Steering Diffusion Policies with Value-Guided Denoising

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Diffusion-based robotic policies trained with imitation learning has achieved remarkable results in complex manipulation tasks. However, such policies are constrained by the quality and coverage of their training data, limiting their adaptation to new environments. Existing approaches to address this obstacle typically rely on fine-tuning the diffusion model, which can be unstable and require costly human demonstrations. We instead study the online adaptation of pretrained diffusion policies without parameter updates. We introduce *Value-Guided Denoising* (VGD), a simple method that steers a frozen diffusion policy using gradients from a reinforcement-learned value function. At inference, VGD guides diffusion denoising steps toward actions with higher Q-values. This enables adaptation with only black-box access to the pretrained policy. On Robomimic benchmarks, our method achieves substantially higher success rates than existing RL-with-diffusion approaches. These results demonstrate that diffusion policies can be steered efficiently at deployment, yielding strong performance gains with minimal data and computation. Code available at https://anonymous.4open.science/r/VGD.

1 Introduction

Large-scale pretraining has produced highly capable foundation models in vision and language [1, 2, 3]. Inspired by this success, robot learning has achieved impressive results with **imitation learning**, where expert demonstrations train policies via supervised behavior cloning (BC). Diffusion models in particular have emerged as a strong parameterization for BC policies, achieving state-of-the-art results in manipulation [4, 5, 6]. Due to their scalability and simplicity, such methods comprise the emerging paradigm for robot learning.

However, imitation learning is inherently limited by its data. Policy performance depends on the quality, coverage, and diversity of data [7]. At test time, small imprecisions in control can accumulate, eventually leading the policy to states far from those in demonstrations. This leads to degraded behavior, such as misaligned grasps or mistimed gripper closure [8]. Consequently, BC-learned policies can struggle to achieve satisfactory performance, especially in novel environments and under nuisance shifts such as changes in lighting or camera pose [9, 10].

How can we improve the proficiency of diffusion-based BC policies? A natural solution is fine-tuning on additional data. However, collecting quality demonstrations require expensive and time-consuming procedures like human teleoperation [11]. Recent work has used reinforcement learning (RL) to fine-tune policies using autonomous interactions between the agent and the environment [12, 13, 14, 15, 16, 17]. But these approaches are often too sample-inefficient or unstable for practical use [17, 15]. These limitations motivate a different question: can we adapt diffusion policies without updating their parameters, and simply steer them towards better actions at inference?

Submitted to NeurIPS 2025 Workshop on Embodied World Models for Decision Making. Do not distribute.

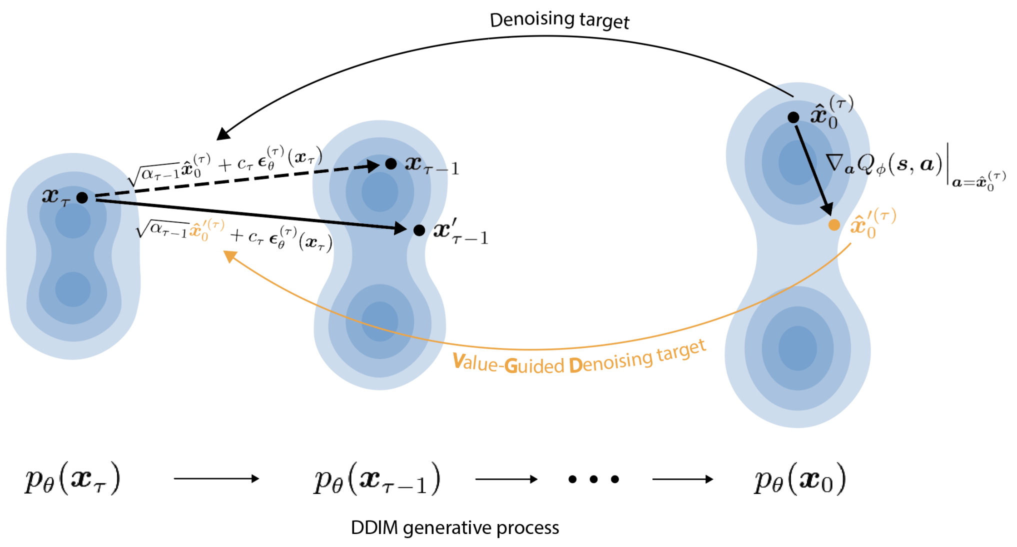
Denoising target
$\sqrt{\alpha_{\tau-1}}\hat{\boldsymbol{x}}_0^{(\tau)} + c_\tau \boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau)$
$\boldsymbol{x}_{\tau-1}$
$\boldsymbol{x}_\tau$
$\sqrt{\alpha_{\tau-1}}\hat{\boldsymbol{x}}_0'^{(\tau)} + c_\tau \boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau)$
$\boldsymbol{x}'_{\tau-1}$
$\hat{\boldsymbol{x}}_0^{(\tau)}$
$\nabla_{\boldsymbol{a}} Q_\phi(\boldsymbol{s},\boldsymbol{a})\big|_{\boldsymbol{a}=\hat{\boldsymbol{x}}_0^{(\tau)}}$
$\hat{\boldsymbol{x}}_0'^{(\tau)}$
Value-Guided Denoising target
$p_\theta(\boldsymbol{x}_\tau)$
$p_\theta(\boldsymbol{x}_{\tau-1})$
$p_\theta(\boldsymbol{x}_0)$
DDIM generative process

