DPDP: DIFFUSION POLICY–STEERED DIFFUSION PLANNER FOR OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In offline reinforcement learning, diffusion models fill two roles with complementary strengths. Typically guided by a critic, a diffusion policy is value-aware and favors high-return, behavior-supported actions, but it is short-sighted: it never lays out the states it intends to visit. A diffusion planner is far-sighted, synthesizing such a long-horizon trajectory, but it is only behavior-driven at execution, realizing each planned transition using only offline behavior. Each is strong exactly where the other is weak. We introduce DPDP to bring them together at the planner's execution step, where a planned transition becomes an action. Reading the planner's inverse-dynamics executor as a posterior over actions: the environment's transition likelihood relates an action to the planned next state, while the behavioral prior determines its weight under the offline data. Replacing this prior with a value-optimized diffusion policy preserves the planned transition likelihood and favors valuable actions that can serve the plan. We target the resulting clean-action distribution in two ways. A training-free sampler composes policy and inverse-dynamics scores at matched action noise levels and applies Langevin correction; this route uses a shared action noise schedule and an approximately common terminal prior. A preference objective instead amortizes the transfer into a single diffusion executor. Across D4RL, DPDP improves the planner it builds on, staying competitive with reactive diffusion policies on locomotion and leading on long-horizon navigation.

1 INTRODUCTION

At every step, an agent faces two coupled questions: where should it go, and what should it do now to get there? The first calls for a plausible view of the future; the second requires choosing an action whose consequences justify taking it. This distinction loosely echoes deliberation and immediate choice in Kahneman et al. (2011)'s account of cognition, and a competent decision needs both at once. In offline reinforcement learning (RL) (Levine et al., 2020), diffusion models answer these questions separately, as policies or as planners: each answers one well, and neither answers both.

A diffusion policy is the immediate system. It models an expressive, state-conditioned action distribution $\pi_\theta(a_t \mid s_t)$, and a learned critic steers it toward behavior-supported actions with high estimated downstream return (Wang et al., 2022; Hansen-Estruch et al., 2023): it answers *what to do now*. Yet an action value says nothing about the intermediate states a distant goal must pass through. A diffusion planner is the deliberative system. It generates a future state or state–action trajectory (Janner et al., 2022; Ajay et al., 2023), avoiding the compounding error of one-step rollouts and exposing an explicit path to the goal: it answers *where to go*. To favor promising plans, it can condition generation on a target return, drawing trajectories resembling those that achieved that return in the data. Yet its denoising objective does not perform Bellman backups, so it does not explicitly rank actions that could realize the same planned next state by downstream value. Each family is thus strong exactly where the other is weak, holding one half of a competent decision. Merging the halves is not a matter of mixing outputs. The two models are defined over different variables, namely actions at the current state versus joint distributions over future trajectories, so the critic's action-level value has no natural port into the planner. A bridge must carry that value into the place the two decisions meet, the step where a plan becomes an action, rather than into a policy standing beside the planner.

This step offers the port we need to close the gap. To execute its plan, a planner turns the first planned transition into an action with an inverse-dynamics model, which is itself a distribution over the current

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

action: a posterior. Its likelihood is the planned transition, the deliberative target for where to go next and its prior is the offline behavior policy, the default action an agent would take before any value is weighed. Read this way, the immediate system belongs in the prior. A value-optimized diffusion policy is exactly that behavior prior improved by the critic, and the log-ratio between the two is a preference over actions, a measure of how much value optimization favors each. Transferring value is then a single Bayesian move: hold the likelihood fixed and replace the behavioral prior with the value policy, updating the posterior so the executed action keeps the planner's intended transition while tilting toward what the critic prefers. The action the agent takes becomes the posterior that fuses both, the deliberated plan entering through the likelihood and the immediate value through the prior, on one decision rather than through a scalar proxy. Because the executor and the value policy are diffusion models over the same action space, the transfer rests on two conditions we arrange by construction: a shared noise schedule and a shared terminal prior.

