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.

Keywords: Reinforcement learning, diffusion models, robotic manipulation

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 the 9th Conference on Robot Learning (CoRL 2025) Workshop RembeRL. Do not distribute.

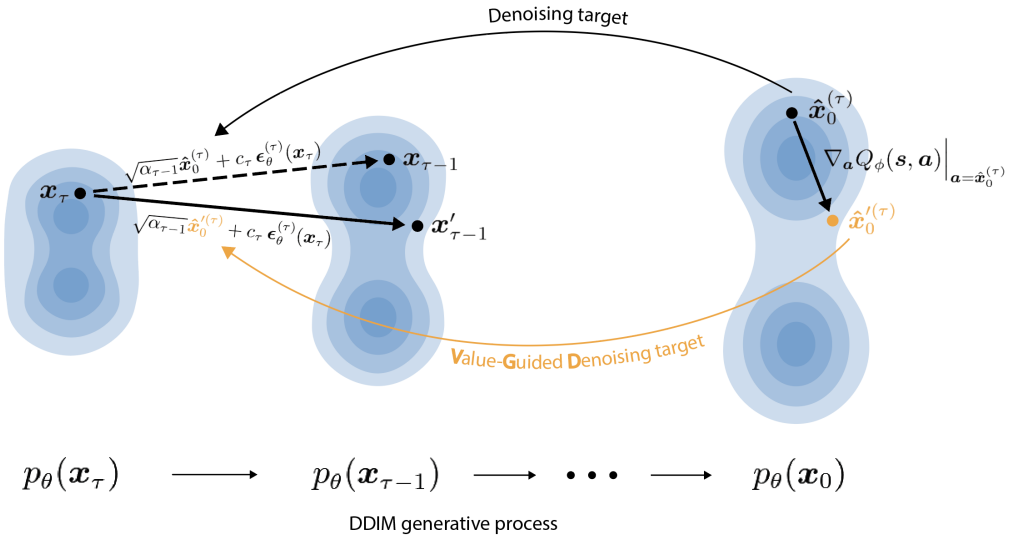

Figure 1: Illustration of our approach, *Value-Guided Denoising* (VGD). In a standard application of a diffusion-based BC policy, we sample an initial noise latent $\boldsymbol{x}_T$, then successively denoise it through the DDIM sampling process. At each denoising step t, the standard DDIM decoding 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

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

[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].

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 $\boldsymbol{x}_\tau$.

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

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

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:

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

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

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

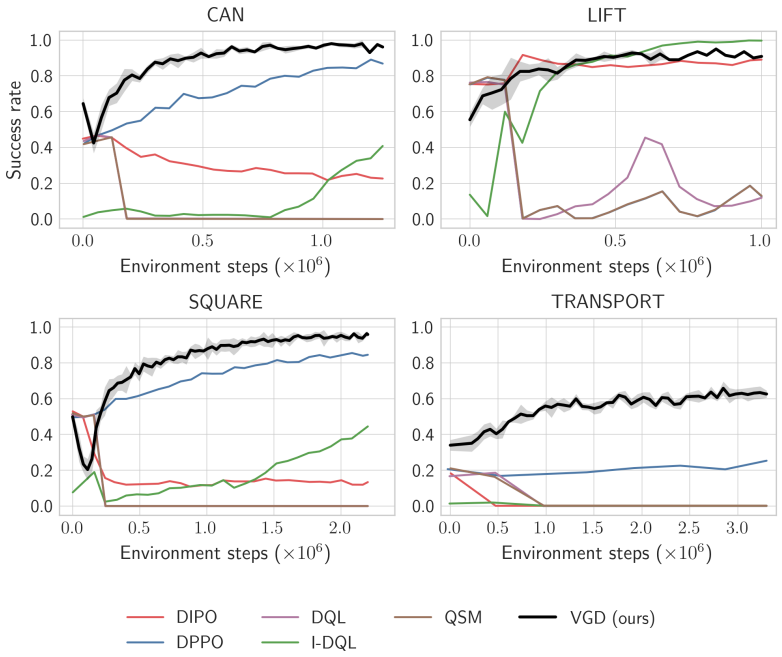

Figure 2: On ROBOMIMIC [21] benchmarks, VGD achieves sample-efficient adaptation of diffusion-based policies using online data.

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]

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

[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.

[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

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.

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.

6 Limitations

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 $\hat{\boldsymbol{x}}_0^{(\tau)}$.

In the paper, we validated our algorithm on only one benchmark (ROBOMIMIC) due to compute constraints. A greater range of empirical experiments would demonstrate the broader applicability of our method. For instance, applying VGD with high-dimensional image input instead of low-dimensional state input. We expect to address these limitations in upcoming work.

References

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, and et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. 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] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. 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] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, and et al. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. C. M. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, Daegu, Republic of Korea, July 2023.

[6] K. Black, M. Nakamoto, P. Atreya, H. R. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024.

