / $\pi_{0.5}$	Base VLA policies	`pi0_base` `pi05_base`	Code: Apache-2.0 Gemma terms for Gemma components	`https://github.com/Physical-Intelligence/openpi`
RLinf / π_{RL}	RL framework and SFT checkpoints	RLinf v0.2 Hugging Face checkpoints	Code: Apache-2.0 `Pi05-LIBERO`: Apache-2.0 ManiSkill SFT: MIT `Pi0-LIBERO`: not specified	`https://github.com/RLinf/RLinf` `https://huggingface.co/RLinf`

C.3 Ablation Protocol and Compute

For the component and hyperparameter ablations in Sections 4.3–4.4, each setting is run with five random seeds. Curves show the mean, and shading denotes ± 1 standard deviation across seeds. We use a lightweight ablation configuration to enable rapid iteration: each rollout iteration collects 32 interaction trajectories, uses batch size 256, and sets the actor learning rate to 5.0×10^{-6}. These ablations were conducted on $8\times$ NVIDIA RTX 4090 (48GB) GPUs, with one ablation run taking approximately 2.5 hours.

We choose the ablation task by evaluating the few-shot SFT initialization from Section C.2 on all LIBERO Object subtasks. As shown in Table 8, post-SFT success varies substantially across subtasks. We therefore use LIBERO Object Task 0, *pick up the alphabet soup and place it in the basket*, as the default ablation task: its unsaturated initial success leaves room for online improvement while avoiding a near-zero starting point that would destabilize quick ablations.

Table 8: Initial few-shot SFT success rates on LIBERO Object subtasks used to select the ablation task.

ID	LIBERO Object task	Success rate
0	pick up the alphabet soup and place it in the basket	36.0%
1	pick up the cream cheese and place it in the basket	85.0%
2	pick up the salad dressing and place it in the basket	97.0%
3	pick up the bbq sauce and place it in the basket	72.0%
4	pick up the ketchup and place it in the basket	90.0%
5	pick up the tomato sauce and place it in the basket	79.0%
6	pick up the butter and place it in the basket	94.0%
7	pick up the milk and place it in the basket	69.0%
8	pick up the chocolate pudding and place it in the basket	54.0%
9	pick up the orange juice and place it in the basket	16.0%
	Average	69.2%

To isolate the effect of subgoal feedback from the confounding factors introduced by multi-task training, we use a single CALVIN long-horizon sequence for the subgoal-feedback analysis. Specifically, we select trial ID 0, which consists of the following five-subgoal instruction chain: *take the blue block and rotate it to the right → push the sliding door to the right side → lift the red block from the sliding cabinet → store the grasped block in the sliding cabinet → use the switch to turn off the light bulb*. We analyze task completion by tracking both the average number of completed subtasks

per rollout and the success rate at each sequence length, which together reveal whether a method merely improves early-stage subtasks or consistently advances further along the long-horizon chain. For this analysis, each rollout iteration collects 64 interaction trajectories and uses a batch size of 512. We train for 100 iterations, with the full run taking approximately 6.5 hours.

Main experiments train on full benchmark suites with longer horizons, so we report results averaged over three seeds. They were conducted on $8\times$ NVIDIA H100 (80GB) GPUs.

C.4 Hyperparameters for Training

The table below reports rollout horizons, parallel environments, rollout epochs, denoising steps, and actor-update settings. SFT checkpoint sources and demonstration counts are given in Appendix C.2.

Feedback construction. The default optimality score on LIBERO and ManiSkill is trajectory-level binary success. For soft feedback, let A_i denote the GRPO or GAE advantage. Following Diffusion-NFT (Zheng et al., 2026), we map it to the bounded optimality score

$$r_i = \frac{1}{2} + \frac{1}{2}\,\mathrm{clip}\left(\frac{A_i}{Z}, -1, 1\right), \qquad y_i = 2r_i - 1. \tag{63}$$

Here $Z > 0$ is a fixed scale hyperparameter. We use the same $Z = 3$ for GRPO- and GAE-based feedback in all experiments. The π-StepNFT + Critic variant computes A_i using GAE from a PPO critic while keeping the π-StepNFT actor objective unchanged. In Figure 2e, Norm. GAE and Norm. GAE + wMSE use identical scores r_i and differ only in the actor objective.

Table 9: Hyperparameter settings for Libero and ManiSkill.