We name the resulting approach DPDP (Diffusion Policy-steered Diffusion Planner): an executor in which a diffusion policy and a diffusion planner act as one. We realize the same posterior in two ways. We realize it in two ways that target the same execution distribution: The first, DPDP-inf, is a training-free, inference-time algorithm composing the two models' action scores at matched noise levels and correcting the samples during sampling, with both models frozen. The second, DPDP-ft, finetunes existing inverse dynamics once, amortizing the transfer into a single executor with a preference objective, so deployment is a single reverse pass with no critic in the loop. On D4RL, both realizations improve the base planner in average return across locomotion, AntMaze, and Maze2D. They narrow its gap to the reactive policy on locomotion and exceed both base models in average return on AntMaze and Maze2D. These results show that value-informed action selection can improve the execution of a state plan without discarding its proposed transition.

Our contributions can be summarized as:

- We cast a planner's execution step as a Bayesian action posterior and recast value transfer as a prior replacement: with the planned transition held as the likelihood, we swap the behavioral prior for a value-optimized diffusion policy. Value-informed action selection thus enters the planner's own execution step. To the best of our knowledge, this work is the first method in which a diffusion policy and a diffusion planner jointly shape the executed action.
- We develop two realizations of the resulting target: DPDP-inf composes frozen diffusion scores with inference-time correction, while DPDP-ft amortizes the policy-induced preference through executor fine-tuning.
- We show that both realizations improve a diffusion planner across the D4RL suites studied and, on AntMaze and Maze2D, outperform both the planner and reactive policy bases in suite-average return.

2 Background and Related Work

Offline decision-making. Consider a discounted Markov decision process (MDP) defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ is the action space, $P(s' \mid s, a)$ is the transition kernel, $r(s, a)$ is the reward function on each transition, ρ_0 is the initial-state distribution, and $\gamma \in [0, 1)$ is the discount factor (Sutton et al., 1998; Haarnoja et al., 2018). A policy $\pi(a \mid s)$ seeks to maximize the expected discounted return $J(\pi) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. Offline reinforcement learning learns such a policy from a fixed dataset $\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}$, where each episode $\tau^{(n)} = (s_0, a_0, r_0, s_1, \ldots)$ consists of ordered transition tuples, without further interaction with the environment (Fujimoto et al., 2019; Levine et al., 2020; Fu et al., 2020). The learner must therefore represent potentially high-dimensional and multimodal behavior distributions while improving upon the behavior policy without exploiting actions unsupported by the data. Long-horizon tasks further require reasoning about delayed consequences and trajectory-level structure. These challenges motivate expressive models capable of representing complex action and trajectory distributions.

Diffusion models. Diffusion models have emerged as a powerful class of generative models capable of representing complex distributions through iterative denoising (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song & Ermon, 2019). Given a clean sample $\boldsymbol{x}^0 \sim \mathcal{P}$, a forward process corrupts it over K steps with the Gaussian kernel $q(\boldsymbol{x}^k \mid \boldsymbol{x}^0) = \mathcal{N}(\sqrt{\bar{\alpha}_k}\boldsymbol{x}^0, (1 - \bar{\alpha}_k)\mathbf{I})$ under a schedule $\{\bar{\alpha}_k\}_{k=1}^{K}$ with $\bar{\alpha}_k$ decreasing to ≈ 0, so that the terminal marginal converges to a data-independent

prior $\nu := \mathcal{N}(\mathbf{0}, \mathbf{I})$[1]. The resulting marginal is $p_k(\boldsymbol{x}^k) = \int q(\boldsymbol{x}^k \mid \boldsymbol{x}^0) p_0(\boldsymbol{x}^0)\, \mathrm{d}\boldsymbol{x}^0$, with $p_0 = \mathcal{P}$. If $\bar{\alpha}_K = 0$, its terminal marginal is exactly $\nu = \mathcal{N}(\mathbf{0}, \mathbf{I})$ for any $\mathcal{P}$. A noise-prediction network ϵ_θ is trained by denoising score matching,

$$\mathcal{L}^{(\text{diffusion})}(\theta) \;=\; \mathbb{E}_{\boldsymbol{x}^0 \sim \mathcal{P},\, k,\, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \big[\, \|\boldsymbol{\epsilon} - \epsilon_\theta(\boldsymbol{x}^k, k)\|_2^2 \,\big], \quad \boldsymbol{x}^k = \sqrt{\bar{\alpha}_k}\, \boldsymbol{x}^0 + \sqrt{1 - \bar{\alpha}_k}\, \boldsymbol{\epsilon}, \tag{1}$$

