FASTER: Value-Guided Sampling for Fast RL

Perry Dong* **Alexander Swerdlow*** **Dorsa Sadigh** **Chelsea Finn**
Stanford University
{perryd, aswerdlo}@stanford.edu

Abstract

Some of the most performant reinforcement learning algorithms today can be prohibitively expensive as they use test-time scaling methods such as sampling multiple action candidates and selecting the best one. In this work, we propose FASTER, a method for getting the benefits of sampling-based test-time scaling of diffusion-based policies without the computational cost by tracing the performance gain of action samples back to earlier in the denoising process. Our key insight is that we can model the denoising of multiple action candidates and selecting the best one as a Markov Decision Process (MDP) where the goal is to progressively filter action candidates before denoising is complete. With this MDP, we can learn a policy and value function in the denoising space that predicts the downstream value of action candidates in the denoising process and filters them while maximizing returns. The result is a method that is lightweight and can be plugged into existing generative RL algorithms. Across challenging long-horizon manipulation tasks in online and batch-online RL, FASTER consistently improves the underlying policies and achieves the best overall performance among the compared methods. Applied to a pretrained VLA, FASTER achieves the same performance while substantially reducing training and inference compute requirements.

Code: https://github.com/alexanderswerdlow/faster

1 Introduction

Recent reinforcement learning (RL) methods have demonstrated strong performance using expressive policy backbones such as diffusion models which are widely used in domains such as image/video generation and robotics. However, some of the most performant of these algorithms are prohibitively expensive both during training and at test time. This is because many of these algorithms rely on sampling multiple candidate actions and selecting the best one, for example as determined by which one has the highest value. While effective, such approaches incur substantial computational costs. This is especially problematic for large models like modern Vision-Language-Action (VLA) models, which already operate near the limits of acceptable latency. A modest sample count of eight will multiply inference costs accordingly and can render these methods impractical in latency-sensitive or resource-constrained settings. In this work, we ask: can we recover the benefits of sampling-based test-time scaling without suffering its computational cost?

This high computational cost comes from policies with generative models such as diffusion or flow matching that need to denoise all candidates. Prior works have shown that test-time scaling methods for generative models are performant. Methods such as best-of-N sampling and self-consistency sample multiple candidates and select the one that maximizes a given metric. While these methods are effective at maximizing performance, they scale poorly and require full denoising over all action candidates. Distillation-based approaches amortize this cost by training the policy to directly reproduce high-value behaviors, but require training a separate policy which can be expensive. The core difficulty underlying all of these approaches is that, without observing the fully denoised samples, it is hard to know which ones are worth executing since there is no direct supervision linking a given

*Equal contribution.

Accepted at ICML 2026 Workshop on Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning.

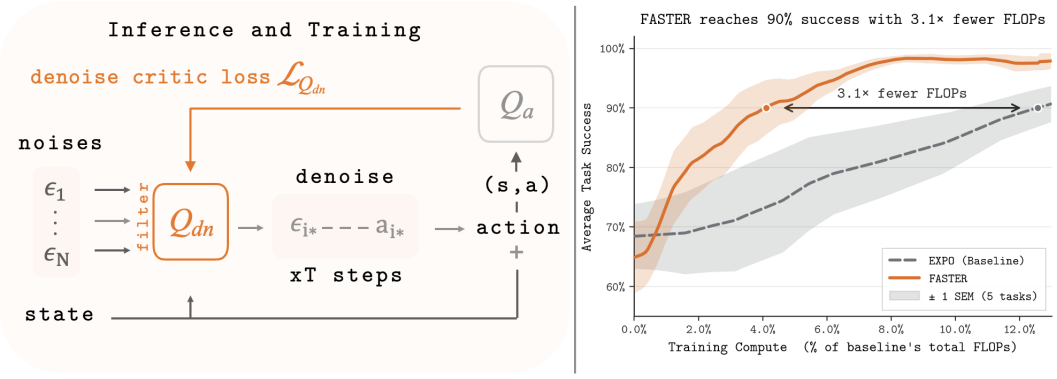

Figure 1: **Left: Overview of FASTER.** Instead of denoising all N candidates and selecting the best action post-hoc (best-of-N), FASTER learns a denoise critic Q^{dn} that scores action samples *during* denoising, often directly on the initial noise itself. **Right: Performance of FASTER compared with baseline** averaged across tasks for $\pi_{0.5}$. The FLOPs are normalized on the x-axis against the amount of FLOPs needed for the baseline to converge.

noise sample to its downstream outcome, and the mapping from noise to action is only defined implicitly through the denoising process.

In this work, our key insight is that we can model the denoising of multiple candidate samples and the selection of the best one as a Markov Decision Process (MDP) where the goal is to progressively filter candidates before denoising is complete. With this MDP, we can learn a denoise Q-function and policy with traditional temporal difference learning that decides which actions to keep and remove while maximizing the returns. To supervise the value function without requiring exhaustive rollouts, we use the value function over final actions as the reward of the action filtering MDP, enabling the model to propagate outcome information back to the denoising space without direct outcome labels for every candidate. Practically, we find that filtering candidates at the noise level—which is the cheapest instantiation computationally—achieves performance equivalent to fully denoising all action samples and selecting the highest value action. The computational cost of denoising—which is the dominant bottleneck at inference time—is thus reduced to that of a single rollout, regardless of the number of action candidates. The result is a lightweight, modular approach that decouples the benefits of broad action sample selection from the cost of full policy execution.

Our main contribution is a framework for obtaining the performance gains of sampling and selecting the best of multiple actions of RL methods without the computational overhead of denoising all actions. Unlike approaches that operate on actions, our method operates directly on the denoising process to filter down candidates, making selection both cheap and modular. We evaluate our method on challenging manipulation benchmarks, including Robomimic and LIBERO, across algorithms and model capacities, demonstrating consistent gains over single-sample baselines and near-parity with best-of-N methods at a fraction of the computational overhead.

2 Related Work

Test-time scaling and best-of-N inference. Significant work has shown test-time scaling methods for generative models to be highly performant. These methods take several forms, including iterative refinement, but the most common involve sampling multiple candidates and performing filtering or refinement based on a group of samples. Best-of-N sampling is the most standard of such methods, with extensive use in many domains including in autoregressive language models [1, 2, 3], diffusion models [4], and policy learning [5, 6, 7, 8].

