



# Falcon: Fast Visuomotor Policies via Partial Denoising

Haojun Chen<sup>\*1,2</sup> Minghao Liu<sup>\*3</sup> Chengdong Ma<sup>1</sup> Xiaojian Ma<sup>2</sup> Zailin Ma<sup>4</sup> Huimin Wu<sup>2</sup> Yuanpei Chen<sup>1,5</sup>  
Yifan Zhong<sup>1,5</sup> Mingzhi Wang<sup>1</sup> Qing Li<sup>2</sup> Yaodong Yang<sup>1</sup>

## Abstract

Diffusion policies are widely adopted in complex visuomotor tasks for their ability to capture multimodal action distributions. However, the multiple sampling steps required for action generation significantly harm real-time inference efficiency, which limits their applicability in real-time decision-making scenarios. Existing acceleration techniques either require retraining or degrade performance under low sampling steps. Here we propose Falcon, which mitigates this speed-performance trade-off and achieves further acceleration. The core insight is that visuomotor tasks exhibit sequential dependencies between actions. Falcon leverages this by reusing partially denoised actions from historical information rather than sampling from Gaussian noise at each step. By integrating current observations, Falcon reduces sampling steps while preserving performance. Importantly, Falcon is a training-free algorithm that can be applied as a plug-in to further improve decision efficiency on top of existing acceleration techniques. We validated Falcon in 48 simulated environments and 2 real-world robot experiments, demonstrating a 2-7x speedup with negligible performance degradation, offering a promising direction for efficient visuomotor policy design. The code is available at <https://github.com/chjchjchjchj/Falcon>.

## 1. Introduction

Diffusion policies have demonstrated remarkable success in addressing complex visuomotor tasks in robotics (Chi et al.,

<sup>\*</sup>Equal contribution <sup>1</sup>Institute for Artificial Intelligence, Peking University <sup>2</sup>National Key Laboratory of General Artificial Intelligence, BIGAI <sup>3</sup>School of Electronic Engineering and Computer Science, Peking University <sup>4</sup>School of Mathematical Sciences, Peking University <sup>5</sup>PKU-PsiBot Joint Lab. Correspondence to: Qing Li <[dylan.liqing@gmail.com](mailto:dylan.liqing@gmail.com)>, Yaodong Yang <[yaodong.yang@pku.edu.cn](mailto:yaodong.yang@pku.edu.cn)>.

*Proceedings of the 42<sup>nd</sup> International Conference on Machine Learning*, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

2023; Reuss et al., 2023; Ze et al., 2024b;a; Ravan et al., 2024; Yang et al.), thanks to their ability to model complex multimodal distributions and maintain stable training dynamics. Essentially, diffusion policies rely on the reverse sampling process of a stochastic differential equation (SDE) (Song et al., b; Karras et al., 2022), where actions are iteratively sampled starting from a standard normal distribution. Each sampling step involves drawing a sample from a Brownian motion distribution, incrementally denoising the initial sample to generate the final action. However, the iterative sampling process required for action generation makes these methods computationally slow, particularly when applied to sequential decision-making in real-time environments (Chen et al., b; Janner et al., 2022; Wang et al., b; Yang et al., 2023; Hansen-Estruch et al., 2023; Chen et al., a).

To address these challenges, existing work (Song et al., a;b; Lu et al., 2022; Zhang & Chen) reformulates the sampling process as an ordinary differential equation (ODE) and uses numerical solvers to reduce the number of denoising steps (Song et al., a; Lu et al., 2022; Zhang & Chen). However, these solvers often suffer from approximation errors when using very few steps, degrading policy performance (Zhao et al., 2024; Wang et al., a). Alternatively, distillation-based approaches (Salimans & Ho; Song et al., 2023; Kim et al.; Prasad et al., 2024; Wang et al., 2024) accelerate inference via one-step generation. However, they typically require task-specific retraining and may degrade multimodal expressiveness (Prasad et al., 2024), making them less flexible and unsuitable as plug-and-play solutions for diverse tasks. Another related approach is Streaming Diffusion Policy (SDP) (Høeg et al., 2024), which also uses partial denoising for acceleration but requires task-specific retraining and a large noise-time buffer, limiting flexibility and memory efficiency.

In this work, we introduce Falcon (**F**ast visuomotor policies via **a**lternative **D**enoising), a novel approach designed to bridge the gap between acceleration and performance preservation in diffusion-based visuomotor policies. The key insight behind Falcon is that sequential dependencies in visuomotor tasks can be exploited to accelerate action generation while maintaining multimodal expressiveness. To effectively leverage this property, we first utilize the previous action prediction as the reference action, then introduce a thresholding

mechanism combined with one-step estimation to evaluate which partial denoising actions are dependent on current timesteps in parallel. Using a temperature-scaled softmax, we then select the most suitable partial denoising action to continue the sampling process, preserving both performance and efficiency. By avoiding the conventional practice of starting the denoising process from a standard normal distribution at each decision step, Falcon begins from partial denoised actions derived from historical observations, significantly reducing the number of sampling steps required. Importantly, Falcon is a training-free method, which allows it to be applied as a plug-in module, enhancing the efficiency of existing diffusion-based policies without additional training or extensive modifications. It integrates seamlessly with solvers like DDIM (Song et al., a) and DPMSSolver (Lu et al., 2022) to further enhance acceleration.

We evaluate Falcon in both simulated and real-world environments. In simulation, we test Falcon across 48 tasks spanning five widely used benchmarks, including RoboMimic (Mandlekar et al., 2022), RoboSuite Kitchen (Gupta et al., 2020), BlockPush (Shafiullah et al., 2022), MetaWorld (Yu et al., 2020) and Maniskill2 (Gu et al., 2023). For real-world experiments, we deploy Falcon on a dual-arm robot platform in two manipulation tasks—dexterous grasping and high-precision insertion. In both settings, Falcon achieves a **2–7×** acceleration in action sampling while preserving high task success rates, validating its effectiveness and practical applicability.

In summary, this work makes three main contributions: **First**, we introduce Falcon, which leverages sequential dependencies in visuomotor tasks to perform partial denoising, significantly reducing the number of sampling steps while preserving the ability to model multimodal action distributions. **Second**, Falcon functions as a training-free plug-in that enhances existing diffusion-based policies, integrating seamlessly with solvers like DDIM and DPMSSolver to further accelerate action generation. **Third**, we demonstrate Falcon’s effectiveness across 48 simulated environments and 2 real-world robot tasks, achieving a **2–7×** speedup in inference while maintaining high action quality.

## 2. Preliminaries

In this section, we briefly introduce the diffusion models, how the diffusion models are used for diffusion policies in visuomotor tasks, the acceleration techniques in the diffusion-based models, and terminology.

**Diffusion Models.** Diffusion models, such as DDPM (Ho et al., 2020), are generative models that learn data distributions by progressively corrupting data through noise in a forward process and then reconstructing it in a reverse denoising process. In the forward process, a data point  $\mathbf{x}_0$

is corrupted K timesteps, resulting in:

$$q(\mathbf{x}_k \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_k; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_k) \mathbf{I}), \quad (1)$$

where  $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$ . After  $K$  steps,  $\mathbf{x}_K$  becomes nearly Gaussian. The reverse process reconstructs  $\mathbf{x}_0$  by iteratively denoising, modeled as

$$p_\theta(\mathbf{x}_{k-1} \mid \mathbf{x}_k) = \mathcal{N}(\mathbf{x}_{k-1}; \mu_\theta(\mathbf{x}_k, k), \sigma_t^2 \mathbf{I}). \quad (2)$$

where  $\mu_\theta$  is the predicted mean,  $\sigma_t^2$  is fixed according to the forward process,  $\alpha_i$  is the variance schedule in the forward diffusion process.

**Diffusion Policies.** Diffusion policy (Chi et al., 2023) extends diffusion models as a powerful policy for visuomotor tasks. At timestep  $t$ , diffusion policy takes the latest  $T_o$  steps of observation  $\mathbf{O}_t$  as input, predicts  $T_p$  action sequence  $\mathbf{A}_t$  and executes  $T_a$  action sequence. The action sequence generation process is a conditional denoising diffusion process modeled by  $p_\theta(\mathbf{A}_t \mid \mathbf{O}_t)$ , where  $\mathbf{A}_t \in \mathbb{R}^{T_p \times D_a}$  and  $\mathbf{O}_t \in \mathbb{R}^{T_o \times D_o}$ ,  $D_a$  and  $D_o$  represent the action and observation’s dimension respectively. Specifically, starting from a pure Gaussian noise sample, diffusion policy leverage the noise prediction network  $\epsilon_\theta$  to predict and remove noise at each denoising step, iterating for  $K$  steps to generate a clean sample  $\mathbf{A}_t^0$ . The action sequence generation process is described as Eq. 3, where  $\mathbf{A}_t^k$  is a partial denoising action in the timestep  $t$  with noise level  $k$ ,  $\mathbf{Z}$  is a standard Gaussian random variable.

$$\mathbf{A}_t^{k-1} = \frac{1}{\sqrt{\alpha_k}} \left( \mathbf{A}_t^k - \frac{1 - \alpha_k}{\sqrt{1 - \bar{\alpha}_k}} \epsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k) \right) + \sigma_k \mathbf{Z}. \quad (3)$$

The training loss is

$$L(\theta) = \mathbb{E}_{k, \mathbf{A}_t^0, \epsilon} \left[ \|\epsilon - \epsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k)\|^2 \right] \quad (4)$$

$$\mathbf{A}_t^k = \sqrt{\bar{\alpha}_k} \mathbf{A}_t^0 + \sqrt{1 - \bar{\alpha}_k} \epsilon$$

**Acceleration techniques in the diffusion-based models.** Some popular works (Song et al., b; Karras et al., 2022; Song et al., a; Lu et al., 2022) interpret the diffusion model as an ODE. Specifically, DDIM (Song et al., a) generalizes the Markov forward process in DDPM to a non-Markov process, allowing the use of shorter Markov chains during sampling. The iterative steps of its sampling process can be rewritten in a form similar to Euler integration, which is a discrete solution process for a specific ODE. DPMSSolver (Lu et al., 2022) accelerates sampling by analytically computing the linear part of the diffusion ODE solution and using an exponential integrator to approximate the nonlinear part. But these methods can reduce the quality degradation of few-step sampling (Shih et al., 2024). SDP (Høeg et al., 2024) improves the sampling speed by generating partial denoised actions with different noise levels, but relies on

handcrafted noise schedules and large buffers, limiting its flexibility. Consistency Policy (CP) (Prasad et al., 2024) distills a pretrained diffusion policy into a one-step sampler via self-consistency on its probability flow ODE. While fast, this requires retraining and may compromise multimodality.

We write  $[a, b]$  to denote the set  $\{a, a+1, \dots, b-1\}$ ,  $\mathbf{x}_{a:b}$  to denote the set  $\{\mathbf{x}_i : I \in [a, b]\}$  and  $[K]$  to denote the set  $\{1, \dots, K\}$ . We define  $\mathbf{A}_t = [\mathbf{a}_{t:t+T_a}, \tilde{\mathbf{a}}_{t+T_a:t+T_p}]$ , where  $\mathbf{a}_{t:t+T_a}$  is the executed part and  $\tilde{\mathbf{a}}_{t+T_a:t+T_p}$  is the unexecuted part in timestep  $t$ .

### 3. Falcon: Fast Visuomotor Policies via Partial Denoising

In this section, we will introduce the core principles of Falcon. **First**, we present the way of leveraging previously generated action sequences  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$ , based on the model’s confidence in its prior predictions. **Second**, we describe the thresholding mechanism that determines which partial denoised action from past timesteps serves as the initialization for the current denoising process. **Lastly**, we outline the implementation details of Falcon.