which recovers the score of the noised marginal through $\nabla_{\boldsymbol{x}^k} \log p_k(\boldsymbol{x}^k) \approx -\epsilon_\theta(\boldsymbol{x}^k, k)/\sqrt{1 - \bar{\alpha}_k}$ (Song et al., 2021). Sampling then starts from $\boldsymbol{x}^K \sim \nu$ and iteratively applies ϵ_θ to denoise back to a clean sample. A trained diffusion model is thus determined by two design choices: its noise schedule $\{\bar{\alpha}_k\}$ and its terminal prior ν. Two models that share both, over the same action representation, define scores on the same noisy action space at matched noise levels, even though their noised marginals $p_{I,k}, p_{Q,k}, p_{B,k}$ differ; this is what enables us to combine their scores level by level.

Current work applying diffusion models to offline RL mainly assigns two roles (Zhu et al., 2023): *i)* serving as **policies** to support complex multimodal-distribution modeling and *ii)* generating long-term trajectories and executing like **planners**.

Diffusion Policy models an expressive, potentially multimodal action distribution $\pi_\theta(a \mid s)$ (Wang et al., 2022; Chi et al., 2025) in place of the restricted actor distributions commonly used in offline RL (Kumar et al., 2020; Kostrikov et al., 2022; Fujimoto et al., 2019; Pearce et al., 2023). Starting from a diffusion behavior model trained on offline state-action pairs using Eq. 1, a learned critic can favor high-value actions by selecting or guiding samples (Chen et al., 2023; Hansen-Estruch et al., 2023; Lu et al., 2023), or by shaping diffusion policy training (Wang et al., 2022). Behavior modeling encourages actions supported by the offline data, while the Bellman-trained critic estimates their downstream returns. Thus, even when several actions are plausible under the data, the critic can favor one with higher estimated value. The policy does not, however, explicitly generate a route through intermediate states toward a distant goal or temporal constraint.

Diffusion Planner models the joint distribution of future states, or state–action trajectories, and generates a plan from which an action is executed or recovered through inverse dynamics (Janner et al., 2022; Ajay et al., 2023; Liang et al., 2023; Li et al., 2023; Chen et al., 2024a; Lu et al., 2025). When the current state and a goal are specified, holding them fixed during denoising casts trajectory generation as a structured inpainting problem; return conditioning or reward guidance can further favor desirable plans. Generating a trajectory jointly avoids the compounding rollout errors of one-step dynamics models and supports long-horizon reasoning in sparse-reward and goal-reaching tasks. Nevertheless, a high-return sample describes one possible continuation; its return does not, by itself, estimate the expected downstream value of executing its first action across possible outcomes. This motivates complementing trajectory-level guidance with a critic $Q(s_t, a_t)$ that evaluates candidate actions by their downstream consequences.

These complementary strengths motivate a policy that combines the planner's view of where the agent should go with the critic's assessment of what action it should take now. The diffusion planner supplies plausible, task-directed future trajectories, while the Q-guided diffusion policy favors behavior-supported actions with high estimated downstream return. What the literature still lacks is a way to carry the critic's action-level preference into the planner's execution step itself, rather than into a separate policy, and that is the gap we address.

3 Methodology

Combining a diffusion planner with a value-optimized diffusion policy hinges on the execution step, the point where a plan becomes an action. The planner proposes a next-state transition, and an inverse-dynamics executor chooses an action to realize it. Trained on offline data, that executor inherits the behavior policy's action preferences; a value-trained policy improves those preferences but says nothing about the proposed transition. We build an executor that respects the planned transition while favoring the actions the value-trained policy prefers.

We build this executor in three steps. First, we cast inverse dynamics as a conditional diffusion posterior and expose its state-conditioned null branch as a behavioral prior (Sec. 3.1). Next, we read that posterior through Bayes' rule and transfer value by replacing its behavioral prior with the value-optimized policy, an update we show is equivalently a preference optimization (Sec. 3.2).

[1]We use superscripts $^{1,\ldots,k}$ to index diffusion steps and subscripts $_{1,\ldots,t}$ to index environment steps.