Diffusion policies and value-guided sampling. Diffusion models have emerged as a strong policy class for a wide variety of domains because they can represent rich, multimodal data distributions [5, 6, 9, 10, 11, 12, 13, 14]. Several recent works study how to improve these policies with value functions or reduce their inference cost. Value-Guided Policy Steering (V-GPS) re-ranks actions from frozen robot policies with a learned value function [15], RoboMonkey scales test-time sampling and verification for vision-language-action policies [16], and EXPO couples an expressive base policy with a lightweight edit policy and selects value-maximizing actions online [5]. Other work amortizes or changes the policy itself: One-Step Diffusion Policy distills multi-step denoising into a

faster actor [17], while DSRL post-trains a behavior-cloned diffusion policy by running RL in its latent-noise space [18]. Compared to these approaches, our method learns a separate critic over *which actions to filter during denoising*, enabling candidate selection before actions are denoised. As such, our method can be directly applied to methods that can benefit from sampling multiple actions.

Initial-noise and noise-space methods for diffusion models. A rapidly growing body of literature shows that the initial seed of a diffusion model has a significant effect on generation quality. Prior work has shown that seed quality is linked to inversion stability [19, 20] and several methods explicitly optimize or learn better starting noises, including InitNO and FIND [21, 22], as well as plug-and-play noise refinement methods such as Golden Noise [23], NoiseRefine [24], and Noise-Level Guidance [25]. Noise Hypernetworks likewise seek to amortize test-time noise optimization into a learned auxiliary module [26]. More closely related, TTSnap [27] prunes candidate trajectories early by decoding intermediate estimates and scoring them with noise-aware reward models trained via self-distillation, allowing it to search over larger candidate sets. This is similar in spirit to our goal of recovering the benefits of best-of-N without denoising every candidate, but the mechanism is distinct. TTSnap still performs partial denoising and reward evaluation at selected timesteps, and is studied in text-to-image generation with image-reward objectives from pretrained models. By contrast, we frame the problem of filtering action candidates as an MDP and learn a *state-conditioned* critic that predicts downstream value directly from the initial noise of a diffusion policy in the context of reinforcement learning.

3 Preliminaries

We consider problems modeled as Markov decision processes (MDPs), each defined by a tuple $\{\mathcal{S}, \mathcal{A}, r, \gamma, T, \rho\}$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $T(s'|s,a)$ comprises the transition dynamics, $\gamma \in [0,1]$ is the discount factor, and $\rho(s)$ is the initial state distribution. In this MDP, the RL agent observes state s_t at every timestep t and chooses action a_t from its policy $\pi(a_t|s_t)$. The tuple of states, actions, rewards, and next states (s, a, r, s') is appended to a replay buffer D for training. The goal of RL is to maximize the expected sum of discounted returns $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$.

In this paper, we study the setting where the policy π is a diffusion or flow policy that requires denoising. We further focus on algorithms that perform test-time scaling through sampling multiple action candidates. Given N initial noise seeds $\{\epsilon_i\}_{i=1}^N$, the policy denoises them into N actions $\{a_i\}_{i=1}^N$. The policy then selects the best action among the candidate set, for example by choosing $i^* = \arg\max_i Q(s, a_i)$ and executing a_{i^*}. Our main goal is to capture the performance gain from these sampling-based test-time scaling approaches without the computational cost of denoising all N action candidates.

4 FASTER

To capture the benefits of test-time scaling without its computational overhead, we leverage a key insight: the sample variance that drives these gains can be traced to early in the denoising process. We introduce FASTER, a framework that models denoising as an MDP and learns a filtering policy to select promising actions before they are denoised. An overview is shown in Figure 1. We first define the filtering MDP (Section 4.1), then describe policy learning within this MDP (Section 4.2). In practice, we find that sample variance is largely determined by the initial noise, motivating a practical instantiation that filters at the noise level — the earliest and computationally cheapest point in the denoising chain. We describe this and further practical implementations in Section 4.3.

4.1 Modeling Action Denoising as an MDP

We want to select the best action candidates *before* denoising completes. To do so, we model the reverse diffusion process as a filtering MDP over N noise candidates $\{\epsilon_i\}_{i=1}^N$, $\epsilon_i \sim \mathcal{N}(0, I)$, where a policy progressively discards unpromising candidates at each denoising step. We illustrate this in Figure 2.

Algorithm 1 Learning procedure in FASTER

Require: s, π_θ, Q^{dn}, Q^a, N

Sample $\{\epsilon_i\}_{i=1}^N$ with $\epsilon_i \sim \mathcal{N}(0, I)$
$i^* \leftarrow \arg\max_i Q^{dn}(s, \epsilon_i)$
$a_{i^*} \leftarrow \pi_\theta(s, \epsilon_{i^*})$
$\mathcal{L}_{\text{reg}} = \left\| Q^{dn}(s, \epsilon_{i^*}) - \text{sg}[Q^a(s, a_{i^*})] \right\|_2^2$

Algorithm 2 Inference procedure in FASTER

Require: s, π_θ, Q^{dn}, N

Sample $\{\epsilon_i\}_{i=1}^N$ with $\epsilon_i \sim \mathcal{N}(0, I)$
$i^* \leftarrow \arg\max_i Q^{dn}(s, \epsilon_i)$
$a_{i^*} \leftarrow \pi_\theta(s, \epsilon_{i^*})$
Execute a_{i^*}

States. A state $\mathbf{s}_t = (s, t, \mathcal{C}_t, \{a_i^{(t)}\}_{i \in \mathcal{C}_t})$ consists of the environment state s, the denoising timestep $t \in \{T, \ldots, 1\}$, the surviving candidate set $\mathcal{C}_t \subseteq \{1, \ldots, N\}$, and the partially denoised intermediates, from pure noise $a_i^{(T)} = \epsilon_i$ to a clean action at t=1.

Actions. At each step the policy retains or discards each candidate: $m_{t,i} \in \{0, 1\}$ for $i \in \mathcal{C}_t$, with at least one retained ($\sum_i m_{t,i} \geq 1$). The surviving set becomes $\mathcal{C}_{t-1} = \{i : m_{t,i} = 1\}$.

Transitions. A denoising step is applied to each candidate, producing $a_i^{(t-1)}$. The episode ends when a single candidate remains ($|\mathcal{C}_{t-1}| = 1$) or denoising finishes ($t = 1$), at which point the highest-scoring survivor is executed. The episode may terminate at $t > 1$, in which case the remaining candidate is denoised until $t = 1$.

Reward. Non-terminal steps receive zero reward. At termination the reward is $Q^a(s, a_{i^*})$, where a_{i^*} is the executed action.

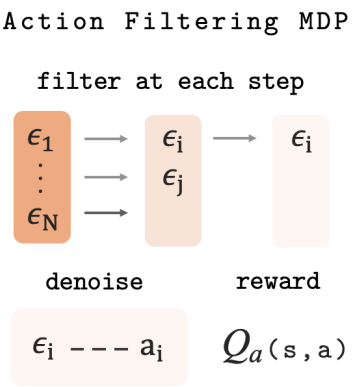

Figure 2: **Action Filtering MDP.** We model the process of denoising action candidates and selecting the best one as an MDP where the goal is to filter action samples during denoising while maximizing returns.