Figure 1: Illustration of our approach, *Value-Guided Denoising* (VGD). In a standard application of a diffusion-based BC policy, we sample an initial noise latent $\boldsymbol{x}_T$, then successively denoise it through the DDIM sampling process. At each denoising step t, the standard DDIM decoding maps $\boldsymbol{x}_\tau$ to $\boldsymbol{x}_{\tau-1} = \sqrt{\alpha_{\tau-1}} \cdot \hat{\boldsymbol{x}}_0^{(\tau)} + \sqrt{1-\alpha_{\tau-1}} \cdot \boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau)$, where $\hat{\boldsymbol{x}}_0$ is the model's predicted denoising target (given by Equation 1) and $\boldsymbol{\epsilon}_\theta^{(\tau)}$ is the predicted noise. To steer the output of the diffusion model towards more desirable actions, we shift the predicted target $\hat{\boldsymbol{x}}_0^{(\tau)}$ along the gradient of a RL-learnt critic to $\hat{\boldsymbol{x}}_0'^{(\tau)}$, which we use as the new denoising target to calculate the next latent $\boldsymbol{x}'_{\tau-1}$. We repeat this procedure repeats throughout the denoising process, steering the pretrained diffusion model onto more desirable actions without altering its weights.

This prompts us to examine the **diffusion sampling process**. Our insight is that each denoising step in the diffusion process is a weighted sum of the predicted denoised target $\hat{\boldsymbol{x}}_0^{(\tau)}$ and the predicted noise $\epsilon_\theta^{(t)}$ [18]. We observe that $\hat{\boldsymbol{x}}_0^{(\tau)}$ can be nudged toward higher-value actions using gradients from a learned critic, while leaving $\epsilon_\theta^{(t)}$ unchanged. Details can be found in Section 2. This procedure enables policy steering at inference time, using only black-box access to the pretrained model. Crucially, it also **avoids unstable backpropagation** through the full diffusion chain and sidesteps the challenges of fine-tuning large, complex architectures [12, 19, 20]. Instead, **we only train a lightweight critic** on state-action pairs – a standard RL task. Figure 1 illustrates this process.

We formalize this steering process as *Value-Guided Denoising* (VGD). Compared to prior RL-with-diffusion methods, we show that VGD leverages the structure of diffusion models to steer actions with greater sample-efficiency. On Robomimic benchmarks [21], VGD substantially **improves success rates over state-of-the-art baselines**.

2 Preliminaries

Markov Decision Process (MDP) We consider a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$. At time t, the agent observes $\boldsymbol{s}_t \in \mathcal{S}$ (i.e. environment and proprioceptive states), takes action $\boldsymbol{a}_t \in \mathcal{A}$, receives reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t)$, and transitions to the next state $\boldsymbol{s}_{t+1} \sim P(\cdot \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$. For a given policy π, the Q-function $Q^\pi(\boldsymbol{s}, \boldsymbol{a})$ represents the the γ-discounted return of policy π from taking action $\boldsymbol{a}$ after observing state $\boldsymbol{s}$. That is,

$$Q^\pi(\boldsymbol{s}, \boldsymbol{a}) := \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \mid \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a} \right].$$

In our VGD algorithm, this state-action critic is the only network we train – all other components of the diffusion policy remain frozen.