Finally, we realize the updated posterior in two ways that mirror these two readings: by composing the models' scores and correcting samples at inference, or by amortizing the same preference into a single diffusion policy (Sec. 3.3).

3.1 Probabilistic Inverse Dynamics for Plan Execution

A state-trajectory planner proposes $\hat{\boldsymbol{\tau}}_t = (s_t, \hat{s}_{t+1}, \ldots, \hat{s}_{t+H})$. To execute the plan, an inverse dynamics model (IDM) must choose an action for its first proposed transition, from s_t to $g := \hat{s}_{t+1}$. State-based diffusion planners commonly use a deterministic inverse model $a_t = f_\phi(s_t, g)$, trained by action regression on offline transitions (Ajay et al., 2023; Liang et al., 2023).

A single predicted action need not represent the choices available for a transition. In offline RL, different actions can lead to similar next states because of redundant actuation, stochastic dynamics, or variation in the behavior that collected the data (Wang et al., 2022). When these choices form distinct modes, squared-error regression tends toward their conditional mean, which may itself be an unlikely action. We instead model the conditional action distribution with a diffusion inverse dynamics model.

To strengthen adherence to the planned transition, we apply classifier-free guidance (CFG) (Ho & Salimans, 2021) to the diffusion inverse dynamics model. For each offline transition (s_t, a_t, s_{t+1}), we train the denoiser $\epsilon_\phi(a_t^k, k, s_t, c)$ using Eq. 1. The conditioning variable c here is s_{t+1} for a fraction of training examples and $\varnothing$ otherwise by CFG dropout, where the dropout is independent of the transition. We always retain s_t, which specifies the current control context. Thus the same denoiser learns two branches: a transition-conditioned inverse dynamics model and a state-conditioned null model. We denote their population targets by

$$p_I(a \mid s, g) = p_{\mathcal{D}}(a \mid s, s' = g), \quad p_B(a \mid s) = p_{\mathcal{D}}(a \mid s). \tag{2}$$

The learned conditional and null branches approximate p_I and p_B, respectively. In particular, the null branch represents behavior at the current state rather than the global action marginal; when trained on the same offline data, it represents a behavior-cloning diffusion policy. This gives a behavioral reference p_B against which another policy $\pi(a \mid s)$ trained on the same data can be compared.

At inference, the executor draws its action from the transition-conditioned branch $p_I(a \mid s, g)$ to realize the planner's proposed transition, while the null branch supplies the behavioral reference $p_B(a \mid s)$.

3.2 Value-Guided Execution of Planned Transitions

The planner proposes a next state g, and the executor must choose an action that can reach it from s. The transition-conditioned distribution $p_I(a \mid s, g)$ and its null counterpart $p_B(a \mid s)$ are, respectively, the conditional and marginal action distributions induced by the offline transitions. For a target g with nonzero behavior-induced transition density, Bayes' rule expresses p_I as the posterior over behavior actions conditioned on the transition from s to g:

$$p_I(a \mid s, g) = \frac{P(g \mid s, a)\, p_B(a \mid s)}{P(g \mid s)}, \qquad P(g \mid s) = \int P(g \mid s, a') p_B(a' \mid s)\, da'. \tag{3}$$

Here, $P(g \mid s, a)$ is the environment's transition likelihood measuring how likely action a is to produce the proposed next state. Although we cannot evaluate it directly, Eq. 3 identifies its action-dependent part:

$$P(g \mid s, a) = P(g \mid s) \frac{p_I(a \mid s, g)}{p_B(a \mid s)} \propto_a \frac{p_I(a \mid s, g)}{p_B(a \mid s)} \approx \frac{p_\phi(a \mid s, g)}{p_\phi(a \mid s, \varnothing)}, \tag{4}$$

The conditional and null branches of the same diffusion inverse dynamics model represent the two distributions in this ratio (Sec. 3.1).