4.2 Learning the filtering policy

To learn the filtering policy in the MDP above, we learn an action-value function over filtering decisions. Concretely, we learn the denoise critic $Q^{dn}(\mathbf{s}_t, \mathbf{m}_t)$, where $\mathbf{s}_t = (s, t, \mathcal{C}_t, \{a_i^{(t)}\}_{i \in \mathcal{C}_t})$ is the filtering state and $\mathbf{m}_t = \{m_{t,i}\}_{i \in \mathcal{C}_t}$ is the filtering action with $m_{t,i} \in \{0, 1\}$ denoting whether candidate i is retained. We train Q^{dn} with a temporal-difference (TD) objective.

$$\mathcal{L}_{\text{TD}} = \mathbb{E}_{(\mathbf{s}_t, \mathbf{m}_t, \mathbf{s}_{t-1})} \left[\left(Q^{dn}(\mathbf{s}_t, \mathbf{m}_t) - \left[r_t + \gamma \max_{\mathbf{m}_{t-1}} \bar{Q}^{dn}(\mathbf{s}_{t-1}, \mathbf{m}_{t-1}) \right] \right)^2 \right] \tag{1}$$

where $r_t = 0$ for non-terminal transitions and, at termination, $r_t = Q^a(s, a_{i^*})$ for the executed action a_{i^*} induced by the final surviving candidate.

At test time, we sample N noise candidates $\{\epsilon_i\}_{i=1}^N$ with $\epsilon_i \sim \mathcal{N}(0, I)$ and repeatedly apply the filtering policy induced by Q^{dn} to choose filtering actions $\mathbf{m}_t$ until a single candidate remains. Let i^* denote the surviving candidate index. That survivor is then denoised to obtain the final action $a_{i^*} = \pi_\theta(s, \epsilon_{i^*})$. Depending on the method (Section 4.3), we either use a_{i^*} as the final action or consider this alongside a proposal action from an edit policy.

4.3 Practical implementation

In our setting, we restrict the MDP to only filter at $t = T$, reducing the multi-step filtering process to a single decision. Since the filtering happens before any denoising, the state collapses to just the environment state s (the denoising timestep is fixed at T and there is no history of prior filtering steps), and the filtering "action" reduces to selecting a single candidate rather than producing a binary mask $\mathbf{m}_t$ over survivors. Under this simplification, the joint $Q^{dn}(\mathbf{s}_t, \mathbf{m}_t)$ from Section 4.2 decomposes into per-candidate scores: we learn a *noise-level critic* $Q^{dn}(s, \epsilon)$ that maps an environment state s and a single noise sample $\epsilon \sim \mathcal{N}(0, I)$ to the expected return of denoising that sample. We denote the standard action-space critic as the *action-level critic* $Q^a(s, a)$, which provides the supervision

signal. We use a greedy filtering policy which selects $i^* = \arg\max_i Q^{dn}(s, \epsilon_i)$ and denoises ϵ_{i^*} into $a_{i^*} = \pi_\theta(s, \epsilon_{i^*})$.

Because the MDP is now one step, the TD objective in Section 4.2 simplifies to a one-step regression against the action-level critic. We sample N noise candidates $\{\epsilon_i\}_{i=1}^N$ with $\epsilon_i \sim \mathcal{N}(0, I)$, select the highest-scoring index $i^* = \arg\max_i Q^{dn}(s, \epsilon_i)$, denoise ϵ_{i^*} to obtain $a_{i^*} = \pi_\theta(s, \epsilon_{i^*})$, and supervise Q^{dn} by regressing to the action-level critic:

$$\mathcal{L}_{\text{reg}} = \left\| Q^{dn}(s, \epsilon_{i^*}) - \text{sg}\left[Q^a(s, a_{i^*})\right] \right\|_2^2 .$$

The regression-based learning and inference procedures are summarized in Algorithms 1 and 2. At test time, we select $i^* = \arg\max_{i \in \{1,\ldots,N\}} Q^{dn}(s, \epsilon_i)$ and denoise only the selected candidate ϵ_{i^*} to obtain the final action $a_{i^*} = \pi_\theta(s, \epsilon_{i^*})$.

We evaluate FASTER with two sampling-based methods in our experiments.

FASTER-EXPO. We implement FASTER on top of EXPO, a high-performing online RL method that leverages a diffusion base policy combined with a lightweight edit policy. In this case, the selected denoised action a_{i^*} serves as the base-policy proposal, and the downstream edit step follows EXPO unchanged.

FASTER-IDQL. We implement FASTER on top of IDQL, which performs implicit action selection from a diffusion policy without an edit policy. In this case, the denoised action a_{i^*} is executed directly.

4.4 Computational profile

The key computational advantage of FASTER over best-of-N sampling is that it replaces N full denoising rollouts with N cheap noise-level critic evaluations followed by a single denoising rollout. Let T denote the number of denoising steps, N the number of candidates, and F_{actor}, F_{critic} the FLOPs for a single forward pass of the actor and critic, respectively. Standard best-of-N sampling denoises all N candidates for T steps each and then scores the resulting actions:

$$C_{\text{BoN}} = T \cdot N \cdot F_{\text{actor}} + N \cdot F_{\text{critic}}.$$

FASTER instead scores all N noise candidates with the noise critic and denoises only the top-scoring one:

$$C_{\text{FASTER}} = N \cdot F_{\text{critic}} + T \cdot F_{\text{actor}}.$$

The savings come from reducing the actor cost from $\mathcal{O}(TN)$ to $\mathcal{O}(T)$, eliminating $(N-1)$ full denoising passes. When $T = N$ and $F_{\text{actor}} = F_{\text{critic}} = F$, the cost reduces from $T(T+1) \cdot F$ to $2T \cdot F$—an approximate $T/2\times$ speedup.

In practice, T typically ranges from 5 to 20 denoising steps. For example, EXPO [5] uses T=10 and N=8. While one-step diffusion methods [28, 29] attempt to distill multi-step generations into a single-step process, state-of-the-art diffusion models in robotics, image generation, and video generation [30, 31, 32, 33] continue to use multi-step sampling in this range, making the cost reduction from FASTER broadly applicable.

In our primary experiments, the actor and critic share the same architecture and FLOP count ($F_{\text{actor}} = F_{\text{critic}}$), consistent with baselines such as [5] and recent work on large-scale RL [34]. For our VLA experiments, the asymmetry is even more favorable: the critic has 20M parameters versus the actor's 3.3B, so $F_{\text{critic}} \ll F_{\text{actor}}$ and the N noise-level evaluations add negligible overhead.