Diffusion policies Diffusion policies treat action generation as conditional denoising [5]. Instead of predicting an action chunk $\boldsymbol{x}_0$ directly, the policy learns to invert a forward noising process that gradually corrupts $\boldsymbol{x}_0$ into Gaussian noise. Concretely, given $\boldsymbol{x}_0$ and a decreasing sequence $\{\alpha_\tau\}_{\tau=1}^{T} \in (0,1]^T$, the forward process produces noisy latents $\boldsymbol{x}_\tau$ via

$$q(\boldsymbol{x}_\tau \mid \boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_\tau}\,\boldsymbol{x}_0,\, (1-\alpha_\tau)\mathbf{I}).$$

A neural network $\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau, \boldsymbol{s})$, conditioned on the current observation $\boldsymbol{s}$, learns to predict the noise injected at step τ, forming the generative process. At inference, we by sampling $\boldsymbol{x}_T \sim \mathcal{N}(0,1)$ and iteratively denoise it using this network until we obtain an action chunk $\boldsymbol{x}_0$ to execute.

While both DDPM [22] and DDIM [18] samplers are compatible with our method, we focus on DDIM – a popular sampling algorithm that enables faster inference with fewer decoding steps. With DDIM, we update from $\boldsymbol{x}_\tau$ to $\boldsymbol{x}_{\tau-1}$ via

$$\boldsymbol{x}_{\tau-1} = \sqrt{\alpha_{\tau-1}}\underbrace{\left(\frac{\boldsymbol{x}_\tau - \sqrt{1-\alpha_\tau}\cdot\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau, \boldsymbol{s})}{\sqrt{\alpha_\tau}}\right)}_{\text{“predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{\tau-1}-\sigma_\tau^2}\cdot\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau, \boldsymbol{s})}_{\text{“direction pointing to } \boldsymbol{x}_\tau\text{”}} + \underbrace{\sigma_\tau\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau, \boldsymbol{s})}_{\text{random noise}} \tag{1}$$

We set $\sigma_\tau = 0$ so that the decoding process is deterministic given the initial noise $\boldsymbol{x}_T$. Here, the first term can be viewed as an estimate of the clean action output $\boldsymbol{x}_0$ [18]. We denote this term as $\hat{\boldsymbol{x}}_0^{(\tau)}$. Thus, each update is a linear combination of $\hat{\boldsymbol{x}}_0^{(\tau)}$ and the noise prediction $\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau)$.

As the diffusion proceeds and $\tau \to 0$, $\alpha_{\tau-1}$ tends to 1 so that $\boldsymbol{x}_\tau$ converges to $\hat{\boldsymbol{x}}_0^{(\tau)}$. Thus, we can interpret $\hat{\boldsymbol{x}}_0^{(\tau)}$ as an evolving "**denoising target**" toward which the latent trajectory drifts. We are motivated to utilize the denoising targets $\hat{\boldsymbol{x}}_0^{(\tau)}$ for steering because they approximately lie in the distribution of action outputs, which is not true of the intermediate latents $\boldsymbol{x}_\tau$.[1] This makes it a natural interface for steering with a learned value function, as we describe next, eliminating the need for backpropagatation through the diffusion policy in previous methods.

3 Value-Guided Denoising

The VGD algorithm comprises two parts: the diffusion procedure, and the training process. We begin by describing diffusion with VGD.

3.1 Diffusion with VGD

At each denoising step τ, the denoising target

$$\hat{\boldsymbol{x}}_0^{(\tau)} = \frac{\boldsymbol{x}_\tau - \sqrt{1-\alpha_\tau}\cdot\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau, \boldsymbol{s})}{\sqrt{\alpha_\tau}} \tag{2}$$

provides an increasingly accurate proxy for the final action that will be produced by the diffusion process. Our goal is to steer this process so that the final action is biased toward higher-value outcomes. Given a pretrained diffusion model, let π_ϕ^{VGD} denote the policy obtained by applying VGD steering on to this model using critic Q_ϕ. To sample from $\pi_\phi^{\text{VGD}}(\boldsymbol{s})$, we begin by sampling a noisy latent $\boldsymbol{x}_T \sim \mathcal{N}(0,1)$. Then, at each step τ, we treat $\hat{\boldsymbol{x}}_0^{(\tau)}$ as an action candidate, shift it in the direction of increasing Q-value, then use this new denoising target to update $\boldsymbol{x}_\tau$.

[1]The intermediate latents lie between the standard Gaussian and the distribution of desired action outputs, so they cannot be evaluated by a state-action critic effectively. For more analysis on the differences between the two terms in the realm of image generation, see Section 4 in [22].