Learning by behavioral cloning only the offline data, the planner executes its proposed transition by sampling from

$$p_I(a \mid s, g) \propto_a P(g \mid s, a)\, p_B(a \mid s). \tag{5}$$

The likelihood favors actions that reach g, while the behavior prior weights them according to their frequency in the offline data. However, if several actions can plausibly realize the planned transition, their relative preference is not explicitly determined by critic-estimated value. On the other hand, a

critic-trained diffusion policy $p_Q(a \mid s)$ parameterized by ψ provides such a preference: its training favors actions with higher learned Q-values while retaining a behavior-cloning objective (Wang et al., 2022).

Because p_Q and p_B are both fit to the same offline data under the same behavior-cloning objective, with Diffusion-QL adding only a value term on top of it, they empirically have approximately the same action support. This makes the two priors directly comparable, and we can replace the behavior prior in Eq. 5 with p_Q while retaining the transition likelihood (Neiswanger & Xing, 2017). The resulting executor is

$$p^*(a \mid s, g) \propto_a P(g \mid s, a)\, p_Q(a \mid s) = p_I(a \mid s, g) \frac{p_Q(a \mid s)}{p_B(a \mid s)}. \tag{6}$$

It remains conditioned on the planned next state through the transition likelihood, while the critic-trained policy replaces behavior frequencies as the action prior. Thus, p_1^* depends on g even though p_Q does not.

In terms of the learned models, its clean-action target is represented by

$$p^*_{\phi,\psi}(a \mid s, g) \propto_a p_\phi(a \mid s, g)\, \frac{p_\psi(a \mid s)}{p_\phi(a \mid s, \varnothing)}, \tag{7}$$

The conditional and null IDM branches represent p_I and p_B, while p_ψ represents p_Q. This identity holds at the clean-action level. It does not imply that combining the denoisers' scores at intermediate noise levels yields the score of the forward-noised target. We elaborate this further in Sec. 3.3.

Eq. 6 also admits a preference interpretation. At a fixed state, the log-density ratio

$$r^*(s, a) = \log \frac{p_Q(a \mid s)}{p_B(a \mid s)} + C(s) \tag{8}$$

measures how much more strongly p_Q favors a than p_B does with the action-independent term $C(s)$ canceling in comparisons. It describes a preference induced by the shift between action distributions, rather than the critic value $Q(s, a)$ itself.

Although the diffusion models do not provide tractable densities for evaluating this ratio directly, its pairwise preference can be elicited from samples. At a fixed state, draw $a^+ \sim p_Q(\cdot \mid s)$ and $a^- \sim p_B(\cdot \mid s)$. For two observed actions a and a', the odds of assigning a to p_Q and a' to p_B, rather than the reverse, are

$$\frac{p_Q(a \mid s)\, p_B(a' \mid s)}{p_Q(a' \mid s)\, p_B(a \mid s)} = \exp\big[r^*(s, a) - r^*(s, a')\big]. \tag{9}$$

Normalizing these odds gives the Bradley–Terry preference model $P_r(a \succ a' \mid s) = \sigma(r^*(s, a) - r^*(s, a'))$ (Bradley & Terry, 1952; Rafailov et al., 2023). Here, the "winner" label identifies the action drawn from p_Q; it does not imply that every sampled a^+ has a larger $Q(s, a)$ than its paired a^-. Appendix A.1 gives the derivation.

The same preference characterizes the tempered executor. For each (s, g), we favor actions with high r^* while anchoring the execution distribution to the planner's posterior:

$$\max_{\rho(\cdot \mid s,g)} \quad \mathbb{E}_{a \sim \rho(\cdot \mid s,g)}[r^*(s, a)] - \beta D_{\mathrm{KL}}(\rho(\cdot \mid s, g) \,\|\, p_I(\cdot \mid s, g)). \tag{10}$$

For $\beta > 0$ and a finite normalizer, its unique optimum is

$$p^*_\beta(a \mid s, g) \propto_a p_I(a \mid s, g) \exp\left(\frac{r^*(s, a)}{\beta}\right) \propto_a p_I(a \mid s, g) \left(\frac{p_Q(a \mid s)}{p_B(a \mid s)}\right)^{1/\beta}, \tag{11}$$

At $\beta = 1$, this preference optimum is exactly the full prior replacement in Eq. 6. More generally, the second line has the same Bayesian form with a composed action prior $p_\beta(a \mid s) \propto_a p_B(a \mid s)^{1-1/\beta} p_Q(a \mid s)^{1/\beta}$. For $\beta \geq 1$, this prior interpolates between p_Q at $\beta = 1$ and p_B as $\beta \to \infty$. Thus, the preference objective extends the full replacement by controlling how strongly the value policy changes the executor's prior. Appendix A.2 proves the optimum.