5 Experiments

The goal of our experiments is to answer the following core questions:

- **(Q1)** How does FASTER perform compared to state-of-the-art RL methods in both the online setting and the batch-online setting?
- **(Q2)** Is FASTER able to recover the performance gains of test-time scaling while reducing the computational cost?
- **(Q3)** Can FASTER be applied to a pretrained Vision-Language-Action (VLA) model?
- **(Q4)** What components of FASTER are most important for performance?

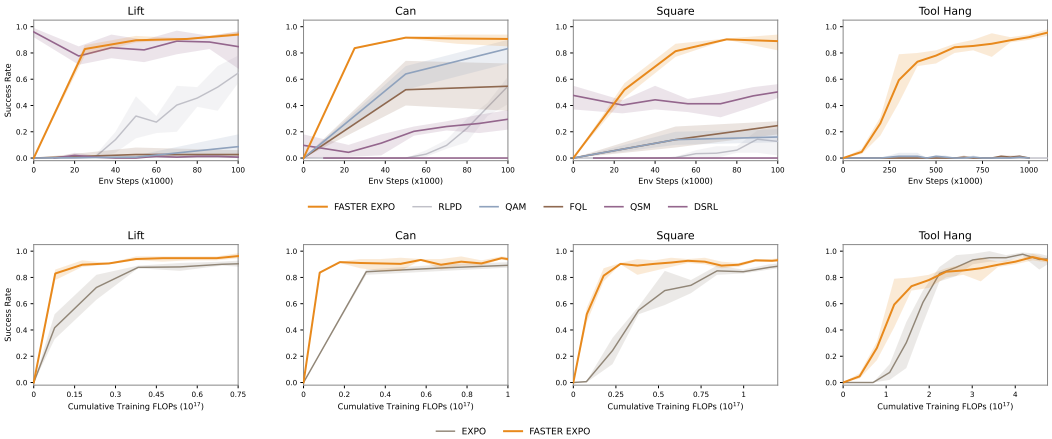

Figure 3: **Top: Success rates** of FASTER and baselines in the online settings. FASTER-EXPO outperforms strong baselines in sample efficiency. **Bottom: Compute comparisons** of FASTER-EXPO and EXPO. FASTER eliminates extra denoising during training and inference, yielding large FLOP reductions relative to the EXPO with comparable task performance.

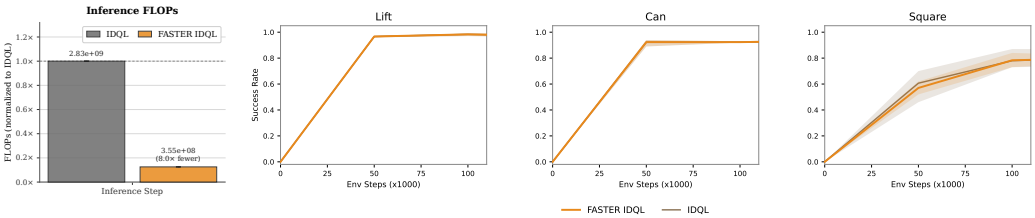

Figure 4: **Success rate and compute comparisons** of FASTER-IDQL and IDQL in the online setting. FASTER can be applied to IDQL to eliminate extra denoising rollouts at inference while obtaining the same performance in success rates (see Figure 3).

Environments. We evaluate FASTER on a set of 9 challenging manipulation tasks from Robomimic and LIBERO. All tasks feature a sparse reward indicating task completion. These environments involve controlling a 7-DoF robot arm to complete complex manipulation tasks. For Robomimic, we evaluate on `Lift`, `Can`, `Square`, and `Tool Hang`, which require picking up a block, picking up a can and placing it, inserting a tool onto a square peg, and hanging a tool on a rack, respectively. For LIBERO, we start from the pretrained `pi05_libero` checkpoint from OpenPI [30], which is trained on `libero_goal`, `libero_object`, `libero_spatial`, and `libero_10` but not `libero_90`; we then select all held-out `libero_90` tasks on which the original policy achieves 40–60% success and run RL on those tasks without any offline data.

Baselines. We compare our method to prior state-of-the-art methods for online and batch-online reinforcement learning [35].

EXPO [5]. EXPO is an online RL method that jointly learns an expressive diffusion policy alongside a lightweight Gaussian edit policy that edits the actions sampled from the base policy toward a higher value distribution.

IDQL [6]. IDQL trains an expressive diffusion policy via imitation learning and uses implicit policy extraction by performing best-of-N sampling, selecting the action that maximizes the Q-value.

RLPD [36]. RLPD is a highly sample-efficient algorithm that leverages prior data and samples from it for learning. RLPD uses a simpler Gaussian policy and has been shown to outperform many offline-to-online methods even without pretraining. For both evaluation settings, we run RLPD without offline pretraining.

QSM [14]. QSM is an online RL method that trains diffusion policies by matching the diffusion loss to action gradients. QSM aims to avoid instability of value propagation to the expressive policy by incorporating losses to guide the denoising process.

DSRL [18]. DSRL is an online RL method that adapts a frozen diffusion-based behavior cloning policy by performing RL over the initial noise seed used for sampling. DSRL learns a clipped Gaussian policy that predicts the initial noise.