Specifically, given $\boldsymbol{x}_\tau$ at step τ, we first compute the denoising target $\hat{\boldsymbol{x}}_0^{(\tau)}$ using Equation 2, then derive the new denoising target as

$$\hat{\boldsymbol{x}}_0^{\prime(\tau)} := \hat{\boldsymbol{x}}_0^{(\tau)} + \lambda \cdot \nabla_{\boldsymbol{a}} Q_\phi(\boldsymbol{s}, \boldsymbol{a})\Big|_{\boldsymbol{a}=\hat{\boldsymbol{x}}_0^{(\tau)}}.$$

Here, $\lambda \geq 0$ is a guidance strength, which we anneal over the start of training to stabilise learning (see Appendix C for details). In practice, we parametrize this Q-value critic as an MLP, and use Pytorch's automatic differentiation to compute the gradient above. Then, we simply substitute this new target into Equation 1 to obtain the next latent:

$$\boldsymbol{x}'_{\tau-1} = \sqrt{\alpha_{\tau-1}} \cdot \hat{\boldsymbol{x}}_0^{\prime(\tau)} + \sqrt{1-\alpha_{\tau-1}} \cdot \boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau). \tag{3}$$

We repeat this procedure, using $\boldsymbol{x}'_{\tau-1}$ as the starting latent $\boldsymbol{x}_{\tau-1}$ for the next denoising step, until we obtain the final action output $\boldsymbol{x}_0 = \boldsymbol{a}$. This describes how we sample $\boldsymbol{a} \sim \pi_\phi^{\text{VGD}}(\boldsymbol{s})$. Algorithm 2 provides a summary. As detailed in Appendix C, we also experiment with disabling VGD for the initial denoising steps to improve performance, as the initial denoising targets $\hat{\boldsymbol{x}}_0^{(\tau)}$ are far away from the target action distribution for large τ.

Crucially, no gradients flow through the pretrained diffusion policy $\boldsymbol{\epsilon}_\theta$; we only backpropagate through the critic. This differs from previous policy fine-tuning methods [12, 19, 20]. We steer directly in action space via $\hat{\boldsymbol{x}}_0^{(\tau)}$, which stabilizes guidance and improves sample-efficiency.

3.2 Critic

Algorithm 1 Online Critic Training with Value-Guided Denoising

```
 1: input: frozen diffusion policy $\boldsymbol{\epsilon}_\theta^{(\tau)}$
 2: Initialize critic $Q_\phi$, replay buffer $\mathcal{B}$.
 3: for each environment step do
 4:     Sample $\boldsymbol{a} \sim \pi_\phi^{\text{VGD}}(\boldsymbol{s})$        ▷ Sample action according to Value-Guided Denoising
 5:     Execute action $\boldsymbol{a}$; observe $(r, \boldsymbol{s}')$, and add $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')$ to $\mathcal{B}$
 6:     for $u = 1$ to updates_per_step do
 7:         Sample $\{(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}'_i)\}_{i=1}^B \sim \mathcal{B}$
 8:         for $i = 1$ to batch_size do
 9:             Sample $\boldsymbol{a}'_i \sim \pi_\phi^{\text{VGD}}(\boldsymbol{s}'_i)$
10:             $y_i \leftarrow r_i + \gamma\, Q_\phi(\boldsymbol{s}'_i, \boldsymbol{a}'_i)$        ▷ Form Q-learning targets
11:         end for
12:         Update $\phi$ to minimize $\mathcal{L}_Q = \frac{1}{B}\sum_{i=1}^{B}\big(Q_\phi(\boldsymbol{s}_i, \boldsymbol{a}_i) - y_i\big)^2$
13:     end for
```

VGD only requires a critic $Q_\phi(\boldsymbol{s}, \boldsymbol{a})$. We train this function online with off-policy TD learning. Transitions $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')$ enter a replay buffer. After every interaction, we update the critic for a fixed number of gradient steps. Each update samples a minibatch of transitions from the buffer. For each transition, we resample a new action from $\pi_\phi^{\text{VGD}}(\boldsymbol{s})$ to obtain a candidate action $\boldsymbol{a}'$. This ensures that the target matches the policy used to act.

Finally, we update the critic parameters ϕ by minimizing the squared Bellman error over the batch. Importantly, the pretrained diffusion policy itself is never updated. The only learning occurs in the critic, whose gradients are later used for steering the denoising process. See Algorithm 1 for the pseudocode summary of this procedure. This setup allows us to benefit from reinforcement signals without disturbing the diffusion policy's prior.