#### 3.1. Reference Actions

Falcon first leverages the unexecuted action sequence  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$ , predicted from the previous observation  $\mathbf{O}_{t-T_a}$ , as a reference for denoising the current action sequence  $\mathbf{a}_{t:t+T_p}$  at time step  $t$ . This approach is motivated by our observation that the Euclidean distance  $\|\tilde{\mathbf{a}}_{t:t-T_a+T_p} - \mathbf{a}_{t:t-T_a+T_p}\|_2$  between  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$  and  $\mathbf{a}_{t:t-T_a+T_p}$  exhibits a high probability density near zero (see Fig. 2), indicating that  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$  closely approximates  $\mathbf{a}_{t:t-T_a+T_p}$  in most cases.

Furthermore, as shown in Eq. 4, diffusion policies take the latest  $T_o$  observations as input and train to output the future  $T_p$  action sequence  $\mathbf{A}_t = [\mathbf{a}_{t:t+T_a}, \tilde{\mathbf{a}}_{t+T_a:t+T_a+T_p}]$ . This training paradigm provides confidence that the predicted action sequence  $\tilde{\mathbf{a}}_{t:t-T_a+T_p} \in \mathbf{A}_t$  is sufficiently accurate for use as a reference. Additionally, Falcon incorporates the score vector field  $\nabla_{\mathbf{A}_t} \log p(\mathbf{A}_t | \mathbf{O}_t)$ , computed from the current observation  $\mathbf{O}_t$ , to guide the denoising process of the partial denoised action at the current time step, ensuring more precise and stable action refinement.

#### 3.2. Thresholding Mechanism

To determine which partial denoised action from historical observations should be used as the initialization for denoising at the current timestep, we employ Tweedie’s approach (Efron, 2011; Chung et al.; Kim & Ye, 2021). Given that the forward process of diffusion policy follows

$$\mathbf{A}_t^k = \sqrt{\bar{\alpha}_k} \mathbf{A}_t^0 + \sqrt{1-\bar{\alpha}_k} \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (5)$$

We derive the posterior expectation, which serves as the one-step estimation of the partial denoised actions  $\mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i}$ , denoted by  $\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i}$ :

$$\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i} = \mathbb{E}[\mathbf{a}_{\tau:\tau-T_a+T_p}^0 | \mathbf{O}_t, \mathbf{a}_{\tau:\tau-T_a+T_p}^k]. \quad (6)$$

This estimation is conditioned on the current observation  $\mathbf{O}_t$ , as formulated in Proposition 3.1. Here,  $\tau < t$  represents a historical decision step and  $k$  denotes the noise level of the partial denoised action.

The dependency between past and current actions is measured by the Euclidean  $\|\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i} - \tilde{\mathbf{a}}_{t:t-T_a+T_p}\|_2$  between  $\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i}$  and the reference action  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$ . If this distance falls below a predefined threshold  $\epsilon$ , the partial denoised action  $\mathbf{a}_{\tau:\tau-T_a+T_p}^k$  is selected, as it is likely to converge to the desired action  $\mathbf{a}_{t:t+T_p}$ .

**Proposition 3.1.** (Tweedie’s formula for denoising) Let  $\mathbf{x}$  be a random variable with a probability distribution  $p(\mathbf{x})$ , and Let  $\mathbf{x}_\sigma := \mathbf{x} + \sigma \mathbf{z}$ , where  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_D)$  and  $\sigma > 0$  is a known scalar. Then, the best estimate  $\hat{\mathbf{x}}_\sigma$  of  $\mathbf{x}$  in mean squared error, given the noisy observation  $\mathbf{x}_\sigma$ , is given by the formula:

$$\hat{\mathbf{x}}_\sigma := \mathbb{E}_{p(\mathbf{x}|\mathbf{x}_\sigma)}[\mathbf{x}] = \mathbb{E}[\mathbf{x} | \mathbf{x}_\sigma] = \mathbf{x}_\sigma + \sigma^2 \nabla_{\mathbf{x}_\sigma} \log p(\mathbf{x}_\sigma) \quad (7)$$

**Remark 3.2.** In the context of DDPM (Ho et al., 2020) whose diffusion forward process is  $\mathbf{x}_t \sim \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1-\bar{\alpha}_t) \mathbf{I}_D)$ , Tweedie’s formula can be rewritten as

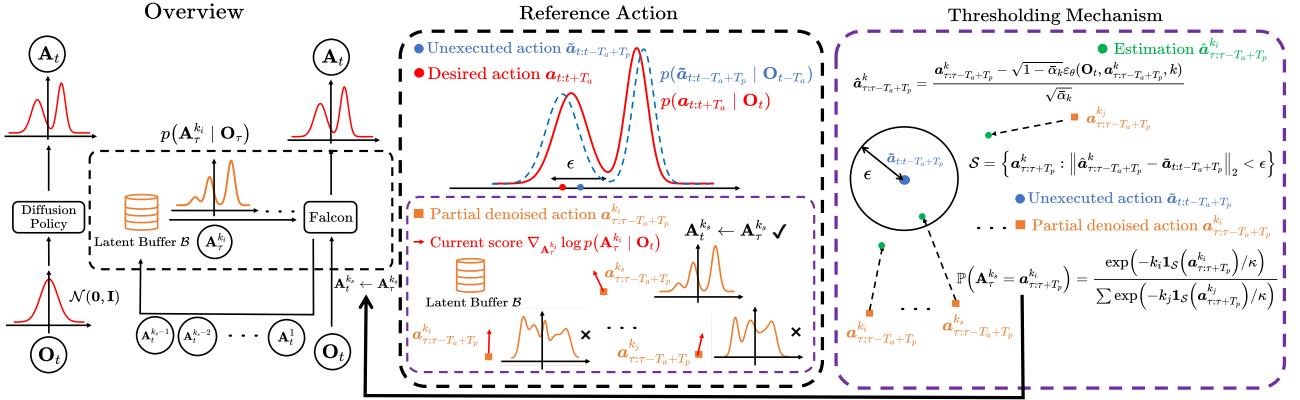
$$\mathbb{E}[\mathbf{x}_0 | \mathbf{x}_t] = (\mathbf{x}_t + (1-\bar{\alpha}_t) \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)) / \sqrt{\bar{\alpha}_t}. \quad (8)$$

Since  $\nabla_{\mathbf{A}_t^k} \log p(\mathbf{A}_t^k | \mathbf{O}_t) = -\frac{\varepsilon}{\sqrt{1-\bar{\alpha}_k}} \approx \frac{\varepsilon_\theta(\mathbf{O}_t, \mathbf{A}_t^k, k)}{\sqrt{1-\bar{\alpha}_k}}$  (Ho et al., 2020; Chi et al., 2023), we can obtain the posterior expectation by

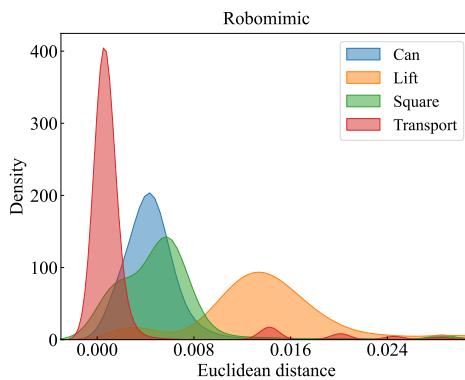
$$\begin{aligned} & \mathbb{E}[\mathbf{a}_{\tau:\tau-T_a+T_p}^0 | \mathbf{O}_t, \mathbf{a}_{\tau:\tau-T_a+T_p}^k] \\ &= \frac{\mathbf{a}_{\tau:\tau-T_a+T_p}^k - \sqrt{1-\bar{\alpha}_k} \varepsilon_\theta(\mathbf{O}_t, \mathbf{a}_{\tau:\tau-T_a+T_p}^k, k)}{\sqrt{\bar{\alpha}_k}} \end{aligned} \quad (9)$$

Finally, we define the set of candidate actions as  $\mathcal{S} = \{\mathbf{a}_{\tau:\tau-T_a+T_p}^k : \|\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^k - \tilde{\mathbf{a}}_{\tau:\tau-T_a+T_p}\|_2 < \epsilon, \forall \tau < t, k \in [K]\}$  and sample the starting point through the following distribution

$$\mathbb{P}\left(\mathbf{A}_t^{k_s} = \mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i}\right) = \frac{\exp\left(-k_i \mathbf{1}_{\mathcal{S}}\left(\mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i}\right) / \kappa\right)}{\sum_{\substack{\tau < t \\ k \in [K]}} \exp\left(-k_j \mathbf{1}_{\mathcal{S}}\left(\mathbf{a}_{\tau:\tau-T_a+T_p}^{k_j}\right) / \kappa\right)}, \quad (10)$$



**Figure 1. Method Description** a) Falcon begins denoising from historically generated partial denoised actions rather than a normal distribution, requiring less than  $k_s$  steps to produce the action sequence  $\mathbf{A}_t$ . The process involves 2 steps: setting reference actions and retrieving a partial denoised action sequence  $\mathbf{A}_\tau^k$  from the latent buffer  $\mathcal{B}$  to start denoising through a thresholding mechanism. b) Reference Action. Falcon uses unexecuted actions  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$  from the previous step  $t - T_a$  as the desired action  $\mathbf{a}_{t:t+T_a}$ , selecting a partial denoised action from  $\mathcal{B}$  as the starting point. c) Thresholding Mechanism. Falcon evaluates all actions in  $\mathcal{B}$  in parallel, identifying those close to the reference action after one-step estimation, and samples the starting point  $\mathbf{A}_t^{k_s}$  based on the noise level.



**Figure 2. Probability density estimation of  $\|\tilde{\mathbf{a}}_{t:t-T_a+T_p} - \mathbf{a}_{t:t-T_a+T_p}\|_2$ .**  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$  is nearly the same as  $\mathbf{a}_{t:t-T_a+T_p}$  since the majority of Euclidean distances between  $\tilde{\mathbf{a}}_{t:t-T_a+T_p}$  and  $\mathbf{a}_{t:t-T_a+T_p}$  are concentrated within the range of less than 0.015. Samples are collected by diffusion policy across four Robomimic environments (Can, Lift, Square, and Transport), where 200 trajectories were generated for each environment.

where  $\kappa$  is the temperature scaling factor and  $\mathbf{1}_S(\cdot)$  is the indicator function of set  $S$ . Since exclusively sampling from the latent buffer may result in idle actions, we introduce an exploration rate  $\delta$ , inspired by the  $\epsilon$ -greedy method from Reinforcement Learning (Sutton & Barto, 2018). With probability  $\delta$ , we sample from the standard Gaussian distribution instead, ensuring more diverse behavior.

### 3.3. Implementation Details

In Algorithm 1, we present the pseudocode of Falcon, including threshold  $\epsilon$ , exploration rate  $\delta$  and latent buffer  $\mathcal{B}$ .

In practice, we can't store all the partial denoising action generated from historical observations because of the limited GPU memory. Considering that earlier actions are generally less relevant to the current decision and that we aim to maximize efficiency, we designed a priority queue to implement the latent buffer. Partial denoising actions  $\mathbf{a}_{\tau:\tau+T_p}^k$  with earlier timesteps  $\tau$  and higher noise level  $k$  are given priority for removal from the queue. To prevent Falcon from repeating previous actions, we introduce  $k_{\min}$  and filter out partial denoised actions with  $k < k_{\min}$  during selection. This ensures that only actions with sufficient noise levels are used in the iterative sampling process.