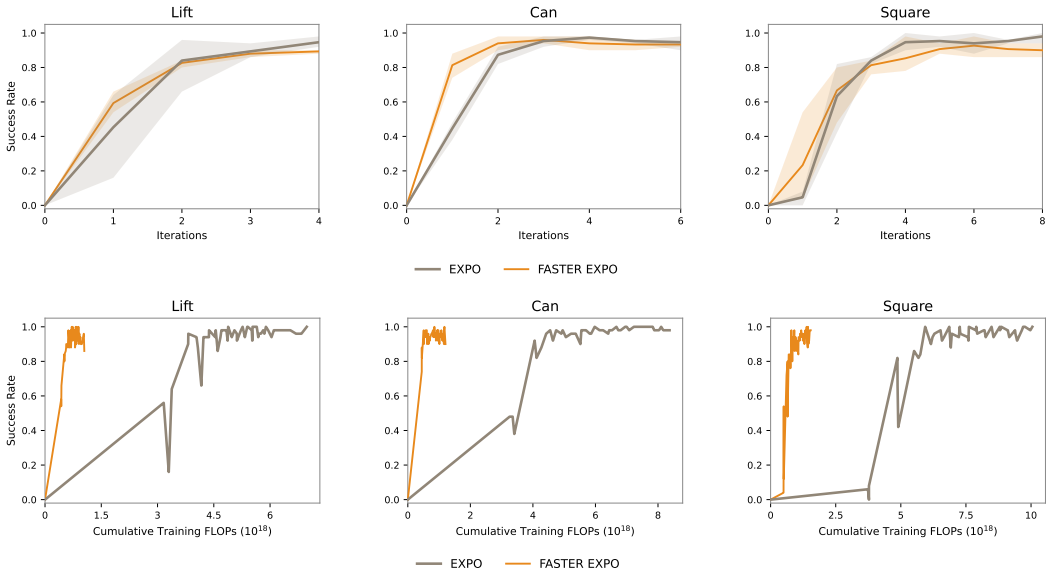

Figure 5: **Top: Success rate curves** of FASTER-EXPO and EXPO in the batch-online setting. FASTER matches the performance of EXPO in iterations. **Bottom: Compute comparisons** of FASTER-EXPO and EXPO in the batch-online setting. Like in the online setting, in the batch-online setting FASTER-EXPO yields a large FLOP reduction compared to EXPO from not needing to denoise all action samples.

QAM [37]. QAM is an online RL method that trains diffusion policies with adjoint matching. QAM aims to avoid instability of value propagation to the expressive policy by using action gradients to form a step-wise objective function that is free from unstable backpropagation.

FQL [12]. FQL uses a one-step flow policy to maximize the Q estimates learned with the standard TD objective. It also incorporates a behavioral regularization term toward a BC flow policy.

5.1 How does FASTER perform compared to state-of-the-art RL methods in both the online setting and the batch-online setting?

We present the main results in Figure 3 and Figure 5. Across both the online and batch-online settings, FASTER-EXPO achieves the strongest overall performance among all baselines. In the online setting, FASTER-EXPO significantly outperforms even highly sample-efficient methods such as RLPD. Methods that rely on action gradients to directly train an expressive policy, such as QSM and QAM, consistently struggle to match this performance. DSRL, which steers diffusion policies through the noise latent space, improves steadily but exhibits worse sample efficiency than methods that optimize directly in the action space. FQL, despite leveraging a one-step flow policy, consistently underperforms as behavioral regularization toward a BC policy can hinder exploration in the online setting. In the batch-online setting, FASTER-EXPO achieves the same performance as EXPO when comparing iterations, and outperforms EXPO in terms of compute efficiency. Overall, FASTER achieves state-of-the-art performance across all evaluated methods.

5.2 Is FASTER able to recover the performance gains of test-time scaling while reducing the computational cost?

We now evaluate whether the noise-level critic can identify high-quality noise seeds *before* denoising, avoiding the redundant computation of fully denoising all N candidates. We compare FASTER-EXPO and FASTER-IDQL against their unfiltered counterparts, EXPO and IDQL, which rely on best-of-N sampling at inference time. As shown in Figure 3 and Figure 4, both FASTER variants achieve task success rates within the margin of error of their respective baselines across the evaluation suite in the online setting, confirming that the noise-level critic effectively recovers the performance gains of test-time scaling. Crucially, this comes at a fraction of the computational cost: as detailed in Section 4.4 and shown empirically in Figure 3, Figure 4 and Figure 6, FASTER reduces inference-time actor forward passes from $\mathcal{O}(TN)$ to $\mathcal{O}(T)$ by replacing N full denoising rollouts with N lightweight critic evaluations followed by a single denoising pass. Concretely, the update step goes from $11.6\,\mathrm{s}$ to

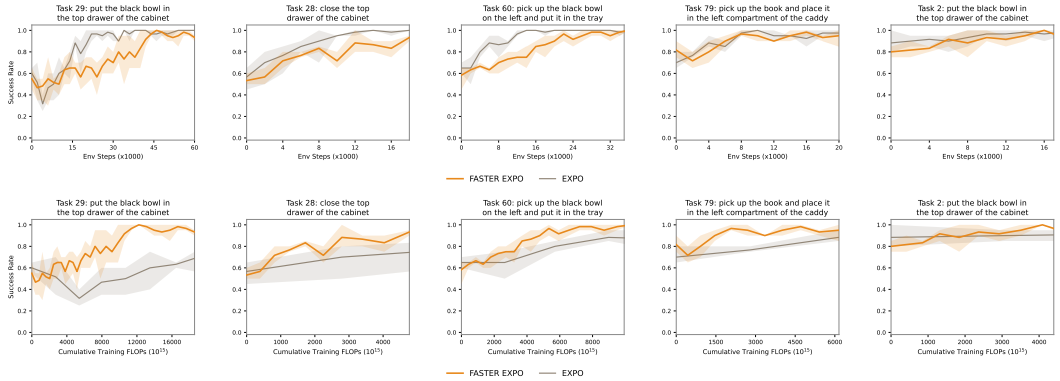

Figure 7: FASTER-EXPO **compared to EXPO on top of $\pi_{0.5}$. Top: Performance with environment steps.** FASTER-EXPO is competitive in performance compared to EXPO. **Bottom: Performance with FLOPs.** FASTER-EXPO performs significantly better than EXPO under the same compute as FASTER-EXPO chooses the best action sample without denoising all sampled actions in inference and training.

2.5 s and the inference step goes from 566 ms to 335 ms. These results demonstrate that the sample variance exploited by best-of-N selection is largely determined at the noise level, and that FASTER can capture this signal without incurring the cost of exhaustive denoising. We further discuss the computational tradeoffs between our methods and baselines in Section C.1.

5.3 Can FASTER be applied to a pretrained Vision-Language-Action (VLA) model?

The computational burden of sampling-based methods is particularly acute as model scale increases; recent VLAs commonly reach 3B parameters, making the cost of denoising N candidates at every environment step prohibitive for both training and deployment. We therefore evaluate whether FASTER can be applied to a pretrained VLA to recover the benefits of best-of-N sampling at substantially lower cost. Concretely, we apply FASTER-EXPO to the 3.3B-parameter `pi05_libero` checkpoint from OpenPI [30] and fine-tune it with online RL on 5 held-out `libero_90` tasks, comparing against EXPO under identical conditions.

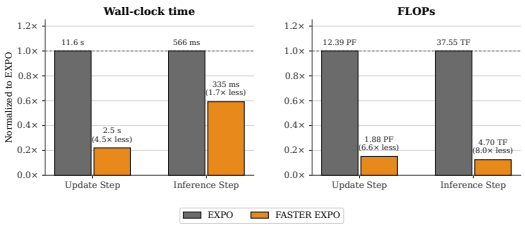

Figure 6: **Training and inference timing** of FASTER-EXPO compared to EXPO. FASTER-EXPO achieves 1.7x improvement in inference time and 4.5x improvement in the update step time.