In summary, we learn a state–action critic $Q_\phi(\boldsymbol{s}, \boldsymbol{a})$ from replay using standard TD targets. At action time, we treat the frozen diffusion policy as a prior and perform a locally greedy improvement at each denoising step. [2]

[2]This approximates solving $\arg\max_{\boldsymbol{a}} Q_\phi(\boldsymbol{s}, \boldsymbol{a})$ in the neighborhood defined by the diffusion decoder, rather than taking a global argmax. It is therefore not Q-learning nor direct policy optimization in the classic sense: it has no explicit argmax and no actor updates. We experimented with applying Soft-Actor-Critic to steer denoising targets in lieu of gradient ascient on Q_ϕ, but this did not yield significant improvements.

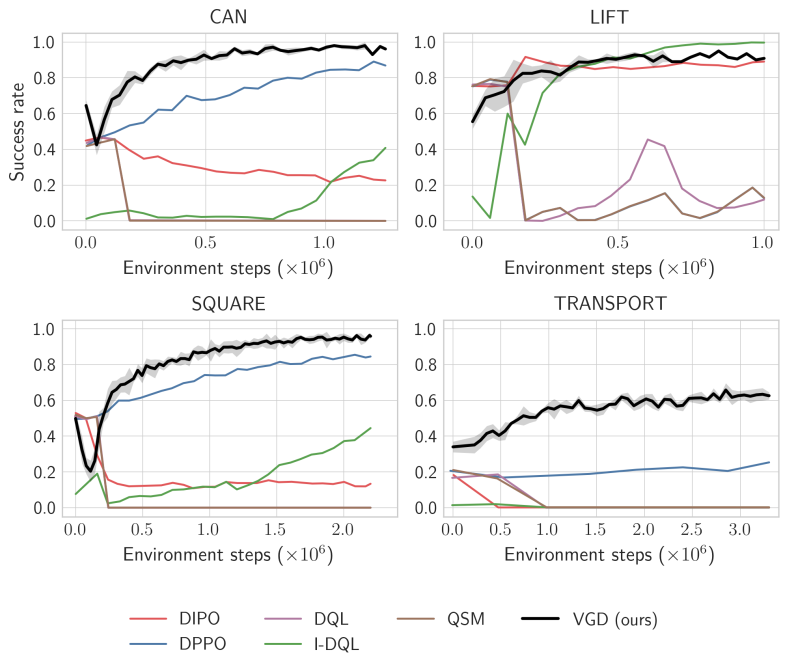

Figure 2: On ROBOMIMIC [21] benchmarks, VGD achieves sample-efficient adaptation of diffusion-based policies using online data.

4 Experiments

We evaluate the ability of *Value-Guided Denoising* (VGD) to improve pretrained diffusion policies using online interaction. Experiments are conducted on ROBOMIMIC [21] manipulation benchmarks, following the diffusion-policy evaluation protocol in prior work. For `Robomimic Square` and `Transport`, we use the diffusion policy checkpoints from Ren et. al [12] as our pretrained model. For `Can` and `Square`, we use the diffusion policies from Wagenmaker et. al [23]. In all cases, we freeze the diffusion policy and apply Algorithm 1 to train a critic online. We also anneal the VGD strength coefficient λ over the initial stage of each run to stabilise learning. Details can be found in Appendix C. Each experiment is averaged over 4 seeds, and error bands show the 95% confidence interval.

We compare against several state-of-the-art methods that combine diffusion policies with reinforcement learning. The first group of these methods directly adapt a pre-trained diffusion policy. DPPO [12] fine-tunes with a PPO-style objective to perform policy-gradient updates to the diffusion model's weights. Additionally, IDQL [13] and IQL [14] add a Q-learning terms to the fine-tuning of the diffusion model. The second group of methods learn from scratch with a diffusion policy. DIPO [17] treats the diffusion model as the policy class and optimizes it online via standard RL gradients, whereas QSM [15] seeks to align the diffusion score with the action-value gradient.

Figure 2 summarizes results. VGD (black) consistently matches or outperforms prior methods across all tasks. In each task, VGD substantially improves upon the pretrained policy, achieving near-perfect success rates on `Can, Lift`, and `Square`. On `Transport`, the most challenging task featuring two robotic arms, VGD delivers the largest relative gains. This highlights how VGD applies corrections without destabilizing the pretrained model. In addition, VGD adapts the pretrained policy with less online data than existing methods on a majority of tasks, highlighting its sample-efficiency.