At timestep  $t = 1$ , where no prior information is available, Falcon directly samples  $\mathbf{a}_{t:t+T_p}^K$  from a Gaussian distribution  $\mathcal{N}(\mathbf{0}, \mathbf{I})$ , and iteratively samples conditional on  $\mathbf{O}_t$  by Eq. 11 like DDPM (Ho et al., 2020) Line 7. During this process, Falcon iteratively performs denoising and stores resulting partial denoised actions  $\mathbf{a}_{t:t+T_p}^k$  in a latent buffer  $\mathcal{B}$  on Line 5. This latent buffer records all partial denoised actions generated during the sampling process.

$$\begin{aligned} \mathbf{a}_{t:t+T_p}^{k-1} &= \frac{1}{\sqrt{\alpha_k}} \left( \mathbf{a}_{t:t+T_p}^k - C \right) + \sigma_k \mathbf{z}, \\ C &= \frac{1 - \alpha_k}{\sqrt{1 - \bar{\alpha}_k}} \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^k, k), \\ \mathbf{z} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned} \quad (11)$$

For timestep  $t > 1$ , Falcon utilizes one-step estimation according to Eq. 9 on all stored samples  $\mathbf{a}_{\tau:\tau+T_p}^{k_i}$  in the latent buffer  $\mathcal{B}$  conditioned on the current observation  $\mathbf{O}_t$ , which is the most compute-intensive part of the algorithm but can be efficiently parallelized. Line 13 obtains the partial



	Lift		Can		Square		Transport		Tool hang	
	ph	Lift mh	ph	Can mh	ph	Square mh	ph	Transport mh	ph	ToolHang ph
DDPM	1.00 $\pm$ 0.00	0.95 $\pm$ 0.07	0.98 $\pm$ 0.13	0.97 $\pm$ 0.15	0.91 $\pm$ 0.07	0.85 $\pm$ 0.35	0.80 $\pm$ 0.39	0.65 $\pm$ 0.47	0.52 $\pm$ 0.49	
DDPM+Falcon	1.00 $\pm$ 0.00	0.97 $\pm$ 0.18	0.97 $\pm$ 0.17	0.97 $\pm$ 0.17	0.95 $\pm$ 0.23	0.82 $\pm$ 0.38	0.85 $\pm$ 0.36	0.66 $\pm$ 0.48	0.55 $\pm$ 0.50	
SDP(DDPM)	1.00 $\pm$ 0.00	0.98 $\pm$ 0.12	0.97 $\pm$ 0.17	0.94 $\pm$ 0.24	0.88 $\pm$ 0.32	0.80 $\pm$ 0.40	0.81 $\pm$ 0.39	0.46 $\pm$ 0.49	0.58 $\pm$ 0.50	
DDIM	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	0.99 $\pm$ 0.07	0.97 $\pm$ 0.15	0.92 $\pm$ 0.26	0.87 $\pm$ 0.33	0.79 $\pm$ 0.40	0.63 $\pm$ 0.48	0.55 $\pm$ 0.50	
DDIM+Falcon	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 1.00	0.98 $\pm$ 0.14	0.91 $\pm$ 0.28	0.85 $\pm$ 0.36	0.81 $\pm$ 0.39	0.63 $\pm$ 0.48	0.54 $\pm$ 0.50	
DPMSolver	0.98 $\pm$ 0.12	0.95 $\pm$ 0.20	0.97 $\pm$ 0.21	0.97 $\pm$ 0.17	0.93 $\pm$ 0.25	0.84 $\pm$ 0.36	0.70 $\pm$ 0.45	0.55 $\pm$ 0.49	0.58 $\pm$ 0.49	
DPMSolver+Falcon	0.98 $\pm$ 0.14	0.96 $\pm$ 0.20	0.96 $\pm$ 0.20	0.98 $\pm$ 0.14	0.94 $\pm$ 0.24	0.90 $\pm$ 0.30	0.74 $\pm$ 0.43	0.54 $\pm$ 0.50	0.56 $\pm$ 0.48	

**Table 1. Success Rate in Robomimic.** We present the success rate with 200 evaluation episodes in the format of (mean of success rate)  $\pm$  (standard deviation of success rate). Our results show that Falcon matches the original methods.

denoising action sequence, and Line 14 sets it as the starting point of the rest of the iterative sampling process.

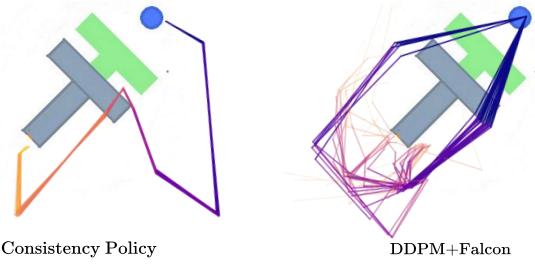
Falcon can also be compatible with other diffusion policy acceleration techniques by replacing the solver in Line 7 and Line 18 with other SDE/ODE solvers like DDIM (Song et al., a) and DPMSolver (Lu et al., 2022). The detailed pseudocode is in the Appendix B.

### Algorithm 1 Falcon

```

Require: Diffusion model  $\epsilon_\theta$  with noise scheduler  $\bar{\alpha}_k$ , variance  $\sigma_k^2$ , threshold  $\epsilon$ , exploration probability  $\delta$ , latest  $T_o$  observations  $\mathbf{O}_t$ , latent buffer  $\mathcal{B}$ , temperature scaling factor  $\kappa$ .
1: for  $t = 1, \dots, T$  do
2:   if  $t = 1$  then
3:      $\mathbf{a}_{t:t+T_p}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
4:     for  $k = K, \dots, 1$  do
5:        $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{a}_{t:t+T_p}^k\}$ .
6:        $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $k > 1$ , else  $\mathbf{z} \leftarrow \mathbf{0}$ 
7:       Sample  $\mathbf{a}_{t:t+T_p}^{k-1}$  according to Eq. 11
8:     end for
9:   end if
10:  if  $t > 1$  then
11:    Compute one-step estimation  $\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i}$  via Eq. 9.
12:     $\mathcal{S} \leftarrow \{\mathbf{a}_{\tau:\tau+T_p}^k : \|\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i} - \hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^k\|_2 < \epsilon\}$ 
13:    Sample  $\mathbf{a}_{\tau:\tau+T_p}^{k_s}$  according to Eq. 10
14:     $\mathbf{a}_{t:t+T_p}^{k_s} \leftarrow \mathbf{a}_{\tau:\tau+T_p}^{k_s}$ 
15:    for  $k = k_s, \dots, 1$  do
16:       $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{a}_{t:t+T_p}^k\}$ .
17:       $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $k > 1$ , else  $\mathbf{z} \leftarrow \mathbf{0}$ 
18:      Sample  $\mathbf{a}_{t:t+T_p}^{k-1}$  according to Eq. 11
19:    end for
20:  end if
21: end for

```



**Figure 3. Multimodal behavior.** At the given symmetric configuration in the PushT task, DDPM+Falcon (right) preserves multimodal behaviors by producing diverse reaching trajectories. In contrast, the Consistency Policy (left) shows bias toward one mode. Each line represents one rollout trajectory, plotted from 20 trajectories sampled at the same state.

## 4. Experiments

Our experiments aim to address four key questions: (1) Can Falcon accelerate diffusion policy, and does it further enhance speed when integrated with other acceleration algorithms (Section 4.3)? (2) Can Falcon effectively accelerate diffusion policies in real-world robotic settings while preserving performance (Section 4.5)? (3) Does Falcon maintain its acceleration advantage in long-sequence tasks (Section E.3)? (4) Can Falcon retain the ability to express multimodality while achieving speed improvements (Section 4.6)?

### 4.1. Metrics

Our experiments are primarily evaluated by two metrics. The first is the success rate, which measures the mean and

	Lift		Can		Square		Transport		ToolHang
	ph	mh	ph	mh	ph	mh	ph	mh	ph
DDPM	100.0 $\pm$ 0.0								
DDPM+Falcon	<b>12.9<math>\pm</math>0.3</b>	<b>24.6<math>\pm</math>1.6</b>	<b>35.2<math>\pm</math>1.3</b>	<b>20.3<math>\pm</math>0.3</b>	<b>37.1<math>\pm</math>1.5</b>	<b>20.5<math>\pm</math>0.6</b>	<b>46.9<math>\pm</math>3.7</b>	<b>48.0<math>\pm</math>4.7</b>	<b>33.6<math>\pm</math>3.2</b>
SDP(DDPM)	50.0 $\pm$ 0.0								
DDIM	16.0 $\pm$ 0.0								
DDIM+Falcon	<b>7.5<math>\pm</math>0.2</b>	<b>7.7<math>\pm</math>0.2</b>	<b>6.5<math>\pm</math>0.3</b>	<b>7.6<math>\pm</math>2.1</b>	<b>7.6<math>\pm</math>1.0</b>	<b>7.2<math>\pm</math>1.2</b>	<b>10.5<math>\pm</math>2.0</b>	<b>9.0<math>\pm</math>1.1</b>	<b>10.1<math>\pm</math>1.4</b>
DPM Solver	16.0 $\pm$ 0.0								
DPM Solver+Falcon	<b>6.4<math>\pm</math>0.1</b>	<b>10.4<math>\pm</math>2.0</b>	<b>14.7<math>\pm</math>0.8</b>	<b>10.9<math>\pm</math>1.8</b>	<b>14.4<math>\pm</math>0.5</b>	<b>7.8<math>\pm</math>0.9</b>	<b>11.0<math>\pm</math>1.2</b>	<b>12.8<math>\pm</math>1.4</b>	<b>12.1<math>\pm</math>1.3</b>

Table 2. NFE in Robomimic. We present the NFE with 200 evaluation episodes in the format of (mean of NFE)  $\pm$  (standard deviation of NFE). Our results show that Falcon drastically reduces the denoising steps compared with the original methods.

standard deviation of task completion across all trials. The second metric is generation time, quantified in terms of the Number of Function Evaluations (NFE) (Prasad et al., 2024). Since the inference cost for these models is primarily determined by NFE, and given that the network architectures are kept constant across experiments, NFE serves as a reliable indicator of relative performance. This metric effectively captures the inference cost, unbiased by GPU imbalances, and allows for a fair comparison across methods.

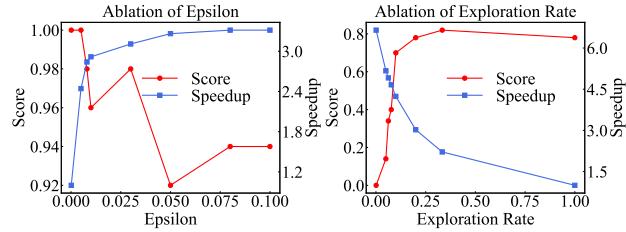


Figure 4. **Left:** Ablation analysis on threshold  $\epsilon$ . The effect of varying the  $\epsilon$  threshold on both the score (red circles) and speedup (blue squares), demonstrating the trade-off between performance and speed as  $\epsilon$  changes. **Right:** Ablation analysis on exploration rate  $\delta$ . The score (red circles) increases sharply with  $\delta$  and then stabilizes, while the speedup (blue squares) decreases as the exploration rate rises. This study highlights the balance between exploration and exploitation.

## 4.2. Simulation and Real-World Environments

We evaluate Falcon across diverse simulated and real-world environments to demonstrate its generality and effectiveness. Simulation benchmarks include RoboMimic (Mandlekar et al., 2022), Franka Kitchen (Gupta et al., 2020), Block-Push (Shafiullah et al., 2022), PushT (Chi et al., 2023), MetaWorld (Yu et al., 2020), and ManiSkill2 (Gu et al., 2023), covering short- and long-horizon tasks, as well as multimodal behavior tasks. For real-world evaluation, we deploy Falcon on a dual-arm robot platform in two manipulation tasks: dexterous grasping and high-precision insertion. Details of simulation settings, network architectures, and hardware platforms are provided in the Appendix A.

## 4.3. Falcon is a Plug & Play Acceleration Algorithm

In the first experiment, we examine our algorithm on the Robomimic benchmarks. We present the success rates, NFE, and speedup for both the original diffusion models, their Falcon-enhanced counterparts, and SDP in Table 1, Table 2, and Table 4, respectively. To better understand Falcon’s impact, we categorize the tasks into simpler and more complex ones based on the level of fine-grained manipulation required. In the Robomimic benchmark, Lift and Can are relatively simple tasks, while the remaining tasks, Square, Transport, and ToolHang, are more complex.