Compute savings. As shown in Figure 6, FASTER-EXPO reduces the per-step update time by approximately 4.5× relative to EXPO, from roughly 11.6 s to 2.5 s. This speedup stems from the fact that FASTER only denoises a single candidate through the 3.3B-parameter actor during each training step, whereas EXPO must denoise all N candidates to score them with the env-level critic. At inference time, FASTER-EXPO reduces latency from 566 ms to 335 ms per action, a 1.7× speedup. The gains are even more pronounced in FLOPs: FASTER-EXPO requires 4.70×10^{12} inference FLOPs per step compared to 3.75×10^{13} for EXPO—an 8× reduction (Figure 6). The theoretical tradeoff described in Section 4.4 is especially favorable in the VLA setting because the noise-level critic (20M parameters) is far smaller than the actor (3.3B parameters), so scoring N noise candidates adds negligible overhead relative to a single denoising pass.

Task performance. Despite these substantial compute reductions, FASTER-EXPO matches EXPO on the majority of the 5 evaluated tasks (Figure 7, top). Across the task suite, FASTER-EXPO achieves comparable aggregate success rates to EXPO while requiring a fraction of the compute budget. These results demonstrate that FASTER scales naturally to large pretrained VLAs, enabling efficient online RL fine-tuning without sacrificing the performance benefits of sampling-based action selection.

5.4 What components of FASTER are most important for performance?

To better understand how different pieces of FASTER contribute to performance, we ablate over three key components: (1) the size of the denoise critic, (2) which step to filter with the denoise critic, and (3) learning the full filtering policy according to the MDP in Section 4.1. We present additional

experiments on comparing FASTER against distilling the value maximizing distribution in Section A as another approach for reducing computational cost.

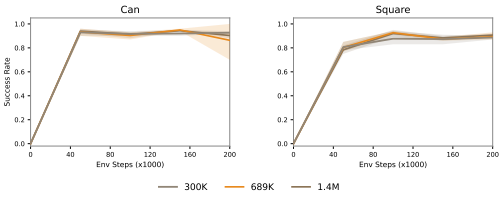

Figure 8: **Critic-size ablation** for FASTER-EXPO on `can` and `square`. We compare filtering critics Q^{dn} with parameter counts set to approximately $1.0\times$, $0.5\times$, and $0.25\times$ that of Q^a. We find that Q^{dn} can be substantially smaller than Q^a without degrading performance.

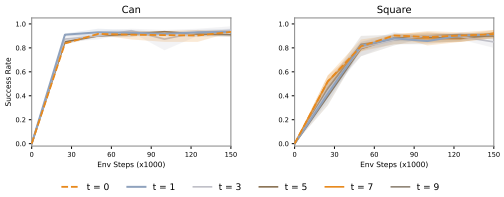

Figure 9: **Filtering step ablation.** FASTER-EXPO achieves similar performance across different filtering steps, suggesting FASTER does not require precise selection of the filtering step and filtering at the initial seed is effective.

How does the size of the denoise critic affect performance? We investigate whether the capacity of the denoise critic is a significant factor in performance. Specifically, we compare architectures ranging from a smaller network (128 hidden units × 3 layers, 300K params) to a medium network (176 hidden units × 3 layers, 689K params) to a larger network (256 hidden units × 3 layers, 1.4M params) on the Robomimic Can and Square tasks. As shown in Figure 8, performance remains largely consistent across all configurations, suggesting that FASTER is robust to moderate variations in critic capacity and does not require careful tuning of this hyperparameter. Importantly, this suggests FASTER can be used with a smaller denoise critic compared to action critic, which can greatly reduce computational cost in the case of large critic networks.

How does the choice of filtering step affect performance? In our experiments, we implement FASTER by filtering only at the initial noise. A natural question is whether the specific denoising timestep at which the critic filter is applied has a meaningful impact on performance. We evaluate several choices of filtering step on the Robomimic Can and Square tasks. As reported in Figure 9, performance is consistent across the range of steps considered, indicating that FASTER does not require precise selection of the filtering timestep and is robust to this design choice; filtering at the initial seed achieves the best performance relative to the computation required.

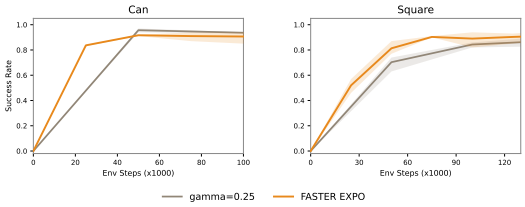

Figure 10: **Learned-filter ablation results.** Filtering at the initial seed performs comparably to learning the full filtering policy in the MDP.

Policy learning with the full filtering MDP. The complete formulation of FASTER learns a filtering policy that can filter at any step of the denoising process. In our experiments, we reduce this multi-step filtering process to a single decision and implement filtering at the initial noise level, as this approach incurs minimal computational overhead while achieving performance on par with fully denoising all action candidates. Here, we present results for the general setting in which the filtering policy is learned to adaptively select the denoising step at which filtering occurs. In this case, γ implicitly encourages the policy to terminate at earlier timesteps to save compute. Evaluating on the Robomimic Can and Square tasks, we find that the learned adaptive policy performs comparably to fixed filtering at the initial noise level. This suggests the single-step simplification of filtering at the initial seed serves as a good proxy for solving the full MDP.

6 Discussion

In this work, we propose FASTER, a method for obtaining the benefits of sampling-based test-time scaling without incurring its computational cost. By framing the denoising of multiple action candidates as an MDP, FASTER learns a filtering policy that selects actions according to their downstream value. Despite these promising results, FASTER has limitations. First, while FASTER substantially improves upon its base algorithms in computational efficiency, it is not designed to improve sample efficiency. In our experiments, FASTER-EXPO achieves better sample efficiency relative to strong baselines, owing largely to the inherent sample efficiency of EXPO, but does not surpass EXPO itself in this regard. Improving the sample efficiency of filtering-based approaches is an interesting direction for future work. Furthermore, FASTER is designed for policy classes that use initial noise seeds,

which encompass some of the most performant policy representations in contemporary reinforcement learning. Extending this to policy classes that lack such a seed structure is left to future work.

References

[1] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171, 2022. doi: 10.48550/arXiv.2203.11171. URL https://arxiv.org/abs/2203.11171.

[2] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314, 2024. doi: 10.48550/arXiv.2408.03314. URL https://arxiv.org/abs/2408.03314.

[3] Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-N sampling in large language models. *CoRR*, abs/2412.15287, 2024. doi: 10.48550/arXiv.2412.15287. URL https://arxiv.org/abs/2412.15287.