Parameters	**LIBERO**								**ManiSkill**	
	π_0				$\pi_{0.5}$				π_0	$\pi_{0.5}$
	Spatial	**Object**	**Goal**	**Long**	**Spatial**	**Object**	**Goal**	**Long**	**Multitask**	**Multitask**
Train epochs	400	400	400	400	400	400	400	400	240	240
Batch size	2048	2048	2048	2048	2048	2048	2048	2048	5120	5120
Update epochs	2	2	4	4	1	1	3	4	5	5
Actor lr	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	8e-6	8e-6
Interaction steps	240	240	320	480	240	240	320	480	60	60
Parallel environments	64	64	64	64	64	64	64	64	320	320
Rollout epochs	8	8	8	8	8	8	8	8	1	1
Action chunk H	5	5	5	10	5	5	5	10	5	5
Denoise steps	4	4	4	4	4	4	4	4	4	4
Noise level σ	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Branch Displacement Scale β	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Initial Decay α_0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
End Decay α_{-1}	0.995	0.995	0.995	0.995	0.995	0.995	0.995	0.995	0.995	0.995

C.5 Hyperparameter Sensitivity Analysis

Figure 3 illustrates the main hyperparameter trade-offs. For noise level σ, excessive noise widens the local transition distribution and impedes convergence, while insufficient noise limits local stochastic exploration. For EMA decay α, high decay causes substantial rollout-policy lag, whereas constant or overly aggressive updates risk collapse. A dynamic schedule balances early improvement with final stability, consistent with the displacement-dependent correction in Appendix B.5. For branch displacement scale β, values in $[1, 2]$ perform best; at a fixed actor–rollout gap, increasing β scales up normalized branch displacement and its finite-update correction, while smaller values weaken the ranking signal. For trust-region weight λ_{TR}, removing regularization reduces performance, whereas overly large values slow convergence; we therefore use $\lambda_{\text{TR}} = 10^{-4}$.

D ADDITIONAL RESULTS

D.1 NOISE VARIANCE AND TRAINING USAGE

We further test whether the gains of SDE exploration can be reproduced by a simpler alternative that injects Gaussian noise only at the final denoised action x_0. This comparison is important because, under our Flow-SDE schedule, the global noise level $\sigma_0 = 0.2$ does not imply that every transition receives standard deviation 0.2. Instead, the actual injected standard deviation is $\sqrt{\delta_t}\sigma_t$, where σ_t follows the solver schedule. With four denoising steps, this gives progressively smaller transition noise and the last step has standard deviation about 0.05. Thus, comparable noise added directly to x_0 might provide a lightweight substitute for SDE exploration.

Table 10: Ablation results under different noise variances and noise-usage strategies.

Variance	**Noisy Actions Used Only for Environment Interaction**		**Noisy Actions Also Used for Training**	
	Decay $\alpha_0 = 0.1$	$\alpha_0 = 0.9$	Decay $\alpha_0 = 0.1$	$\alpha_0 = 0.9$
0.01	fails at startup	fails at startup	0.74	0.78
0.1	-	-	fails at startup	0.60
0.2	-	-	fails at startup	fails at startup

Table 10 shows that this is not the case. When noisy actions are used only for environment interaction while training still targets the original x_0, the policy collapses immediately, consistent with noisy x_0 perturbations producing executions poorly matched by the training signal. When the same noisy actions are also used for training with a variance-normalized correction, performance improves only under very small noise, reaching around 0.74–0.78, which is comparable to deterministic ODE training but far below the step-wise target results in Figure 2b. As the injected noise increases, training again becomes unstable or collapses.

These results suggest that simply perturbing the final action is insufficient in this diagnostic, and that denoising-path noise should be paired with supervision on the corresponding local transitions. Adding noise only to the executed action does not directly supervise adjacent solver transitions, whereas our step-wise formulation fits the stochastic transition that generated the explored state.

D.2 TRAINING-TIME SYSTEM PROFILING

We profile peak allocated memory and wall-clock runtime on 8× NVIDIA A100 (80GB) GPUs. As shown in Table 11, π-StepNFT uses 18,822.0 MiB of peak memory, 5.7% more than Flow-SDE with its lightweight critic (17,812.7 MiB), but 51.5% less than Flow-Noise (38,819.3 MiB), which retains the full solver chain. Its global training iteration takes 286 seconds, compared with 310 seconds for Flow-SDE and 354 seconds for Flow-Noise, corresponding to reductions of 7.7% and 19.2%, respectively. The saving over Flow-Noise is concentrated in actor training, where π-StepNFT updates one sampled solver transition instead of backpropagating through the full multi-step solver chain, reducing actor time from 196 to 146 seconds. By contrast, its actor-training cost is nearly identical to Flow-SDE (146 versus 147 seconds), and the smaller overall time reduction is primarily observed during rollout generation (132 versus 151 seconds).