**Simple tasks.** The Lift and Can tasks involve smooth and predictable movements. In Lift, the robotic arm moves downward to grasp a small cube and then upward to lift it, with minimal variation in action within each phase. Can follows a similar pattern, where the robot picks up a coke can from a bin and places it in a target bin, primarily involving steady translational motion. Both tasks exhibit strong dependencies between consecutive actions, which makes them ideal for Falcon acceleration. As shown in Fig. 13 (left), Falcon starts denoising with low noise, significantly speeding up the process. Table 4 shows Falcon achieving a **7x** speedup in Lift and **2-3x** in Can, demonstrating its efficiency in structured motion tasks.

**Difficult tasks.** Square, Transport, and ToolHang are more complex and involve finer, more precise movements. For example, in the Transport task, two robot arms must work together to transfer a hammer from a closed container on a shelf into a target bin on another shelf. One arm retrieves the hammer from the container, while the other arm first clears the target bin by removing trash. Then, one arm hands over the hammer to the other, which must place it in the target bin. These actions are dynamic and entail larger changes between consecutive movements. As a result, Falcon can’t start denoising actions with a low noise level (see Fig. 13 (right)). Consequently, Falcon’s speedup is not as pronounced as in simpler tasks. However, even in these more challenging scenarios, as shown in Table 4, Falcon still

	Lift		Can		Square		Transport		ToolHang
	ph	mh	ph	mh	ph	mh	ph	mh	ph
DPMsolver*	1.00 $\pm$ 0.00	0.95 $\pm$ 0.20	0.94 $\pm$ 0.23	0.96 $\pm$ 0.20	0.84 $\pm$ 0.37 ( $\downarrow$ 10.6%)	0.58 $\pm$ 0.50 ( $\downarrow$ 35.6%)	0.69 $\pm$ 0.46 ( $\downarrow$ 6.8%)	0.54 $\pm$ 0.50	0.53 $\pm$ 0.50
DPMsolver+Falcon	0.98 $\pm$ 0.14	0.96 $\pm$ 0.20	0.96 $\pm$ 0.20	0.98 $\pm$ 0.14	0.94 $\pm$ 0.24	0.90 $\pm$ 0.30	0.74 $\pm$ 0.43	0.54 $\pm$ 0.50	0.56 $\pm$ 0.48

**Table 3. Reduced-step DPMsolver.** We report the success rate across 200 evaluation rollouts in the format of (mean  $\pm$  std). DPMsolver\* uses the same number of sampling steps as NFE of DPMsolver(16 steps)+Falcon. Entries marked with  $\downarrow$  indicate performance drops compared to DPMsolver(16 steps)+Falcon. The percentage drop is computed as (Falcon score – DPMsolver\* score)/Falcon score  $\times$  100%.

	Lift		Can		Square		Transport		ToolHang
	ph	mh	ph	mh	ph	mh	ph	mh	ph
DDPM+Falcon	7.78 $\pm$ 0.19	4.07 $\pm$ 0.23	2.84 $\pm$ 0.01	4.93 $\pm$ 0.06	2.70 $\pm$ 0.10	4.88 $\pm$ 0.13	2.14 $\pm$ 0.13	2.10 $\pm$ 0.16	2.86 $\pm$ 0.26
SDP(DDPM)	2.00 $\pm$ 0.00								
DDIM+Falcon	2.13 $\pm$ 0.03	2.08 $\pm$ 0.06	2.45 $\pm$ 0.11	2.15 $\pm$ 0.24	2.15 $\pm$ 0.22	2.27 $\pm$ 0.32	1.57 $\pm$ 0.24	1.88 $\pm$ 0.18	1.61 $\pm$ 0.20
DPMsolver+Falcon	2.48 $\pm$ 0.02	1.60 $\pm$ 0.32	1.09 $\pm$ 0.07	1.51 $\pm$ 0.25	1.11 $\pm$ 0.04	2.09 $\pm$ 0.21	1.48 $\pm$ 0.17	1.27 $\pm$ 0.13	1.32 $\pm$ 0.12

**Table 4. Speedup in Robomimic.** We present the speed with 200 evaluation episodes in the format of (mean of speedup)  $\pm$  (standard deviation of speedup). Speed for X+Falcon is calculated by NFE of X / NFE of X+Falcon.

achieves around **2x** speedup while maintaining performance close to that of the original models.

We further evaluate Falcon in MetaWorld environments using 3D Diffusion Policy (Ze et al., 2024b). The architecture follows the same setup as 3D Diffusion Policy, with a DDIM scheduler using 10-step discretization. As shown in the Appendix C, Falcon achieves a **3-4x** acceleration on top of DDIM, further demonstrating its effectiveness in accelerating existing diffusion-based policies across different tasks.

#### 4.4. Falcon mitigates performance degradation in reduced-step solvers

To further demonstrate Falcon’s compatibility with existing acceleration techniques, we compare Falcon-enhanced DPMsolver with a reduced-step DPMsolver baseline (denoted as DPMsolver\*), where both use the same number of denoising steps. As shown in Table 3, DPMsolver\* exhibits significant performance drops in several tasks when the number of steps is aggressively reduced (e.g.,  $-10.6\%$  in Square.ph,  $-35.6\%$  in Square.mh), while Falcon maintains high success rates with the same number of steps. These results highlight Falcon’s ability to recover performance in low-step regimes and demonstrate its practical value as a plug-and-play solution to boost efficiency without sacrificing accuracy.

#### 4.5. Real-World Evaluation of Falcon

To validate the practicality of Falcon beyond simulation, we evaluate it in two real-world robotic manipulation tasks: Dexterous Grasping (Fig. 5) and Square Stick Insertion (Fig. 6).

**Dexterous Grasping.** The first task involves dexterous

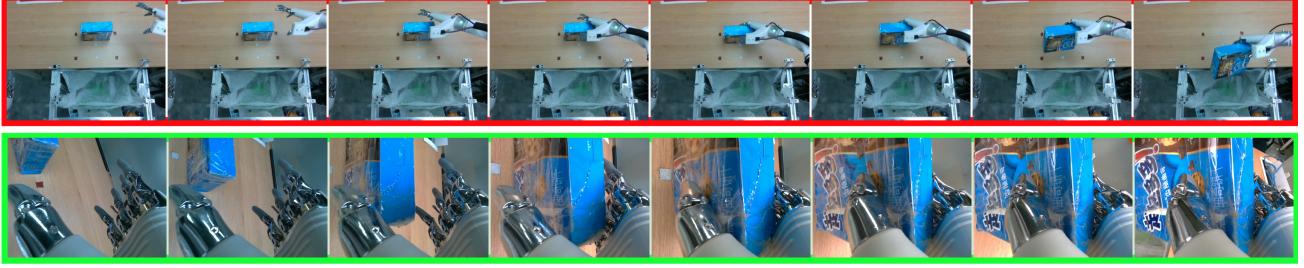
grasping to pick up the blue object placed on a table. Falcon is deployed on top of DDPM without any additional retraining. We compare Falcon+DDPM against SDP(DDPM) and DDPM alone. As shown in Table 5, Falcon achieves a **3.07x** speedup over DDPM and outperforms SDP in both runtime (0.145s vs. 0.233s) and CPU memory usage (3730MB vs. 3808MB).

**High-Precision Insertion.** The second task involves inserting a square stick into a tall cylindrical chip can. This task requires precise 3D alignment; even slight deviations in height or angle result in failure. Falcon is again applied as a plug-in to DDPM. We train both DDPM and SDP using 50 human demonstrations. As shown in Table 6, Falcon matches DDPM in 90% success rate while being **2.86x** faster and more memory efficient.

These results confirm that Falcon not only accelerates diffusion policy inference in simulation but also effectively transfers to real-world robotic applications.

#### 4.6. Falcon retains the ability to express multimodality policy

We further evaluate Falcon in the PushT task under a symmetric configuration, where multimodality can be visualized directly through trajectory diversity. As shown in Fig. 3, DDPM+Falcon produces diverse reaching trajectories, while Consistency Policy exhibits mode collapse, predominantly favoring a single direction. This qualitative difference demonstrates Falcon’s ability to maintain diverse outputs even in symmetric settings, without retraining.



**Figure 5. Real-world Dexterous Grasping with DDPM+Falcon.** Top row (red): head camera view (RealSense D435). Bottom row (green): wrist camera view (RealSense D405C). The sequence shows the successful execution of grasping using Falcon-accelerated policy.

Method	Speedup	NFE	Run Time (s)	GPU MEMS (MB)	CPU MEMS (MB)	Success rate
DDPM	1.0x	$100.00 \pm 0.00$	$0.43 \pm 0.01$	$3730.17 \pm 0.56$	$3737.06 \pm 2.63$	100%
SDP(DDPM)	1.86x	$52.16 \pm 2.18$	$0.23 \pm 0.01$	$3733.11 \pm 2.68$	$3808.67 \pm 7.68$	100%
Falcon+DDPM	<b>3.07x</b>	$30.98 \pm 1.33$	$0.14 \pm 0.01$	$3730.19 \pm 0.56$	$3746.67 \pm 3.03$	100%

**Table 5. Dexterous Grasping in Real-world Environment.** Each entry is evaluated with 20 rollouts in the mean  $\pm$  standard deviation format. DDPM and SDP(DDPM) are trained with 1000 epochs with 37 collected expert demonstrations. Our results show that Falcon outperforms SDP in runtime reduction with lower memory usage.

#### 4.7. Ablation analysis: sensitivity to the threshold $\epsilon$

We explore how the threshold  $\epsilon$  influences Falcon’s performance in the Square\_ph task. The threshold  $\epsilon$  determines the confidence in reusing partial denoised actions toward the desired action  $a_{t:t+T_p}$  guided by  $\nabla_{A_t^k} \log p(A_t^k | O_t)$ . We conducted 50 evaluations using a range of  $\epsilon$  values:  $\{10^{-4}, 5 \times 10^{-3}, 8 \times 10^{-3}, 10^{-2}, 3 \times 10^{-2}, 5 \times 10^{-2}, 8 \times 10^{-2}, 10^{-1}\}$ . As shown in Fig. 4 (left), when  $\epsilon$  is small, Falcon is highly selective about using partial denoised actions, leading to slow denoising because Falcon frequently samples from the standard normal distribution. This results in no significant improvement. When  $\epsilon$  is large, Falcon becomes overly confident in the partial denoised actions, leading to selecting many incorrect actions that do not align with  $a_{t+T_a:t+T_p}$ , which causes a sharp drop in Score. The optimal range of  $\epsilon$  lies in the middle, where Falcon can achieve speedup without sacrificing task performance, demonstrating that a balanced  $\epsilon$  allows Falcon to maintain both efficiency and accuracy.

#### 4.8. Ablation analysis: sensitivity to exploration rate $\delta$

We assess the impact of the exploration rate  $\delta$  on Falcon by running 50 evaluations on the Transport\_ph task, using  $\delta \in \{0.001, 0.05, 0.0625, 0.076, 0.1, 0.2, 0.33, 1\}$ . Exploration Rate  $\delta$  controls the probability that Falcon starts denoising from  $\mathcal{N}(0, I)$  rather than the partial denoised action. This exploration mechanism allows Falcon to fill the latent buffer with more partial denoised actions, increasing exploration and preventing over-reliance on the reference action. As shown in Fig. 4 (right), when  $\delta$  is set to 0, Falcon primarily aligns with the reference action at each step. Since the

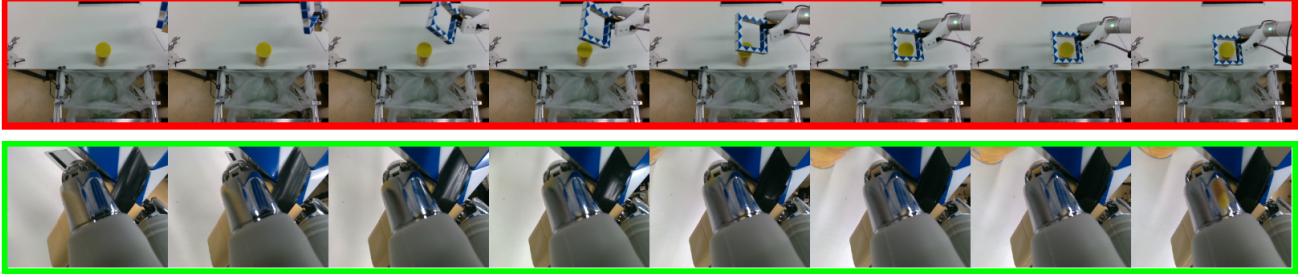
reference action is not perfectly aligned with the desired action  $A_t$ , errors accumulate over time, causing Falcon to generate incorrect actions, resulting in a significant drop in score. As  $\delta$  increases, Falcon samples more diverse actions, improving score by reducing the accumulation of errors. However, further increasing  $\delta$  results in a decrease in speed, as the model becomes less reliant on the history of denoised actions and starts exploring more random samples from the normal distribution.