5 Discussion and Limitations

We introduce *Value-Guided Denoising* (VGD), a method that steers frozen diffusion policies using reinforcement-learned value gradients. VGD provides a practical solution to the performance gaps of BC-trained policies. This makes it a promising tool for **model deployment and sim-to-real transfer**

[24], even when the underlying model weights are not available. VGD applies broadly to any policy with a diffusion action head. This includes generalist **vision-language-action** (VLA) models that condition on language, such as Nvidia's GR00T N1 [25]. We are currently running such experiments. Another natural extension is to pretrain the critic with offline RL, providing a stronger initialization before online adaptation.

Despite its lightweight training requirements, VGD introduces higher inference costs: each environment step requires differentiating the critic at multiple denoising steps. One solution is to use VGD to generate additional demonstrations, and then fine-tune the diffusion model on this data to remove the critic from the loop. Another limitation is that fixed-size gradient updates may be suboptimal for tasks requiring very fine-grained control. Future work could address this by learning an actor to adaptively adjust denoising targets $\hat{\boldsymbol{x}}_0^{(\tau)}$. A greater range of empirical experiments would also demonstrate the broader applicability of our method. We expect to address these limitations in upcoming work.

In summary, VGD highlights how reward signals can be leveraged to guide pretrained diffusion policies efficiently at deployment. We hope this perspective motivates further exploration of inference-time steering as a complement to traditional fine-tuning in robot learning.

References

[1] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, and et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 1877–1901, 2020.

[3] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831, 2021.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, and et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin C. M. Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 2023.

[6] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Rich Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024.

[7] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, 2022.

[8] Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[9] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

[10] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[11] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[12] Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *Proceedings of Robotics: Science and Systems 2024 (RSS)*, 2024.

[13] 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, 2023.

[14] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[15] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[16] Hyun Tae Suh, Guanlong Chou, Hongkai Dai, L. Benjamin Yang, Anirudh Gupta, and Russ Tedrake. Fighting uncertainty with gradients: Offline reinforcement learning via diffusion score matching. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[17] 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. *arXiv preprint arXiv:2305.13122*, 2023.

[18] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021.

[19] Max S. Mark, Tony Gao, Gabriel G. Sampaio, Manoj K. Srirama, Animesh Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

[20] Seohong Park, Qingyun Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

[21] Ajay Mandlekar, Stephen Zhu, Animesh Garg, Jeffrey Booher, Max Spero, Albert Tung, Johnny Gao, Anchit Gupta, Poorva Sundaresan, Emre Orbay, Homer Walke, Stephen Tian, Lianmin Wang, Kyle Hsu, Brijen Thananjeyan, Youngwoon Lee Jiang, Jiayuan Gu, Ajay Jain, Chelsea Finn, Ken Goldberg, and Sergey Yu. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 6840–6851, 2020.

[23] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[24] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *arXiv preprint arXiv:2009.13303*, 2020.

[25] 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, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen 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. *arXiv preprint arXiv:2503.14734*, 2025.

[26] Yueyang Ze, Guangyuan Zhang, Kai Zhang, Cheng Hu, Mingxing Wang, and Hao Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

[27] Akshay Sridhar, Devendra Shah, Christian Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[28] Sudeep Dasari, Oier Mees, Shikai Zhao, Mahesh K. Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

[29] Louis Ankile, Anthony Simeonov, Ilya Shenfeld, and Pulkit Agrawal. Juicer: Data-efficient imitation learning for robotic assembly. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5096–5103. IEEE, 2024.

[30] O. M. Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, John Hejna, Tomaso Kreiman, Chen Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[31] 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. : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[32] Ling Yang, Ziyu Huang, Fan Lei, Yifei Zhong, Yu Yang, Chen Fang, Shusen Wen, Bolei Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

[33] Louis Ankile, Anthony Simeonov, Ilya Shenfeld, Marc Torne, and Pulkit Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[34] Chen Lu, Hongchang Chen, Jialong Chen, Hao Su, Chunyuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 22825–22855. PMLR, 2023.

[35] Bingyi Kang, Xuezhi Ma, Chao Du, Tong Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

[36] Sihong Ding, Kai Hu, Zheng Zhang, Kaixuan Ren, Weinan Zhang, Jun Yu, Jian Wang, and Yanhong Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *arXiv preprint arXiv:2405.16173*, 2024.

[37] Hao Chen, Chen Lu, Chongyi Ying, Hao Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

[38] Hao Chen, Chen Lu, Zihan Wang, Hao Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. *arXiv preprint arXiv:2310.07297*, 2023.

[39] Lingxiao He, Lijun Shen, Jing Tan, and Xiaolong Wang. Aligniql: Policy alignment in implicit q-learning through constrained optimization. *arXiv preprint arXiv:2405.18187*, 2024.