[4] Nanye Ma, Shangyuan Tong, Haolin Jia, Hexiang Hu, Yu-Chuan Su, Mingda Zhang, Xuan Yang, Yandong Li, Tommi Jaakkola, Xuhui Jia, and Saining Xie. Inference-time scaling for diffusion models beyond scaling denoising steps. *CoRR*, abs/2501.09732, 2025. doi: 10.48550/arXiv.2501.09732. URL https://arxiv.org/abs/2501.09732.

[5] Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. EXPO: Stable reinforcement learning with expressive policies. *CoRR*, abs/2507.07986, 2025. doi: 10.48550/arXiv.2507.07986. URL https://arxiv.org/abs/2507.07986.

[6] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit q-learning as an actor-critic method with diffusion policies. *CoRR*, abs/2304.10573, 2023. doi: 10.48550/arXiv.2304.10573. URL https://arxiv.org/abs/2304.10573.

[7] Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows, 2026. URL https://arxiv.org/abs/2510.07650.

[8] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling, 2023. URL https://arxiv.org/abs/2209.14548.

[9] Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *CoRR*, abs/2208.06193, 2022. doi: 10.48550/arXiv.2208.06193. URL https://arxiv.org/abs/2208.06193.

[10] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *CoRR*, abs/2303.04137, 2023. doi: 10.48550/arXiv.2303.04137. URL https://arxiv.org/abs/2303.04137.

[11] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning, 2023. URL https://arxiv.org/abs/2305.13122.

[12] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu, editors, *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/park25f.html.

[13] Andrew Wagenmaker, Perry Dong, Raymond Tsao, Chelsea Finn, and Sergey Levine. Posterior behavioral cloning: Pretraining bc policies for efficient rl finetuning, 2025. URL https://arxiv.org/abs/2512.16911.

[14] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[15] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *CoRR*, abs/2410.13816, 2024. doi: 10.48550/arXiv.2410.13816. URL https://arxiv.org/abs/2410.13816.

[16] Jacky Kwok, Christopher Agia, Rohan Sinha, Matthew Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. *CoRR*, abs/2506.17811, 2025. doi: 10.48550/arXiv.2506.17811. URL https://arxiv.org/abs/2506.17811.

[17] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *CoRR*, abs/2410.21257, 2024. doi: 10.48550/arXiv.2410.21257. URL https://arxiv.org/abs/2410.21257.

[18] 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. *CoRR*, abs/2506.15799, 2025. doi: 10.48550/arXiv.2506.15799. URL https://arxiv.org/abs/2506.15799.

[19] Yuanhao Ban, Ruochen Wang, Tianyi Zhou, Boqing Gong, Cho-Jui Hsieh, and Minhao Cheng. The crystal ball hypothesis in diffusion models: Anticipating object positions from initial noise. *CoRR*, abs/2406.01970, 2024. doi: 10.48550/arXiv.2406.01970. URL https://arxiv.org/abs/2406.01970.

[20] Zipeng Qi, Lichen Bai, Haoyi Xiong, and Zeke Xie. Not all noises are created equally: Diffusion noise selection and optimization. *CoRR*, abs/2407.14041, 2024. doi: 10.48550/arXiv.2407.14041. URL https://arxiv.org/abs/2407.14041.

[21] Xiefan Guo, Jinlin Liu, Miaomiao Cui, Jiankai Li, Hongyu Yang, and Di Huang. InitNO: Boosting text-to-image diffusion models via initial noise optimization. *CoRR*, abs/2404.04650, 2024. doi: 10.48550/arXiv.2404.04650. URL https://arxiv.org/abs/2404.04650.

[22] Changgu Chen, Libing Yang, Xiaoyan Yang, Lianggangxu Chen, Gaoqi He, Changbo Wang, and Yang Li. FIND: Fine-tuning initial noise distribution with policy optimization for diffusion models. *CoRR*, abs/2407.19453, 2024. doi: 10.48550/arXiv.2407.19453. URL https://arxiv.org/abs/2407.19453.

[23] Zikai Zhou, Shitong Shao, Lichen Bai, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. *CoRR*, abs/2411.09502, 2024. doi: 10.48550/arXiv.2411.09502. URL https://arxiv.org/abs/2411.09502.

[24] Donghoon Ahn, Jiwon Kang, Sanghyun Lee, Jaewon Min, Minjae Kim, Wooseok Jang, Hyoungwon Cho, Sayak Paul, SeonHwa Kim, Eunju Cha, Kyong Hwan Jin, and Seungryong Kim. A noise is worth diffusion guidance. *CoRR*, abs/2412.03895, 2024. doi: 10.48550/arXiv.2412.03895. URL https://arxiv.org/abs/2412.03895.

[25] Harvey Mannering, Zhiwu Huang, and Adam Prügel-Bennett. Noise-level diffusion guidance: Well begun is half done. *CoRR*, abs/2509.13936, 2025. doi: 10.48550/arXiv.2509.13936. URL https://arxiv.org/abs/2509.13936.

[26] Luca Eyring, Shyamgopal Karthik, Alexey Dosovitskiy, Nataniel Ruiz, and Zeynep Akata. Noise hypernetworks: Amortizing test-time compute in diffusion models. *CoRR*, abs/2508.09968, 2025. doi: 10.48550/arXiv.2508.09968. URL https://arxiv.org/abs/2508.09968.

[27] Qingtao Yu, Changlin Song, Minghao Sun, Zhengyang Yu, Vinay Kumar Verma, Soumya Roy, Sumit Negi, Hongdong Li, and Dylan Campbell. TTSnap: Test-time scaling of diffusion models via noise-aware pruning. *CoRR*, abs/2511.22242, 2025. doi: 10.48550/arXiv.2511.22242. URL https://arxiv.org/abs/2511.22242.

[28] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

[29] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models, 2025. URL https://arxiv.org/abs/2410.12557.

[30] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A Vision-Language-Action Model with Open-World Generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[31] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

[32] Yu Gao, Lixue Gong, Qiushan Guo, Xiaoxia Hou, Zhichao Lai, Fanshi Li, Liang Li, Xiaochen Lian, Chao Liao, Liyang Liu, Wei Liu, Yichun Shi, Shiqi Sun, Yu Tian, Zhi Tian, Peng Wang, Rui Wang, Xuanda Wang, Xun Wang, Ye Wang, Guofeng Wu, Jie Wu, Xin Xia, Xuefeng Xiao, Zhonghua Zhai, Xinyu Zhang, Qi Zhang, Yuwei Zhang, Shijia Zhao, Jianchao Yang, and Weilin Huang. Seedream 3.0 technical report, 2025. URL https://arxiv.org/abs/2504.11346.