#### 4.9. Ablation analysis: sensitivity to selection mechanism

We evaluate Falcon’s partial denoised action selection mechanism. Falcon adaptively selects partial denoised actions based on  $\epsilon$  and the one-step estimation, estimating whether each action can be denoised to  $A_t$ . We compare this with the case where Falcon is fixed to always sample from actions with noise levels of  $K/2$  or  $K/5$  (represented by the red and green bars), without considering the adaptive selection mechanism. Fig. 7 (left) shows a significant decrease in Score when Falcon is restricted to these fixed actions, demonstrating that not leveraging the selection mechanism compromises performance. This confirms that Falcon’s Selection Mechanism is crucial for ensuring accurate action choices, allowing the model to maintain high performance while accelerating the denoising process.

## 5. Related Work

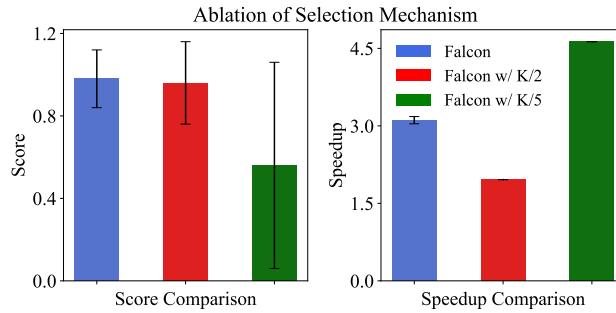
Diffusion policies are widely used for modeling complex behaviors in robotics. 3D Diffusion Policy (Ze et al., 2024b;



**Figure 6. Real-world Square Stick Insertion with DDPM+Falcon.** Top row (red): head camera view (RealSense D435). Bottom row (green): wrist camera view (RealSense D405C). The sequence shows the successful execution of insertion using Falcon-accelerated policy.

Method	Speedup	NFE	Run Time (s)	GPU MEMS (MB)	Success rate
DDPM	1.0x	$100.00 \pm 0.00$	$0.43 \pm 0.01$	$3735.76 \pm 0.48$	90%
SDP(DDPM)	1.95x	$50.00 \pm 0.00$	$0.22 \pm 0.01$	$3743.28 \pm 1.26$	85%
Falcon+DDPM	<b>2.86x</b>	$25.57 \pm 7.10$	$0.15 \pm 0.12$	$3731.50 \pm 5.08$	90%

**Table 6. Square Stick Insertion in Real-world Environment.** Each entry is evaluated with 20 rollouts in the mean  $\pm$  standard deviation format. Falcon is set with  $T_p = 32$ ,  $T_a = 16$ ,  $T_o = 1$ ,  $\epsilon = 0.02$ ,  $|\mathcal{B}| = 20$ .



**Figure 7. Ablation analysis on selection mechanism.** This figure compares Falcon’s performance with different partial denoised action selection configurations. Restricting Falcon to choose actions with fixed noise levels (K/2, K/5) significantly reduces score, highlighting that the selection mechanism ensures accurate action choices while accelerating denoising and maintaining high performance.

Ke et al.) improves visuomotor performance by integrating 3D visual data from sparse point clouds. Large-scale models like Octo (Team et al., 2024) and RDT (Liu et al., 2024) scale diffusion policy parameters to build vision-language-action foundation models. However, their high computational cost limits real-time applications, necessitating acceleration techniques.

To speed up inference, ODE-based sampling methods like DDIM (Song et al., a) and DPMSolver (Lu et al., 2022) reduce denoising steps, while ParaDiGMS (Shih et al., 2024) leverages parallel computing to accelerate sampling on GPUs. Distillation-based methods further improve efficiency by training models for single-step inference. CP (Prasad et al., 2024) builds on the Consistency Trajectory

Model (Kim et al.), enabling a pre-trained diffusion policy to generate actions within a few steps. ManiCM (Lu et al., 2024) extends this approach to 3D robotic tasks. OneDP (Wang et al., 2024) and SDM (Jia et al., 2024) introduce score-based distillation to reduce performance degradation. However, these methods require task-specific training, limiting adaptability. Streaming Diffusion Policy (SDP) (Høeg et al., 2024) uses partial denoised action trajectories, similar to Falcon. However, SDP requires task-specific training and lacks compatibility with general acceleration techniques.

In contrast, Falcon is a training-free acceleration framework that leverages sequential dependencies to improve sampling speed while preserving multimodal expressiveness. It integrates seamlessly with DDIM and DPMSolver, making it a flexible, plug-and-play solution across various robotic tasks.

## 6. Conclusion

This paper introduces Falcon, the first diffusion policy approach to accelerate action generation in complex visuomotor tasks by leveraging inter-step dependencies in decision-making. Empirical results confirm that Falcon outperforms strong baselines, such as DDIM and DPMSolver, in terms of speed without sacrificing accuracy or expressiveness. Overall, Falcon offers a simple yet effective solution for real-time robotic tasks, providing efficient action generation for complex visuomotor environments. However, one limitation of Falcon is that it does not use a single set of parameters to accelerate all tasks, since different environments may require task-specific parameter tuning. Future work will explore adaptive parameter selection strategies to enhance its generalizability across diverse robotic applications further.

## Impact Statement

Falcon removes the efficiency bottleneck of diffusion-based visuomotor policies, making real-time action generation feasible for robotic manipulation, autonomous navigation, and interactive AI. By enabling fast, high-quality decision-making in long-horizon tasks, Falcon expands the practical use of diffusion policies in real-world robot applications.

## Acknowledgements

This work is sponsored by National Natural Science Foundation of China (62376013). We sincerely thank Tianjie He for his invaluable support in setting up the hardware platform used in our real-world experiments.

## References

- Chen, H., Lu, C., Wang, Z., Su, H., and Zhu, J. Score regularized policy optimization through diffusion behavior. In *The Twelfth International Conference on Learning Representations*, a.
- Chen, S.-F., Wang, H.-C., Hsu, M.-H., Lai, C.-M., and Sun, S.-H. Diffusion model-augmented behavioral cloning. In *Forty-first International Conference on Machine Learning*, b.
- Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, pp. 02783649241273668, 2023.
- Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*.
- Efron, B. Tweedie’s formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.
- Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. In *Conference on Robot Learning*, pp. 158–168. PMLR, 2022.
- Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., and Su, H. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.
- Gupta, A., Kumar, V., Lynch, C., Levine, S., and Hausman, K. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*, pp. 1025–1037. PMLR, 2020.
- Hansen-Estruch, P., Kostrikov, I., Janner, M., Kuba, J. G., and Levine, S. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.
- Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- Høeg, S. H., Du, Y., and Egeland, O. Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models. *arXiv preprint arXiv:2406.04806*, 2024.
- Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.
- Jia, B., Ding, P., Cui, C., Sun, M., Qian, P., Fan, Z., and Wang, D. Score and distribution matching policy: Advanced accelerated visuomotor policies via matched distillation. *arXiv preprint arXiv:2412.09265*, 2024.
- Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.
- Ke, T.-W., Gkanatsios, N., and Fragkiadaki, K. 3d diffuser actor: Policy diffusion with 3d scene representations. In *8th Annual Conference on Robot Learning*.
- Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*.
- Kim, K. and Ye, J. C. Noise2score: tweedie’s approach to self-supervised image denoising without clean images. *Advances in Neural Information Processing Systems*, 34: 864–874, 2021.
- Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., and Zhu, J. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.
- Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.
- Lu, G., Gao, Z., Chen, T., Dai, W., Wang, Z., and Tang, Y. Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. *arXiv preprint arXiv:2406.01586*, 2024.

- Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., and Martín-Martín, R. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, pp. 1678–1690. PMLR, 2022.
- Prasad, A., Lin, K., Wu, J., Zhou, L., and Bohg, J. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.
- Ravan, Y., Yang, Z., Chen, T., Lozano-Pérez, T., and Kaelbling, L. P. Combining planning and diffusion for mobility with unknown dynamics. *arXiv preprint arXiv:2410.06911*, 2024.
- Reuss, M., Li, M., Jia, X., and Lioutikov, R. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.
- Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*.
- Shafiuallah, N. M., Cui, Z., Altanzaya, A. A., and Pinto, L. Behavior transformers: Cloning  $k$  modes with one stone. *Advances in neural information processing systems*, 35: 22955–22968, 2022.
- Shih, A., Belkhale, S., Ermon, S., Sadigh, D., and Anari, N. Parallel sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.
- Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, a.
- Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, b.
- Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.
- Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.
- Team, O. M., Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.
- Wang, G., Cai, Y., Peng, W., Su, S.-Z., et al. Pfddiff: Training-free acceleration of diffusion models combining past and future scores. In *The Thirteenth International Conference on Learning Representations*, a.
- Wang, Z., Hunt, J. J., and Zhou, M. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, b.
- Wang, Z., Li, Z., Mandlekar, A., Xu, Z., Fan, J., Narang, Y., Fan, L., Zhu, Y., Balaji, Y., Zhou, M., et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.
- Yang, J., Cao, Z., Deng, C., Antonova, R., Song, S., and Bohg, J. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. In *8th Annual Conference on Robot Learning*.
- Yang, L., Huang, Z., Lei, F., Zhong, Y., Yang, Y., Fang, C., Wen, S., Zhou, B., and Lin, Z. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.
- Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.
- Ze, Y., Chen, Z., Wang, W., Chen, T., He, X., Yuan, Y., Peng, X. B., and Wu, J. Generalizable humanoid manipulation with improved 3d diffusion policies. *arXiv preprint arXiv:2410.10803*, 2024a.
- Ze, Y., Zhang, G., Zhang, K., Hu, C., Wang, M., and Xu, H. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024b.
- Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*.
- Zhao, W., Bai, L., Rao, Y., Zhou, J., and Lu, J. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

## A. Experimental Setup

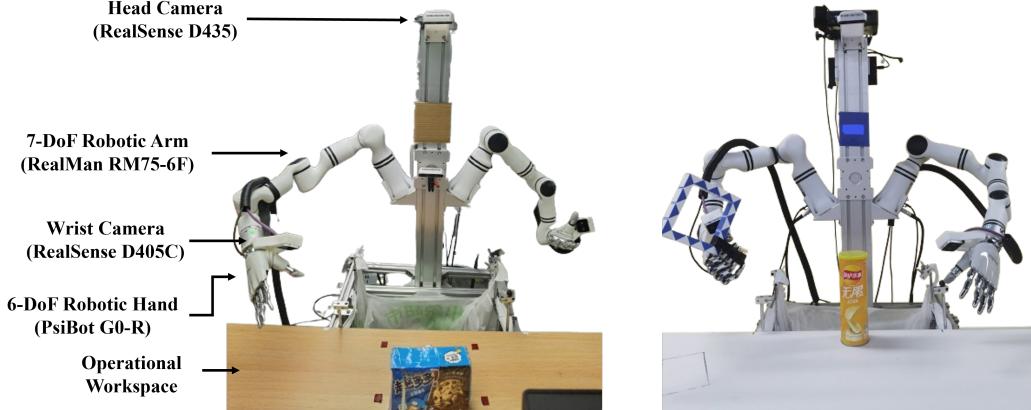


Figure 8. The hardware platform. Left: Task for Dexterous Grasping. Right: Task for inserting a stick into a chip can

### A.1. Simulation Environments

**Robomimic** (Mandlekar et al., 2022) is a large-scale benchmark designed to evaluate robotic manipulation algorithms using human demonstration datasets. Each task provides two types of human demonstrations: proficient human (PH) demonstrations and mixed proficient and nonproficient human (MH) demonstrations. Each environment uses an action prediction horizon  $T_p = 16$  and an action execution horizon  $T_a = 8$ , and all tasks use state-based observations. We construct a CNN-based Diffusion Policy (Chi et al., 2023) with DDPM scheduler using 100 denoising steps and the DDIM/DPMsolver scheduler using 16 denoising steps. Models are taken from the Diffusion Policy repository: [https://diffusion-policy.cs.columbia.edu/data/experiments/low\\_dim/](https://diffusion-policy.cs.columbia.edu/data/experiments/low_dim/).