Table 11: System profiling per global training iteration on 8× NVIDIA A100 (80GB) GPUs. Peak allocated memory is reported in MiB and stage times in seconds. Overall iteration time includes rollout generation, actor training, weight synchronization, and other minor overheads.

Metric	π-StepNFT	Flow-SDE	Flow-Noise
Peak allocated memory (MiB)	18,822.0	17,812.7	38,819.3
Rollout generation (s)	132	151	149
Actor training (s)	146	147	196
Overall iteration (s)	286	310	354

D.3 FULL MANISKILL OOD RESULTS

For ManiSkill, we follow the PutOnPlateInScene OOD taxonomy used by RL4VLA and the π_{RL} grouping convention, placing instruction-phrasing shifts together with visual perturbations under Visual-Language Variations, since both preserve the underlying manipulation objective while changing the observation/language surface form. Our primary ManiSkill baselines are the π_{RL} Flow-SDE and Flow-Noise variants. To keep the comparison protocol conservative and traceable, we use the π_{RL} results reported by the original authors for the variants covered in their evaluation, rather than mixing reported baseline numbers with separately re-run numbers in the same table. Consequently, for additional RL4VLA-style variants that are not reported in π_{RL}, including Carrot-v1, Plate-v1, and EEPose-v1, we mark the corresponding π_{RL} entries as unavailable and report π-StepNFT only for completeness of our evaluation coverage. All direct category averages and comparisons are computed only over the shared variants for which results are available for every compared method.

Table 12: Success rates (%) on ManiSkill for π_0 and $\pi_{0.5}$ across in-distribution and OOD variants.

Environment	**Variation-Version-Type**	π_0-SFT	π_0-RL Flow-SDE	π_0-RL Flow-Noise	π_0-StepNFT	$\pi_{0.5}$-SFT	$\pi_{0.5}$-RL Flow-SDE	$\pi_{0.5}$-RL Flow-Noise	$\pi_{0.5}$-StepNFT
In distribution	Main-v3-train	38.4	**78.8**	77.8	73.6	40.1	**90.9**	89.7	84.3
Visual-Language Variations	Instruct-v1-test	30.1	64.6	**66.5**	62.5	46.6	77.0	**85.7**	82.9
	VisionImage-v1-test	38.3	68.8	71.7	**71.8**	46.2	78.8	**83.1**	80.5
	VisionTexture03-v1-test	35.1	66.0	66.8	**72.7**	36.7	69.6	75.0	**80.5**
	VisionTexture05-v1-test	31.0	55.8	60.5	**63.1**	32.7	58.0	62.2	**71.2**
	VisionWhole03-v1-test	35.4	62.4	69.0	**70.2**	40.1	69.6	71.6	**82.0**
	VisionWhole05-v1-test	28.5	49.0	53.9	**54.7**	30.7	55.0	57.0	**62.5**
	Visual Avg. (5)	33.7	60.4	64.4	**66.5**	37.3	66.2	69.8	**75.3**
	Visual-Language Avg. (6)	33.1	61.1	64.7	**65.8**	38.8	68.0	72.4	**76.6**
Semantic Reasoning (object/receptacle confounders)	MultiCarrot-v1-test	7.8	**28.2**	23.0	26.0	16.7	36.8	38.2	**39.1**
	MultiPlate-v1-test	5.0	16.4	**18.3**	17.6	11.8	**29.4**	28.3	21.9
	Carrot-v1-test	23.5	-	-	58.3	31.2	-	-	65.2
	Plate-v1-test	38.0	-	-	72.7	39.1	-	-	78.9
	Avg. over shared variants	6.4	**22.3**	20.7	21.8	14.3	33.1	**33.3**	30.5
Action Execution	PositionChangeTo-v1-test	9.6	17.4	10.9	**19.5**	13.5	36.2	**54.7**	28.1
	Position-v1-test	16.9	**45.6**	37.5	39.8	31.2	54.5	55.0	**56.3**
	EEPose-v1-test	11.7	-	-	29.3	18.6	-	-	49.1
	Avg. over shared variants	13.3	**31.5**	24.2	29.7	22.4	45.4	**54.9**	42.2

E LIMITATIONS

Online RL for large flow-based VLAs still requires many environment interactions, while real-robot rollouts are slow and difficult to parallelize. The sampling efficiency of the current method therefore does not yet support direct online training on real hardware. Improving interaction efficiency and validating the resulting method on physical robots remain important directions for future work.