[40] Luka Eyring, Shyamgopal Karthik, Kevin Roth, Alexey Dosovitskiy, and Zeynep Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. *Advances in Neural Information Processing Systems*, 37, 2024.

[41] Jinglong Mao, Xiaoyu Wang, and Kiyoharu Aizawa. The lottery ticket hypothesis in denoising: Towards semantic-driven initialization. In *European Conference on Computer Vision (ECCV)*, 2024.

[42] Donghoon Ahn, Juyeon Kang, Seunghyun Lee, Jaeho Min, Minseok Kim, Wonseok Jang, Hyunsu Cho, Saurabh Paul, Sungnyun Kim, Eunho Cha, et al. A noise is worth diffusion guidance. *arXiv preprint arXiv:2412.03895*, 2024.

[43] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

.

A Related Works

Behavior cloning and Diffusion models Diffusion-based behavioral cloning has emerged as a strong class of policies for robotic control. Diffusion Policy demonstrated visuomotor policy learning via conditional action denoising [5]. Extensions have incorporated 3D representations [26], goal-masking for exploration [27], and transformer-based backbones [28]. Methods such as JUICER enable efficient long-horizon assembly from few demonstrations [29], while DP3 attains strong generalization across real-world tasks [26]. Diffusion-based policies have also been scaled to multi-task and generalist settings, including Octo [30], π_0 [31] and Gr00t N1 [25]. These works establish diffusion models as a scalable and expressive policy class, but do not address adaptation after pretraining.

RL-based adaptation of diffusion policies. Several approaches apply reinforcement learning to improve diffusion policies. DPPO fine-tunes diffusion models with PPO-style objectives [12]. IDQL combines diffusion actors with implicit Q-learning critics [13]. QSM matches denoiser scores with Q-gradients [15]. DIPO formulates diffusion policies as the policy class within standard actor-critic frameworks [32]. Recent efforts avoid weight updates, instead steering behavior via auxiliary policies or noise perturbations. RESIP adds a residual RL policy to refine pretrained actions [33], while DSRL optimizes over the diffusion noise space to adapt behavior with black-box access [23]. These methods highlight the potential of RL-guided adaptation, though most involve fine-tuning or auxiliary networks, unlike our lightweight steering approach. As we demonstrate empirically, altering diffusion weights lead to greater instability and is less sample-efficient compared to our method.

Value-guided or critic-guided diffusion. Beyond robotics, value or critic functions have been integrated into diffusion sampling. Q-score matching [15], energy-weighted diffusion [34, 35], and diffusion-based variational optimization [36] embed critic signals into denoising objectives. Other approaches use rejection sampling [37, 13], score regularization [38], or advantage-weighted classifiers [39] to bias samples toward higher-value actions. Analogous techniques exist in image generation, where classifier or latent noise optimization guides diffusion outputs [40, 41, 42]. In contrast, our method applies critic gradients directly to denoising targets at inference, enabling fine-grained, step-wise policy steering without retraining.

B VGD algorithm

Algorithm 2 Value-Guided Denoising

```
Require: state $s$; pretrained $\epsilon_\theta^{(\tau)}(\cdot, s)$; critic $Q_\phi$; guidance strength $\lambda \geq 0$
1: Sample initial latent $x_T \sim \mathcal{N}(0, I)$
2: for $\tau = T, T-1, \dots, 1$ do
3:     $\hat{x}_0^{(\tau)} \leftarrow \big(x_\tau - \sqrt{1-\alpha_\tau}\,\epsilon_\theta^{(\tau)}(x_\tau, s)\big)\big/\sqrt{\alpha_\tau}$        ▷ predicted $x_0$
4:     $g_\tau \leftarrow \nabla_a Q_\phi(s, a)\big|_{a=\hat{x}_0^{(\tau)}}$        ▷ autodiff; no gradients through $\epsilon_\theta$
5:     $\hat{x}_0'^{(\tau)} \leftarrow \hat{x}_0^{(\tau)} + \lambda\, g_\tau$
6:     $x_{\tau-1} \leftarrow \sqrt{\alpha_{\tau-1}}\,\hat{x}_0'^{(\tau)} + \sqrt{1-\alpha_{\tau-1}}\,\epsilon_\theta^{(\tau)}(x_\tau, s)$
7: end for
8: return $a \leftarrow x_0$ =0
```

C Experimental details

Code is available at https://anonymous.4open.science/r/VGD.