3.3 Realizing the Unified Execution Plan with Diffusion Models

Section 3.2 defines the full prior replacement in Eq. 6 and its extension in Eq. 11. The clean-action composition in Eq. 7 motivates combining pretrained diffusion models at inference. The

KL-regularized preference formulation in Eq. 10 motivates amortizing the update into a single executor reverse chain. We develop both realizations below.

Composed score path. The inverse dynamics model and Diffusion-QL policy do not provide tractable clean-action densities. Since forward noising does not commute with a product of densities, combining their scores at level k generally does not give the score of $q_k * p^*_\beta$ (Du et al., 2023a). We instead compose their noised marginals. Using a shared action representation and forward kernel q_k, let $p_{j,k} = q_k * p_j$ for $j \in \{I, Q, B\}$ and define

$$\tilde{p}_k(a^k \mid s_t, g) \propto p_{I,k}(a^k \mid s_t, g) \left(\frac{p_{Q,k}(a^k \mid s_t)}{p_{B,k}(a^k \mid s_t)} \right)^{1/\beta}. \tag{12}$$

For exact component scores, this product has the combined noise prediction

$$\hat{\epsilon}^*(a^k_t, k) = \epsilon_\phi(a^k_t, k, s_t, g) + \frac{1}{\beta}\Big[\epsilon_\psi(a^k_t, k, s_t) - \epsilon_\phi(a^k_t, k, s_t, \varnothing)\Big], \tag{13}$$

which adds to the transition-conditioned inverse-dynamics prediction the Q-versus-behavior score difference, other arguments suppressed. At the clean endpoint, Eq. 3 gives

$$\tilde{p}_0(a \mid s, g) = p^*_\beta(a \mid s, g) \propto_a P(g \mid s, a)\, p_Q(a \mid s)^{1/\beta} p_B(a \mid s)^{1-1/\beta}. \tag{14}$$

Proposition 3.1 (Composed annealing path)**.** *Suppose the component denoisers give the exact scores of $p_{I,k}$, $p_{Q,k}$, and $p_{B,k}$; the products in Eq. 12 are normalizable; and $p_{I,K} = p_{Q,K} = p_{B,K} = \nu$. Then Eq. 13 represents $\nabla \log \tilde{p}_k$, with $\tilde{p}_K = \nu$ and $\tilde{p}_0 = p^*_\beta$. In general, however, $\tilde{p}_k \neq q_k * \tilde{p}_0$ at intermediate levels. Thus, the combined field does not generally provide the forward-marginal scores required by an ordinary reverse diffusion chain.*

Appendix A.4 proves the proposition and gives a Gaussian case in which the intermediate marginals do agree. Sharing q_k does not establish that it scores $q_k * p_Q$. In practice, we use it as an estimate, and $\bar{\alpha}_K \approx 0$ makes the common terminal prior approximate. The clean-endpoint identity remains algebraic.

Training-free sampling. We initialize $a^K_t \sim \nu$ and use the combined prediction $\hat{\epsilon}^*$ for each reverse step. Because the intermediate $\tilde{p}_k$ are generally not the forward marginals of $\tilde{p}_0$, an ordinary reverse chain need not preserve the composed path. After stepping from k to $j = k - 1$, we therefore apply M Langevin corrections at each reached level $j > 0$:

$$a^j_t \leftarrow a^j_t - \eta_j \frac{\hat{\epsilon}^*(a^j_t, j)}{\sqrt{1 - \bar{\alpha}_j}} + \sqrt{2\eta_j}\, z, \qquad z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{15}$$

The drift is $\eta_j \nabla \log \tilde{p}_j$, so each correction moves the sample toward the composed marginal before the next reverse step. An ideal annealed sampler with exact, fully mixed transitions at every level, including the clean endpoint, would sample $\tilde{p}_0$ (Du et al., 2023a; Liu et al., 2022). Our finite-step sampler corrects only at positive noise levels and finishes with a denoising step, so its output approximates that target; unadjusted Langevin also has step-size bias at fixed η_j. Each correction evaluates both IDM branches and the Diffusion-QL denoiser.