[33] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[34] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

[35] Perry Dong, Suvir Mirchandani, Dorsa Sadigh, and Chelsea Finn. What matters for batch online reinforcement learning in robotics?, 2025. URL https://arxiv.org/abs/2505.08078.

[36] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023.

[37] Qiyang Li and Sergey Levine. Q-learning with adjoint matching, 2026. URL https://arxiv.org/abs/2601.14234.

[38] Perry Dong, Alec M. Lessing, Annie S. Chen, and Chelsea Finn. Reinforcement learning via implicit imitation guidance, 2025. URL https://arxiv.org/abs/2506.07505.

[39] Perry Dong, Kuo-Han Hung, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. Tql: Scaling q-functions with transformers by preventing attention collapse, 2026. URL https://arxiv.org/abs/2602.01439.

[40] 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.

[41] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

[42] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

[43] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas, 2025. URL https://arxiv.org/abs/2512.15692.

A Additional Experiments

A natural question is whether the best-of-N sampling benefits observed in EXPO and IDQL can be recovered by a single policy trained via distillation. Concretely, an alternative to FASTER would proceed as follows: sample N noise candidates $\{\epsilon_i\}_{i=1}^N$ with $\epsilon_i \sim \mathcal{N}(0, I)$, denoise each to obtain actions $a_i = \pi_\theta(s, \epsilon_i)$, select the highest-scoring index $i^* = \arg\max_i Q^a(s, a_i)$, and distill the selected action a_{i^*} into a new policy π'_θ by supervising against a_{i^*}. We evaluate this distillation baseline in Figure 11 and find that it performs substantially worse than FASTER. We attribute this gap to the instability inherent in distillation: the distilled policy must continually chase a moving target defined by the Q-function, yielding a non-stationary training distribution that destabilizes learning. By contrast, FASTER operates directly in noise space, framing selection as an easier filtering problem that is more amenable to stable optimization.

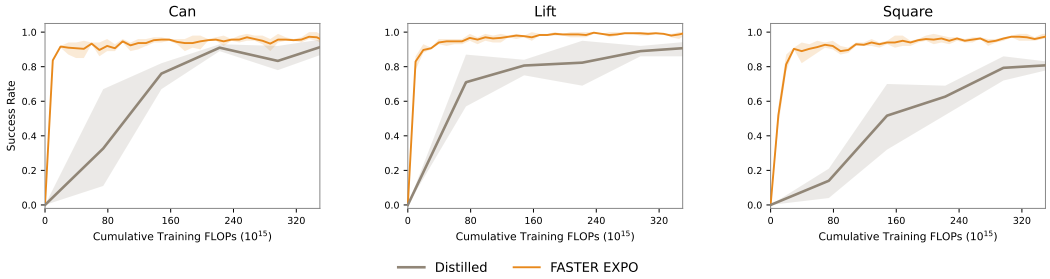

Figure 11: **Distillation ablation results.** Distilling the value maximizing distribution into a policy performs significantly worse compared to FASTER.

B Experiment Details

Hyperparameter	FASTER-EXPO	FASTER-IDQL
Actor / critic learning rate	3×10^{-4} / 3×10^{-4}	3×10^{-4} / 3×10^{-4}
Batch size	256	256
Discount	0.99	0.99
Target update τ	0.005	0.005
UTD ratio	20	20
Candidates N	8	8
Candidates kept after filtering	1	1
Denoising steps T	10	100
Filter temperature mode	z-score	z-score
Expectile	—	0.8
Edit scale	0.15	—

Table 1: **Hyperparameters** for the FASTER-EXPO and FASTER-IDQL variants on Robomimic.

Datasets and evaluation. For the Robomimic experiments, all tasks use sparse binary rewards and report success rate. In the online setting, we do not pretrain on offline data, but we retain the same task conventions used in prior work to define task difficulty: `Lift` uses a 10-episode subset of the original dataset to make it more challenging, `Can` uses the multi-human dataset, and `Square` and `Tool Hang` use the standard proficient-human split. In the batch-online setting, we initialize training from these same demonstration sources and then continue interaction online. For the batch-online setting, we use $T = 10$ denoising steps for `Lift` and `Can`, and $T = 100$ denoising steps for `Square`. For the final rerun FASTER-EXPO experiments, evaluation is performed every 50k environment steps on `Lift`, `Can`, and `Square`, and every 100k steps on `Tool Hang`; FASTER-IDQL uses the same evaluation protocol. Evaluation-time filtering with Q^{dn} is greedy (temperature 0) for all Robomimic tasks, while at training time we sample candidates from a softmax over z-score–normalized Q-values with temperature 1.0 for `Lift`, `Can`, and `Square`, and 0.1 for `Tool Hang`. Unless otherwise specified, line plots report the mean success rate over three seeds and the shaded regions denote the max and min.

VLA experiments. For the pretrained VLA experiments, we follow the held-out LIBERO protocol described in the main text and fine-tune entirely online, without mixing in offline data. We use a batch size of 32, UTD ratio 10, and 100k online training steps. We use $N = 8$ candidates, retain a single candidate after filtering for FASTER-EXPO, and denoise for $T = 10$ steps. The VLA predicts action chunks over 10 steps and we execute the entire chunk before replanning. Online optimization is performed at the episode level rather than every environment step: after each completed episode, we run three updates on newly sampled replay minibatches, with each update containing UTD 10 critic updates. To best utilize the replay data, we train on all sliding window chunks. For this setting, the critic is lightweight relative to the actor, so the additional cost of evaluating multiple noise candidates remains small compared to a full multi-candidate denoising pass. Both training-time and evaluation-time candidate selection use greedy sampling (temperature 0), and we use an edit scale of 0.2.

Implementation Details. We run experiments on a mix of NVIDIA A40, L40S, and H100 GPUs. All timing experiments are conducted on L40S GPUs. For update step timing, we consider the time taken for a single critic update, with other parameters as described in the prior paragraph. For inference step timing, we report BS=1 inference.

C Computational Profile

C.1 Comparison to non-diffusion-based methods

Our work focuses on capturing the benefits of sampling-based methods for diffusion models without incurring the costs of these approaches. Diffusion and flow-based models are the dominant policy class in robotics, particularly in the real-world with VLAs [40, 41] and WAMs [42, 43], hence our focus. Other methods, such as the Gaussian policies used in RLPD [36], are generally more efficient due to lower typical parameter counts and no iterative computation—i.e., only a single forward step at inference. However, their limited expressivity makes them ill-suited for many real-world applications. Thus, this work focuses on the performance of diffusion methods with the same parameter count, which we believe to be the most apt comparison.