**Franka Kitchen** (Gupta et al., 2020) is designed for evaluating algorithms on long-horizon, multi-stage robotics tasks. The task involves action sequences of dimension 112 and an episode length of 1200, with an action prediction horizon of  $T_p = 16$  and an action execution horizon of  $T_a = 8$ . We construct a transformer-based Diffusion Policy using a DDPM scheduler with 100-step discretization.

**Push-T** is adapted from IBC (Florence et al., 2022) and involves pushing a block in the shape of a T to a fixed target using a circular end effector. The task features randomized initial poses of the block and end-effector, requiring contact-rich, precise motion. We use state-based observations.

**Multimodel Block Pushing** (Shafiullah et al., 2022) tests the policy’s ability to model multimodal action distributions by pushing two blocks into two squares in any order. The demonstration data is generated by a scripted oracle with access to groundtruth state info. This oracle randomly selects an initial block to push and moves it to a randomly selected square. The remaining block is then pushed into the remaining square.

**MetaWorld** (Yu et al., 2020) is an open-source simulation benchmark for meta-reinforcement learning and multi-task learning. It consists of 50 different robotic manipulation tasks categorized into different difficulty levels ranging from simple to very challenging. We follow the setup and model training in the official 3D Diffusion Policy codebase (Ze et al., 2024b): <https://github.com/YanjieZe/3D-Diffusion-Policy>.

**ManiSkill2** (Gu et al., 2023) is a benchmark for robotic manipulation with 20 tasks, 2000+ objects, and 4M+ demonstrations. It supports rigid/soft-body tasks and fast visual learning (2000 FPS).

### A.2. Real-world Experiments

We conduct two real-world manipulation tasks—dexterous grasping and high-precision insertion. As shown in Figure 8, our robot consists of a 7-DoF RealMan RM75-6F arm and a 6-DoF PsiBot hand. It is equipped with a wrist-mounted RealSense D405C camera and a head-mounted RealSense D435 camera. Objects are placed on a table in front of the robot for manipulation. For each real-world task, we collect 50 human demonstrations and use the open-source Diffusion Policy implementation [https://github.com/real-stanford/diffusion\\_policy](https://github.com/real-stanford/diffusion_policy) and SDP implementations [https://github.com/Streaming-Diffusion-Policy/streaming\\_diffusion\\_policy](https://github.com/Streaming-Diffusion-Policy/streaming_diffusion_policy).

## B. Pseudocode of Falcon with other sampling solvers

In this section, we provide the pseudocode for Falcon when integrated with alternative sampling solvers, such as DDIM (Song et al., a) and DPMsolver (Lu et al., 2022).

### B.1. Falcon with DDIM

DDIM's sampling process follows Eq. 12, where  $\mathbf{x}_k$  represents the sample at noise level  $k$ ,  $\alpha_k$  is the noise scheduler and  $\epsilon_\theta$  is the noise prediction network. To integrate Falcon with DDIM, we replace  $\mathbf{x}_k$  with partial denoised action sequence  $\mathbf{a}_{t:t+T_p}^k$  at time step  $t$  with noise level  $k$  and substitute  $\epsilon_\theta(\mathbf{x}_k, k)$  with  $\epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^k, k)$  where  $\mathbf{O}_t$  is the latest  $T_o$  observations. This results in the modified sampling process for Falcon-enhanced diffusion policy (Eq. 13). The corresponding pseudocode is provided in Algorithm 2.

$$\mathbf{x}_{k-1} = \sqrt{\alpha_{k-1}} \left( \frac{\mathbf{x}_k - \sqrt{1 - \alpha_k} \epsilon_\theta(\mathbf{x}_k, k)}{\sqrt{\alpha_k}} \right) + \sqrt{1 - \alpha_{k-1} - \sigma_k^2} \cdot \epsilon_\theta(\mathbf{x}_k, k) + \sigma_k \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (12)$$

$$\mathbf{a}_{t:t+T_p}^{k-1} = \sqrt{\alpha_{k-1}} \left( \frac{\mathbf{a}_{t:t+T_p}^k - \sqrt{1 - \alpha_k} \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^k, k)}{\sqrt{\alpha_k}} \right) + \sqrt{1 - \alpha_{k-1} - \sigma_k^2} \cdot \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^k, k) + \sigma_k \mathbf{z} \quad (13)$$

---

**Algorithm 2** Falcon: Fast Visuomotor Policies via Partial Denoising (DDIM)

**Require:** Diffusion model  $\epsilon_\theta$  with noise scheduler  $\bar{\alpha}_k$ , variance  $\sigma_k^2$ , threshold  $\epsilon$ , exploration probability  $\delta$ , latest  $T_o$  observations  $\mathbf{O}_t$ , latent buffer  $\mathcal{B}$ ,  $M + 1$  denoising steps  $\{k_i\}_{i=0}^M$ .

```

1: for  $t = 1, \dots, T$  do
2:   if  $t = 1$  then
3:      $\mathbf{a}_{t:t+T_p}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
4:     for  $i = M, \dots, 1$  do
5:        $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{a}_{t:t+T_p}^{k_i}\}$ .
6:        $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $k_i > 1$ , else  $\mathbf{z} \leftarrow \mathbf{0}$ 
7:        $\mathbf{a}_{t:t+T_p}^{k_{i-1}} \leftarrow \sqrt{\alpha_{k_{i-1}}} \left( \frac{\mathbf{a}_{t:t+T_p}^{k_i} - \sqrt{1 - \alpha_{k_i}} \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i)}{\sqrt{\alpha_{k_i}}} \right) + \sqrt{1 - \alpha_{k_{i-1}} - \sigma_{k_{i-1}}^2} \cdot \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i) + \sigma_{k_i} \mathbf{z}$ 
8:     end for
9:   end if
10:  if  $t > 1$  then
11:     $\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i} \leftarrow \frac{1}{\sqrt{\alpha_{k_i}}} \left( \mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i} - \sqrt{1 - \bar{\alpha}_{k_i}} \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i}, k_i) \right) \quad \forall \mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i} \in \mathcal{B}$ 
12:     $\mathcal{S} \leftarrow \{\mathbf{a}_{\tau:\tau-T_a+T_p}^k : \|\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i} - \tilde{\mathbf{a}}_{\tau:\tau-T_a+T_p}^k\|_2 < \epsilon, \forall \tau < t, k \in \{k_i\}_{i=0}^M\}$ 
13:    Sample  $\mathbf{a}_{\tau:\tau+T_p}^{k_s}$  according to Eq. 10
14:     $\mathbf{a}_{t:t+T_p}^{k_s} \leftarrow \mathbf{a}_{\tau:\tau+T_p}^{k_s}$ 
15:    for  $i = s, \dots, 1$  do
16:       $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{a}_{t:t+T_p}^{k_i}\}$ .
17:       $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $k_i > 1$ , else  $\mathbf{z} \leftarrow \mathbf{0}$ 
18:       $\mathbf{a}_{t:t+T_p}^{k_{i-1}} \leftarrow \sqrt{\alpha_{k_{i-1}}} \left( \frac{\mathbf{a}_{t:t+T_p}^{k_i} - \sqrt{1 - \alpha_{k_i}} \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i)}{\sqrt{\alpha_{k_i}}} \right) + \sqrt{1 - \alpha_{k_{i-1}} - \sigma_{k_{i-1}}^2} \cdot \epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i) + \sigma_{k_i} \mathbf{z}$ 
19:    end for
20:  end if
21: end for
```

---

## B.2. Falcon with DPMSSolver

Given a noise prediction network  $\epsilon_\theta$ , denoising steps  $\{k_i\}_{i=0}^M$ , DPMSSolver's sampling process follows Eq. 14, where  $h_{i-1} = \log \frac{\alpha_{k_{i-1}}}{\sigma_{k_{i-1}}} - \log \frac{\alpha_{k_i}}{\sigma_{k_i}}$ . To integrate Falcon, we replace  $\mathbf{x}_{k_i}$  with  $\mathbf{a}_{t:t+T_p}^{k_i}$  and substitute  $\epsilon_\theta(\mathbf{x}_{k_i}, k_i)$  with  $\epsilon_\theta(\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i)$ , where  $\mathbf{O}_t$  represents the latest  $T_o$  observations. This yields the sampling process in diffusion policy with DPMSSolver, as expressed in Eq. 15. The corresponding pseudocode is provided in Algorithm 3.

$$\mathbf{x}_{k_{i-1}} = \frac{\alpha_{k_{i-1}}}{\alpha_{k_i}} \mathbf{x}_{k_i} - \sigma_{k_{i-1}} (e^{h_{i-1}} - 1) \epsilon_\theta (\mathbf{x}_{k_i}, k_i) \quad (14)$$

$$\mathbf{a}_{t:t+T_p}^{k_{i-1}} = \frac{\alpha_{k_{i-1}}}{\alpha_{k_i}} \mathbf{a}_{t:t+T_p}^{k_i} - \sigma_{k_{i-1}} (e^{h_{i-1}} - 1) \epsilon_\theta (\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i) \quad (15)$$

---

**Algorithm 3** Falcon: Fast Visuomotor Policies via Partial Denoising (DPMSSolver)

**Require:** Diffusion model  $\epsilon_\theta$  with noise scheduler  $\bar{\alpha}_k$ , variance  $\sigma_k^2$ , threshold  $\epsilon$ , exploration probability  $\delta$ , latest  $T_o$  observations  $\mathbf{O}_t$ , latent buffer  $\mathcal{B}$  and  $M + 1$  denoising steps  $\{k_i\}_{i=0}^M$ .

```

1: for  $t = 1, \dots, T$  do
2:   if  $t = 1$  then
3:      $\mathbf{a}_{t:t+T_p}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
4:      $\mathbf{a}_{t:t+T_p}^{k_M} \leftarrow \mathbf{a}_{t:t+T_p}^K$ 
5:   for  $i = M, \dots, 1$  do
6:      $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{a}_{t:t+T_p}^{k_i}\}$ .
7:      $h_{i-1} \leftarrow \log \frac{\alpha_{k_{i-1}}}{\sigma_{k_{i-1}}} - \log \frac{\alpha_{k_i}}{\sigma_{k_i}}$ 
8:      $\mathbf{a}_{t:t+T_p}^{k_{i-1}} = \frac{\alpha_{k_{i-1}}}{\alpha_{k_i}} \mathbf{a}_{t:t+T_p}^{k_i} - \sigma_{k_{i-1}} (e^{h_{i-1}} - 1) \epsilon_\theta (\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i)$ 
9:   end for
10:  end if
11:  if  $t > 1$  then
12:     $\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i} \leftarrow \frac{1}{\sqrt{\alpha_{k_i}}} (\mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i} - \sqrt{1 - \bar{\alpha}_i} \epsilon_\theta (\mathbf{O}_t, \mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i}, k_i)) \quad \forall \mathbf{a}_{\tau:\tau-T_a+T_p}^{k_i} \in \mathcal{B}$ 
13:     $\mathcal{S} \leftarrow \{\mathbf{a}_{\tau:\tau+T_p}^k : \|\hat{\mathbf{a}}_{\tau:\tau-T_a+T_p}^{k_i} - \tilde{\mathbf{a}}_{\tau:\tau-T_a+T_p}^k\|_2 < \epsilon, \forall \tau < t\}$ 
14:    Sample  $\mathbf{a}_{\tau:\tau+T_p}^{k_s} \sim P(\mathbf{a}_{\tau:\tau+T_p}^{k_s}) = \frac{\exp k_s}{\sum \exp k_i}$  in  $\mathcal{S}$ 
15:     $\mathbf{a}_{t:t+T_p}^{k_s} \leftarrow \mathbf{a}_{\tau:\tau+T_p}^{k_s}$ 
16:    for  $i = s, \dots, 1$  do
17:       $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathbf{a}_{t:t+T_p}^{k_i}\}$ .
18:       $h_{i-1} \leftarrow \log \frac{\alpha_{k_{i-1}}}{\sigma_{k_{i-1}}} - \log \frac{\alpha_{k_i}}{\sigma_{k_i}}$ 
19:       $\mathbf{a}_{t:t+T_p}^{k_{i-1}} = \frac{\alpha_{k_{i-1}}}{\alpha_{k_i}} \mathbf{a}_{t:t+T_p}^{k_i} - \sigma_{k_{i-1}} (e^{h_{i-1}} - 1) \epsilon_\theta (\mathbf{O}_t, \mathbf{a}_{t:t+T_p}^{k_i}, k_i)$ 
20:    end for
21:  end if
22: end for
```

---

## C. Detail performance of Falcon with 3D Diffusion Policy

In this section, we provide a detailed results of Falcon’s acceleration performance when applied to 3D Diffusion Policy ([Ze et al., 2024b](#)) in MetaWorld environments. Falcon is integrated with DDIM using 10-step discretization(we call 3D FalconDDIM), following the original 3D Diffusion Policy architecture to ensure a fair comparison.