Amortized preference alignment. The second realization trains a conditional diffusion policy $\pi_\theta(a \mid s, g)$ toward the same clean target, so deployment requires only its reverse chain. We initialize π_θ from the transition-conditioned IDM and keep that IDM frozen as the reference p_I.

Section 3.2 identifies the preference $r^*(s, a) = \log[p_Q(a \mid s)/p_B(a \mid s)] + C(s)$. With p_I as the reference, the policy's implicit preference is

$$R_\theta(a; s, g) = \beta \log \frac{\pi_\theta(a \mid s, g)}{p_I(a \mid s, g)}. \tag{16}$$

Setting $R_\theta(a; s, g) = r^*(s, a) + C(s, g)$ and solving for π_θ,

$$\pi_\theta(a \mid s, g) \propto_a p_I(a \mid s, g) \left(\frac{p_Q(a \mid s)}{p_B(a \mid s)} \right)^{1/\beta} = p^*_\beta(a \mid s, g) \quad \text{after normalization.} \tag{17}$$

The Q-versus-behavior preference thus supplies the value that shifts the executor from p_I toward p^*_β.

We obtain preference pairs at a fixed state by drawing $a^+ \sim p_Q(\cdot \mid s_t)$ and $a^- \sim p_B(\cdot \mid s_t)$, as in Sec. 3.2. These are distributional labels, with no sample-wise guarantee that $Q(s_t, a^+) > Q(s_t, a^-)$.

For training, the pair shares (s_t, g) and a noise level $k \sim \mathcal{U}\{1, \dots, K\}$, while its actions are corrupted independently: $a_{\pm}^k = \sqrt{\bar{\alpha}_k}\, a^{\pm} + \sqrt{1-\bar{\alpha}_k}\, \epsilon^{\pm}$, with $\epsilon^+, \epsilon^- \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Clean-action likelihood ratios in Eq. 16 are intractable for these diffusion models. Following diffusion preference optimization (Wallace et al., 2024; Li et al., 2026), we use differences of denoising errors as per-level surrogates. For either member (a, ϵ) of the pair, let $a^k = \sqrt{\bar{\alpha}_k} a + \sqrt{1-\bar{\alpha}_k}\epsilon$ and define

$$D_\theta(a, \epsilon) = \|\epsilon - \epsilon_\theta(a^k, k, s_t, g)\|^2 - \|\epsilon - \epsilon_\phi(a^k, k, s_t, g)\|^2. \tag{18}$$

D_θ compares the trainable policy with its frozen reference and is the per-level surrogate for the log-ratio in Eq. 16. We minimize

$$\mathcal{L}_{\text{align}}(\theta) = -\mathbb{E}\left[\log \sigma\left(-\frac{\beta}{2}\left[D_\theta(a^+, \epsilon^+) - D_\theta(a^-, \epsilon^-)\right]\right)\right]. \tag{19}$$

With exact clean-action likelihood ratios and a realizable policy, the corresponding pairwise objective has Eq. 17 as its population optimum; the derivation appears in Appendix A.3. Equation 19 replaces those ratios with a single-timestep denoising surrogate, so it targets that optimum without guaranteeing exact recovery. The uniform timestep weighting shown here makes β an effective preference-strength parameter; diffusion-ELBO coefficients would give a differently weighted loss. All reported runs minimize this loss together with a behavior-cloning anchor to the reference p_I, so the trained objective is

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{align}}(\theta) + \lambda\, \mathcal{L}_{\text{BC}}(\theta), \qquad \mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{(s_t, a, g),\, k, \epsilon}\left\|\epsilon - \epsilon_\theta(a^k, k, s_t, g)\right\|^2, \tag{20}$$

with $\mathcal{L}_{\text{BC}}$ the goal-conditioned denoising loss that trained the executor and anchor weight $\lambda = 1$; at $\lambda = 0$ it reduces to Eq. 19. At deployment, only π_θ is used, without Langevin corrections or calls to the frozen models.

We provide the algorithmic description of both implementations in Appendix B.