We evaluate VGD on four ROBOMIMIC: Can, Lift, Square, and Transport using frozen diffusion policies and training only a state–action critic online. Data for the methods we compare to (eg. DPPO, DIPO) are taken from [23]. Otherwise, VGD experiments were run on a Nvidia Geforce RTX 5090 GPU, with each run taking ≈ 12 hours. Environments are vectorized. Observations are low-dimensional, comprised of proprioceptive and object states. The frozen diffusion policy conditions on each observation step, and executes actions in chunks (see Table 1 for sizes).

Base policies. For each task we load a pretrained diffusion checkpoint and keep all policy weights frozen. For `Robomimic Square` and `Transport`, we use the diffusion policy checkpoints from Ren et. al [12] as our pretrained model. For the more challenging tasks `Can` and `Square`, we use the diffusion policies from Wagenmaker et. al [23], which fine-tune upon the Ren et. al policies to provide stronger initial learning signals for our RL experiments. Decoding uses DDIM with a fixed number of denoising steps depending on task (see Table 1). Additionally, to stabilise learning, we clip each component of predicted clean actions to 1.0.

VGD decoding. At each denoising step we form the DDIM predicted clean action $\hat{\boldsymbol{x}}_0^{(\tau)}$, nudge it along the critic gradient by a step-dependent coefficient λ, and substitute the modified target back into the update. We are motivated to set $\lambda = 0$ for the initial few denoising steps, when $\hat{\boldsymbol{x}}_0^{(\tau)}$ is still noisy and contain little information about the eventual output of the diffusion process. Empirically, for ROBOMIMIC tasks with 8 to 10 DDIM steps, we find that setting $\lambda = 0$ for the first 5 and 7 steps respectively reduces compute without changing performance. Thus, we use this setting for the experiments. Additionally, we anneal λ during the very start of training to stabilize learning. In particular, we increase λ from 0 to its full value over a set number of warmup steps (see Table 1). Note that they are insignificant in proportion to the total number of environment steps. However, later experimentation revealed that they have no impact on performance.

Critic and RL loop. We train only the critic (double-Q with `n_critics=2`, `min` backup) via off-policy TD with Polyak averaging ($\tau = 0.005$) to a target critic. This is implemented via the algorithms provided in STABLE BASELINES 3 [43]. Before training begins, we first run the frozen diffusion policy for a set number of steps to initialize the replay buffer with rollouts. In all cases, we use a sparse 0/1 reward: a positive reward is given only at steps where the robot completes the given task. Parts of this training code is adapted from [23].

Task	Action chunk size	UTD	γ	λ	warmup (updates)	DDIM steps	initial rollout
Can	4	20	0.99	0.01	0	8	1,501
Lift	4	30	0.99	0.005	50,000	8	1,501
Square	4	20	0.999	0.005	80,000	8	2,001
Transport	8	20	0.99	0.0008	100,000	10	20,001

Table 1: Per-task hyperparameters for ROBOMIMIC tasks.

Hyperparameter	Value
Optimizer	Adam
Learning rate	3×10^{-4}
Number of environments	4
Batch size	512 or 1024 (per task)
Replay buffer size	10,000,000 transitions
Critic MLP	3 layers $\times$ 2048 units, Tanh activations
Target network smoothing τ	0.005
Q critics	2

Table 2: Shared training hyperparameters used across VGD experiments.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction explained that diffusion policies face obstacles, what VGD is, and how VGD helps overcome these obstacles by improving upon previous methods. Then, the paper explains VGD in detail, then shows that it alleviates the aforementioned obstacles through experiments in Robomimic.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss limitations towards the end of the paper, including: higher compute cost at inference and the need for more empricial experiments.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

   Justification: The paper does not include theoretical results. The preliminaries have been checked.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have included all information necessary for reproduction, including experimental setups. Importantly, we have also included our code in a Github repository (which contains instructions for installation). We have also included pseudocode.

   Guidelines:

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

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

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

   Answer: [Yes]

   Justification: We have released the code (see abstract); the Github repository contains the exact commands and environments needed to run the code.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We have presented all experimental setup details in the Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We have given error bands for our experimental results, which represent the 95% confidence interval. The experiments were averaged over 4 runs each.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Yes, we have provided the compute information in the Appendix: one RTX 5090, running for around 15 hours per run.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the Code of Ethics and confirm that all codes are satisfied.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: Our work contributes to the development of robotics models and test them in virtual environments, which do not have a direct societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have cited and acknowledged the papers and authors that our code has been adapted from.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: Our code release (the Github repository) contains all necessary information. Please let us know if any details are missing.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: The work does not involve human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.