[7] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. 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. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

[8] X. Yuan, T. Mu, S. Tao, Y. Fang, M. Zhang, and H. Su. Policy decorator: Model-agnostic online refinement for large policy model. *arXiv preprint arXiv:2412.13630*, 2024.

[9] S. Ross, G. J. Gordon, and J. A. 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] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

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

[12] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz. Diffusion policy policy optimization. In *Proceedings of Robotics: Science and Systems 2024 (RSS)*, 2024.

[13] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. Grudzien Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies, 2023.

[14] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[15] M. Psenka, A. Escontrela, P. Abbeel, and Y. Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[16] H. T. Suh, G. Chou, H. Dai, L. B. Yang, A. Gupta, and R. Tedrake. Fighting uncertainty with gradients: Offline reinforcement learning via diffusion score matching. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[17] L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

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

[19] M. S. Mark, T. Gao, G. G. Sampaio, M. K. Srirama, A. Sharma, C. Finn, and A. Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

[20] S. Park, Q. Li, and S. Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

[21] A. Mandlekar, S. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, A. Gupta, P. Sundaresan, E. Orbay, H. Walke, S. Tian, L. Wang, K. Hsu, B. Thananjeyan, Y. L. Jiang, J. Gu, A. Jain, C. Finn, K. Goldberg, and S. Yu. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

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

[23] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

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

[25] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. an, Y. Fang, D. Fox, F. Hu, S. Huang, J. Jang, Z. Jiang, J. Kautz, K. Kundalia, L. Lao, Z. Li, Z. Lin, G. Liu, E. Llontop, L. Magne, A. Mandlekar, A. Narayan, S. Nasiriany, S. Reed, Y. L. Tan, G. Wang, J. Wang, Q. Wang, J. Xiang, Y. Xie, Y. Xu, S. Ye, Z. Yu, A. Zhang, H. Zhang, Y. Zhao, R. Zheng, and Y. Zhu. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[26] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

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

[28] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

[29] L. Ankile, A. Simeonov, I. Shenfeld, and P. 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, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[31] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. : A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[32] L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

[33] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024.

[34] C. Lu, H. Chen, J. Chen, H. Su, C. Li, and J. 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] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023.

[36] S. Ding, K. Hu, Z. Zhang, K. Ren, W. Zhang, J. Yu, J. Wang, and Y. Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *arXiv preprint arXiv:2405.16173*, 2024.

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

[38] H. Chen, C. Lu, Z. Wang, H. Su, and J. Zhu. Score regularized policy optimization through diffusion behavior. *arXiv preprint arXiv:2310.07297*, 2023.

[39] L. He, L. Shen, J. Tan, and X. Wang. Aligniql: Policy alignment in implicit q-learning through constrained optimization. *arXiv preprint arXiv:2405.18187*, 2024.

[40] L. Eyring, S. Karthik, K. Roth, A. Dosovitskiy, and Z. Akata. Reno: Enhancing one-step text-to-image models through reward-based noise optimization. *Advances in Neural Information Processing Systems*, 37, 2024.

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

[42] D. Ahn, J. Kang, S. Lee, J. Min, M. Kim, W. Jang, H. Cho, S. Paul, S. Kim, E. Cha, et al. A noise is worth diffusion guidance. *arXiv preprint arXiv:2412.03895*, 2024.

[43] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

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 $\boldsymbol{s}$; pretrained $\boldsymbol{\epsilon}_\theta^{(\tau)}(\cdot, \boldsymbol{s})$; critic Q_ϕ; guidance strength $\lambda \geq 0$

Sample initial latent $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

for $\tau = T, T-1, \ldots, 1$ **do**

  $\hat{\boldsymbol{x}}_0^{(\tau)} \leftarrow \big(\boldsymbol{x}_\tau - \sqrt{1-\alpha_\tau}\,\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau, \boldsymbol{s})\big)\big/\sqrt{\alpha_\tau}$ ▷ predicted x_0

  $\boldsymbol{g}_\tau \leftarrow \nabla_{\boldsymbol{a}} Q_\phi(\boldsymbol{s}, \boldsymbol{a})\big|_{\boldsymbol{a}=\hat{\boldsymbol{x}}_0^{(\tau)}}$ ▷ autodiff; no gradients through $\boldsymbol{\epsilon}_\theta$

  $\hat{\boldsymbol{x}}_0'^{(\tau)} \leftarrow \hat{\boldsymbol{x}}_0^{(\tau)} + \lambda\, \boldsymbol{g}_\tau$

  $\boldsymbol{x}_{\tau-1} \leftarrow \sqrt{\alpha_{\tau-1}}\,\hat{\boldsymbol{x}}_0'^{(\tau)} + \sqrt{1-\alpha_{\tau-1}}\,\boldsymbol{\epsilon}_\theta^{(\tau)}(\boldsymbol{x}_\tau, \boldsymbol{s})$

end for

return $\boldsymbol{a} \leftarrow \boldsymbol{x}_0$ =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 $\approx$ 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.