Tables 7, 8 and 9 present the success rates, NFE and speedup respectively, for both 3D Diffusion Policy and 3D FalconDDIM. These results further validate Falcon’s effectiveness in reducing inference time while maintaining task performance across different robotic manipulation tasks.

**Table 7. Detailed results for 39 simulated tasks with success rates.** We evaluated 39 challenging tasks using 50 random seeds and reported the average success rate (%) and standard deviation for each task individually. The 3D FalconDDIM algorithm demonstrates nearly no performance drop.

		Meta-World (Easy)							
Alg \ Task		Button Press	Coffee Button	Plate Slide Back Side	Plate Slide Side	Window Close	Window Open		
3D Diffusion Policy		100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0		
3D FalconDDIM		100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0		
		Meta-World (Easy)							
Alg \ Task		Button Press Topdown	Button Press Topdown Wall	Button Press Wall	Peg Unplug Side	Door Close	Door Lock		
3D Diffusion Policy		100 ± 0	99 ± 2	99 ± 1	75 ± 5	100 ± 0	98 ± 2		
3D FalconDDIM		100 ± 0	100 ± 0	100 ± 0	75 ± 43	100 ± 0	96 ± 19		
		Meta-World (Easy)							
Alg \ Task		Door Open	Door Unlock	Drawer Close	Drawer Open	Faucet Close	Faucet Open		
3D Diffusion Policy		99 ± 1	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0		
3D FalconDDIM		100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0		
		Meta-World (Easy)							
Alg \ Task		Handle Pull Side	Lever Pull	Plate Slide	Plate Slide Back	Dial Turn	Reach		
3D Diffusion Policy		85 ± 3	79 ± 8	100 ± 1	99 ± 0	92 ± 27	68 ± 46		
3D FalconDDIM		87 ± 33	81 ± 39	86 ± 34	100 ± 0	89 ± 31	68 ± 46		
		Meta-World (Medium)					Meta-World (Hard)		
Alg \ Task		Hammer	Basketball	Push Wall	Box Close	Sweep	Sweep Into	Assembly	Hand Insert
3D Diffusion Policy		88 ± 32	98 ± 2	88 ± 32	56 ± 49	96 ± 3	15 ± 5	99 ± 1	12 ± 32
3D FalconDDIM		83 ± 37	100 ± 0	88 ± 32	55 ± 49	100 ± 0	13 ± 33	99 ± 9	26 ± 43
		Meta-World (Hard)		Meta-World (Very Hard)					
Alg \ Task		Push		Shelf Place	Disassemble	Stick Pull	Stick Push	Pick Place	Wall
3D Diffusion Policy		51 ± 3		52 ± 49	72 ± 44	68 ± 46	97 ± 4	80 ± 40	
3D FalconDDIM		53 ± 49		47 ± 49	75 ± 43	68 ± 46	100 ± 0	87 ± 33	

**Table 8. Detailed results for 39 simulated tasks with NFE.** We evaluated 39 challenging tasks using 50 random seeds and reported the average Number of Function Evaluations (nfe) per action generation and standard deviation for each domain individually. The 3D FalconDDIM algorithm reduces the nfe to a range of 2-4 compared to the 3D Diffusion Policy

		Meta-World (Easy)					
Alg \ Task		Button Press	Coffee Button	Plate Slide Back Side	Plate Slide Side	Window Close	Window Open
3D Diffusion Policy		10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0
3D FalconDDIM		<b>2.12 ± 0.05</b>	<b>2.47 ± 0.09</b>	<b>3.91 ± 0.22</b>	<b>3.27 ± 0.68</b>	<b>2.54 ± 0.18</b>	<b>3.58 ± 0.90</b>
		Meta-World (Easy)					
Alg \ Task		Button Press Topdown	Button Press Topdown Wall	Button Press Wall	Peg Unplug Side	Door Close	Door Lock
3D Diffusion Policy		10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0
3D FalconDDIM		<b>2.48 ± 0.20</b>	<b>2.63 ± 0.40</b>	<b>3.36 ± 0.61</b>	<b>3.25 ± 0.38</b>	<b>2.81 ± 0.05</b>	<b>4.06 ± 1.00</b>
		Meta-World (Easy)					
Alg \ Task		Door Open	Door Unlock	Drawer Close	Drawer Open	Faucet Close	Faucet Open
3D Diffusion Policy		10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0
3D FalconDDIM		<b>3.04 ± 0.26</b>	<b>4.70 ± 0.86</b>	<b>2.91 ± 0.47</b>	<b>3.90 ± 0.18</b>	<b>4.06 ± 0.62</b>	<b>4.94 ± 0.78</b>
		Meta-World (Easy)					
Alg \ Task		Handle Pull Side	Lever Pull	Plate Slide	Plate Slide Back	Dial Turn	Reach
3D Diffusion Policy		10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0
3D FalconDDIM		<b>3.59 ± 0.51</b>	<b>3.79 ± 0.77</b>	<b>3.11 ± 0.15</b>	<b>3.49 ± 0.16</b>	<b>4.26 ± 1.16</b>	<b>2.85 ± 0.05</b>
		Meta-World (Medium)					
Alg \ Task		Hammer	Basketball	Push Wall	Box Close	Sweep	Sweep Into
3D Diffusion Policy		10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0
3D FalconDDIM		<b>3.25 ± 0.60</b>	<b>2.97 ± 0.14</b>	<b>3.35 ± 0.22</b>	<b>4.51 ± 1.12</b>	<b>3.09 ± 0.18</b>	<b>3.59 ± 0.19</b>
		Meta-World (Hard)					
Alg \ Task		Push	Shelf Place	Disassemble	Stick Pull	Stick Push	Pick Place Wall
3D Diffusion Policy		10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0	10 ± 0
3D FalconDDIM		<b>2.88 ± 0.38</b>	<b>4.28 ± 2.33</b>	<b>3.25 ± 0.37</b>	<b>3.78 ± 0.73</b>	<b>3.41 ± 0.32</b>	<b>3.04 ± 0.15</b>

**Table 9. Detailed results for 39 simulated tasks with speedup.** We evaluated 39 challenging tasks using 50 random seeds and reported the average speedup and standard deviation for each task individually. The 3D FalconDDIM algorithm reduces the NFE to a range of 2-4 compared to the 3D Diffusion Policy

		Meta-World (Easy)					
Alg \ Task		Button Press	Coffee Button	Plate Slide Back Side	Plate Slide Side	Window Close	Window Open
3D FalconDDIM		4.71 ± 0.12	4.03 ± 0.14	2.55 ± 0.14	3.05 ± 0.60	3.93 ± 0.28	2.79 ± 0.66
		Meta-World (Easy)					
Alg \ Task		Button Press Topdown	Button Press Topdown Wall	Button Press Wall	Peg Unplug Side	Door Close	Door Lock
3D FalconDDIM		4.02 ± 0.32	3.80 ± 0.48	2.97 ± 0.47	3.07 ± 0.34	3.55 ± 0.06	2.46 ± 0.51
		Meta-World (Easy)					
Alg \ Task		Door Open	Door Unlock	Drawer Close	Drawer Open	Faucet Close	Faucet Open
3D FalconDDIM		3.28 ± 0.26	2.12 ± 0.32	3.42 ± 0.36	2.55 ± 0.12	2.45 ± 0.35	2.02 ± 0.35
		Meta-World (Easy)					
Alg \ Task		Handle Pull Side	Lever Pull	Plate Slide	Plate Slide Back	Dial Turn	Reach
3D FalconDDIM		2.77 ± 0.36	2.63 ± 0.44	3.21 ± 0.15	2.86 ± 0.14	2.34 ± 0.56	3.50 ± 0.06
		Meta-World (Medium)					
Alg \ Task		Hammer	Basketball	Push Wall	Box Close	Sweep	Sweep Into
3D FalconDDIM		3.25 ± 0.60	2.97 ± 0.14	3.35 ± 0.22	4.51 ± 1.12	3.09 ± 0.18	3.59 ± 0.19
		Meta-World (Hard)					
Alg \ Task		Push	Shelf Place	Disassemble	Stick Pull	Stick Push	Pick Place Wall
3D FalconDDIM		2.88 ± 0.38	2.33 ± 0.65	3.07 ± 0.30	2.64 ± 0.44	2.92 ± 0.25	3.28 ± 0.14

## D. Experiment Details

In this section, we provide the detailed experimental setup for Robomimic and analyze Falcon’s memory cost compared to the original samplers (DDPM, DDIM, and DPMSolver). Table 10 reports the hyperparameter settings and the peak memory usage for each experiment.

To evaluate Falcon’s computational overhead, we measure the peak memory cost, denoted in the format (original sampler cost) + (incremental cost due to Falcon integration). As shown in Table 10, Falcon introduces an additional 12 MB of memory overhead, which is negligible compared to the original 1876 MB cost. This demonstrates that Falcon achieves acceleration with minimal memory overhead, making it a practical and efficient enhancement to diffusion-based policies.

	DDPM+Falcon					DDIM+Falcon					DPMSolver+Falcon				
	$\epsilon$	$\delta$	$k_{\min}$	$ \mathcal{B} $	Peak Memory Cost	$\epsilon$	$\delta$	$k_{\min}$	$ \mathcal{B} $	Peak Memory Cost	$\epsilon$	$\delta$	$k_{\min}$	$ \mathcal{B} $	Peak Memory Cost
Lift ph	0.04	0.1	20	50	1876+12 MB	0.05	0.2	15	50	1876+12 MB	0.08	0.2	20	50	1876+12 MB
Lift mh	0.04	0.1	20	50	1876+12 MB	0.03	0.25	10	50	1876+12 MB	0.005	0.2	20	50	1876+12 MB
Can ph	0.01	0.1	20	50	1876+12 MB	0.01	0.20	8	50	1876+12 MB	0.003	0.2	20	50	1876+12 MB
Can mh	0.01	0.1	20	50	1876+12 MB	0.005	0.20	8	50	1876+12 MB	0.003	0.2	20	50	1876+12 MB
Square ph	0.04	0.1	25	50	1876+12 MB	0.01	0.20	8	50	1876+12 MB	0.003	0.2	25	20	1876+12 MB
Square mh	0.04	0.1	25	50	1876+12 MB	0.005	0.20	3	50	1876+12 MB	0.005	0.2	20	20	1876+12 MB
Transport ph	0.01	0.2	25	50	1876+12 MB	0.001	0.33	8	50	1876+12 MB	0.005	0.2	20	20	1876+12 MB
Transport mh	0.01	0.2	25	50	1876+12 MB	0.01	0.33	5	50	1876+12 MB	0.003	0.2	60	50	1876+12 MB
ToolHang ph	0.01	0.1	20	50	1876+12 MB	0.003	0.33	5	50	1876+12 MB	0.003	0.2	60	50	1876+12 MB

Table 10. Hyperparameters and Memory Cost in Robomimic.

## E. Additional Experiments

### E.1. Acceleration on Vision-Language-Action Foundation Model (RDT)

To assess Falcon compatibility with pre-trained diffusion foundation models, we integrate it with RDT-1B (Liu et al., 2024), a vision language action foundation model trained in ManiSkill2. Falcon is applied directly to RDT’s DPMSolver++ inference without retraining. The official model weights are obtained from the RDT repository on <https://huggingface.co/robotics-diffusion-transformer/maniskill-model>.

As shown in Table 11, Falcon achieves substantial inference speedups (31x and 34x vs. DDPM) with no loss in success rate on task PickCube (see Fig. 9) and task PushCube (see Fig. 10), demonstrating its plug-and-play compatibility with VLA Foundation model.

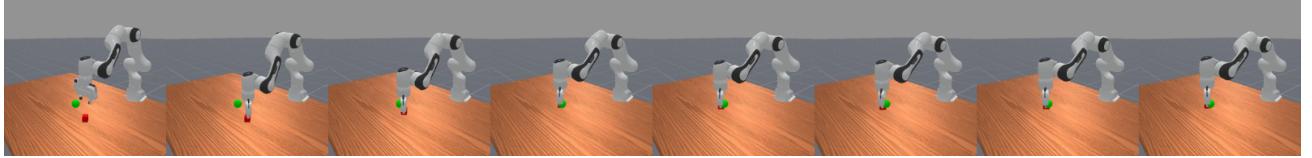


Figure 9. RDT evaluated in PickCube with Falcon

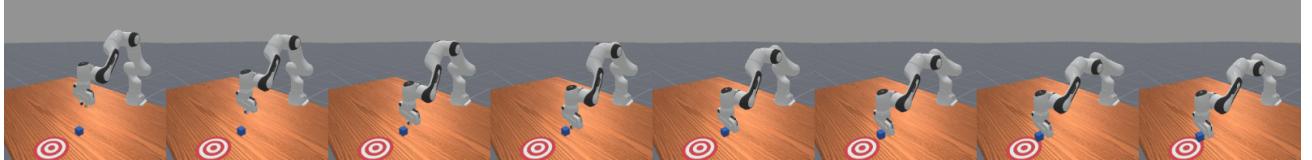


Figure 10. RDT evaluated in PushCube with Falcon

PickCube					
Method	Success Rate	NFE	Time (s)	Speedup	GPU MEMS (MB)
DPMSolver++ (5 steps)	0.75	5.00	0.08	20x	15479
DPMSolver++ (5 steps)+Falcon	0.75	3.22	0.04	31x	15483

PushCube					
Method	Success Rate	NFE	Time (s)	Speedup	GPU MEMS (MB)
DDPMSSolver++ (5 steps)	1.00	5.00	0.08	20x	15483
DPMSolver++ (5 steps)+Falcon	1.00	2.91	0.05	34x	15483

Table 11. Falcon improves inference speed for RDT on unseen ManiSkill tasks ( $T_p = 64$ ,  $T_a = 32$ ,  $T_o = 2$ ,  $\epsilon = 0.02$ ,  $|\mathcal{B}| = 2$ ) using the same pre-trained checkpoint. Speedup is relative to 100-step DDPM. Falcon achieves comparable performance with significantly fewer denoising steps and lower runtime.

## E.2. Additional Ablation on Threshold $\epsilon$

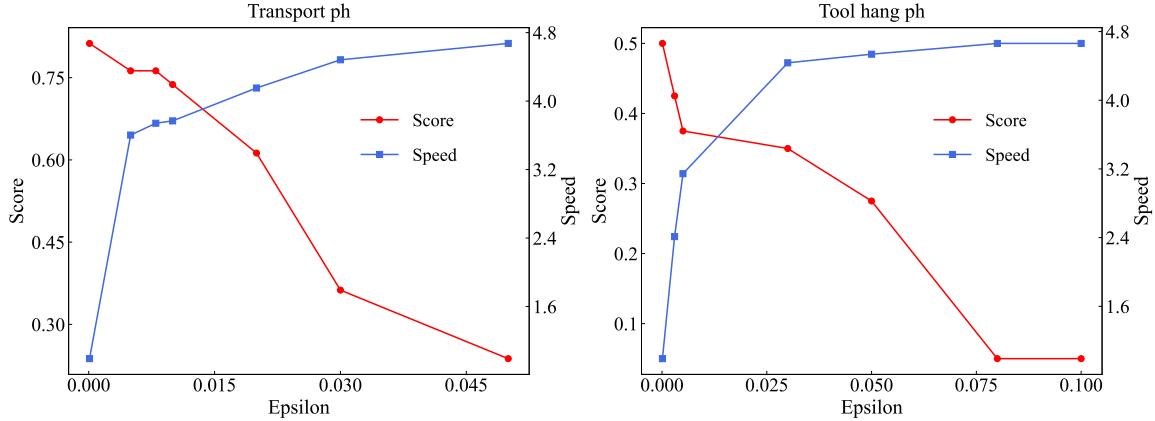
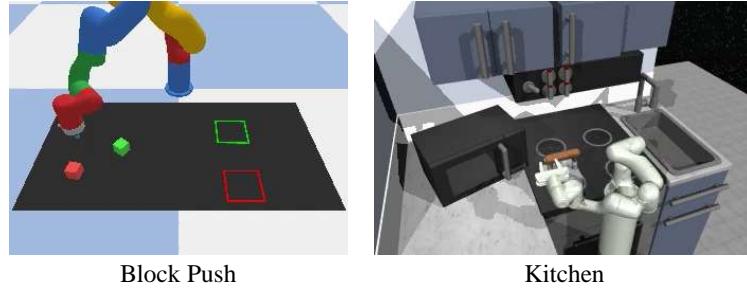


Figure 11. **Additional Ablation Analysis on threshold  $\epsilon$ .** We conduct extended experiments with DDPM+Falcon on *Transport ph* and *Tool hang ph*, varying the threshold  $\epsilon$ . Results show a trade-off between success rate (Score) and speedup.

## E.3. Falcon achieves accelerating diffusion policy in long-horizon tasks

To evaluate Falcon’s effectiveness in long-horizon tasks, we apply it to the Franka Kitchen environment. As shown in Table 12, Falcon maintains the same high success rates as the original DDPM model, confirming that the acceleration introduced by Falcon does not compromise task performance, even for long-horizon tasks. The key benefit of Falcon lies in its ability to reduce NFE and accelerate the denoising process, which is evident in the speedup shown in Table 13. Falcon achieves a **5.25x** speedup in Kitchen, which shows that Falcon can accelerate DDPM in long-horizon tasks and can be adopted into transformer architecture.



	BlockPush		Kitchen			
	p1	p2	p1	p2	p3	p4
DDPM	0.99	0.94	1.00	0.99	0.99	0.96
DDPM+Falcon	0.99	0.97	1.00	0.99	0.99	0.96

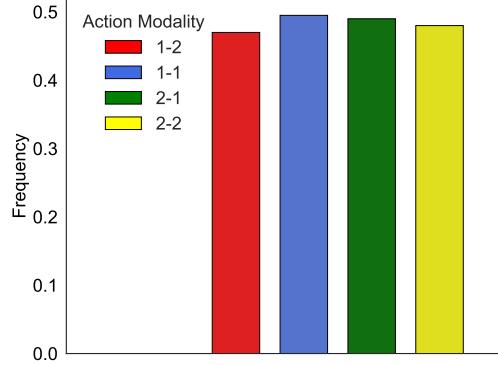
Table 12. **Success Rate in BlockPush and Kitchen.** For BlockPush,  $p_x$  refers to the frequency of pushing  $x$  blocks into the targets. For Kitchen,  $p_x$  refers to the frequency of interacting with  $x$  or more objects.

	BlockPush		Kitchen	
	NFE	Speedup	NFE	Speedup
DDPM	100	1.0x	100	1.0x
DDPM+Falcon	32.7	<b>2.8x</b>	19.03	<b>5.25x</b>

Table 13. **Speedup in BlockPush and Kitchen.** Falcon can accelerate diffusion policy in long-horizon tasks.

#### E.4. More results of Multimodality

In this section, we provide a visualization of the multimodal action distributions generated by Falcon in the BlockPush environment. As shown in Fig. 12, the frequency distribution of different action modalities in the BlockPush task, where each modality corresponds to a distinct combination of blocks being pushed into two squares. The chart illustrates the uniformity of the action modality frequencies, with the four modalities (1-2, 1-1, 2-1, and 2-2) being equally represented. This visualization confirms that Falcon is capable of expressing multimodal actions, effectively handling different action combinations without bias.

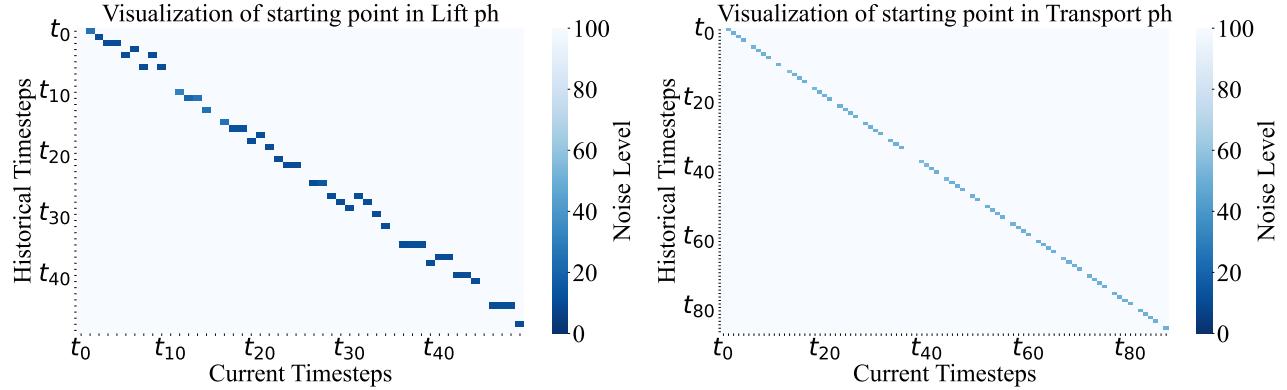


**Figure 12. Action Modalities Distribution in BlockPush Task.** The bar chart shows the frequency of different policy modalities for pushing two blocks into two squares in any order. The modalities are represented as 1-2, 1-1, 2-1, and 2-2, where the numbers indicate the block number and square number respectively. The chart illustrates the frequency distribution of the action modalities generated by Falcon in the BlockPush task.

## F. Visualization of Starting Points

This section visualizes where Falcon starts the denoising process at each time step, specifically from which past time step and at what noise level the partial denoised actions originate. We analyze two tasks: Lift ph and Transport ph, representing high and low acceleration scenarios, respectively.

As shown in Fig.13 (left), in the Lift task, Falcon consistently starts denoising from the partial denoised action of the previous time step, with a low noise level. This indicates a significant reduction in sampling steps, leading to substantial acceleration. In contrast, Fig.13 (right) shows that the Transport task starts from actions with higher noise levels, limiting the acceleration effect. This suggests that Falcon’s speedup is more pronounced in tasks with smoother, more predictable action transitions.



**Figure 13. Visualization of denoising starting point.** This figure shows the heatmaps  $A \in \mathbb{R}^{T \times T}$  in Lift ph task (left) and Transport ph task (right).  $A[j, i] = k$  means that at time step  $i$ , Falcon starts denoising at the partial denoised actions  $\alpha_{j:j+T_p}^k$ .