4 Experiments

Dataset. We evaluate on the D4RL Gym-MuJoCo locomotion, AntMaze, and Maze2D datasets (Fu et al., 2020). We report D4RL-normalized returns from closed-loop rollouts of each method's policy, averaged over five random seeds. For every seed we roll out 50 parallel environments and run, per environment, 3 episodes on the dense-reward locomotion tasks and 20 on the sparse-reward AntMaze and Maze2D suites. Each locomotion score therefore aggregates $50{\times}3{\times}5{=}750$ rollouts, and each AntMaze or Maze2D score $50{\times}20{\times}5{=}5000$ rollouts (1000 per seed); the larger budget on the binary-success navigation suites resolves per-seed success to 0.1% and makes the reported standard errors meaningful.

Baseline Methods. We compare against behavior cloning (BC) (Ross et al., 2011), Gaussian policies (CQL (Kumar et al., 2020), IQL (Kostrikov et al., 2022)), diffusion policies (DQL (Wang et al., 2022), SfBC (Chen et al., 2023), IDQL (Hansen-Estruch et al., 2023), QGPO (Lu et al., 2023), SRPO (Chen et al., 2024b)), and diffusion planners (Diffuser (Janner et al., 2022), AD (Liang et al., 2023), DD (Ajay et al., 2023), HD (Chen et al., 2024a), DV (Lu et al., 2025)). For these literature baselines, we use their best reported results. We show both DQL's original results and its CleanDiffuser replication (Dong et al., 2024), denoted DQL*. The latter generally performs better in our comparison and serves as the basis for our adapted DQL+.

Implementation Details. We build DPDP-ft and DPDP-inf on DQL and DV because their components supply the two signals our method combines. DQL learns a diffusion action policy shaped by a Q-function, providing value-informed action preferences; DV generates state plans and uses a diffusion inverse dynamics model to execute their transitions (Wang et al., 2022; Lu et al., 2025). Both are competitive baselines, and DV's diffusion executor gives us a direct way to condition action selection on the planner's proposed next state.

Score composition requires the policy and inverse dynamics model to use the same action diffusion process. We therefore implement DQL and DV's inverse dynamics with a shared 10-step variance-preserving DDPM schedule (Ho et al., 2020; Karras et al., 2022). Because this differs from the configurations underlying their published scores, we report our schedule-matched adaptations separately as DQL+ and DV+. For DV+, we train the inverse dynamics model with random next-state conditioning dropout, yielding the conditional and null branches used in Sec. 3.1; execution uses only the transition-conditioned branch. Finally, we rescale the shared schedule to give $\bar{\alpha}_K \approx 0$; the three forward marginals then approach the common terminal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Table 1: **Long-horizon navigation** on D4RL. Normalized return averaged over five seeds. The blue DQL+ (value policy) and DV+ (planner) are our reimplemented schedule-matched base models; the purple DPDP-inf and DPDP-ft are the two realizations of the transfer, each annotated with its gain over the base planner DV+. On both suites, the policy and the planner are individually comparable, and unifying them exceeds both on the suite average.

Dataset	Gaussian policies			Diffusion policies							Diffusion planners							
	BC	CQL	IQL	SfBC	DQL	DQL⋆	DQL+	IDQL	QGPO	SRPO	Diffuser	AD	DD	HD	DV	DV+	DPDP-inf	DPDP-ft
Antmaze-M-P	0.0	14.9	71.2	81.3	76.6	**87.3**	86.0	84.5	83.6	80.7	6.7	12.0	8.0	–	**89.0**	84.0	84.5 (+0.5)	84.8 (+0.8)
Antmaze-M-D	0.0	15.8	70.0	82.0	78.6	82.6	87.3	84.8	83.8	75.0	2.0	6.0	4.0	**88.7**	87.4	82.9	86.0 (+3.1)	83.5 (+0.6)
Antmaze-L-P	0.0	53.7	39.6	59.3	46.4	81.3	80.9	63.5	66.6	53.6	17.3	5.3	0.0	–	76.4	79.5	78.0 (-1.5)	**92.8** (+13.3)
Antmaze-L-D	0.0	61.2	47.5	45.5	56.6	70.6	71.2	67.9	64.8	53.6	27.3	8.7	0.0	**83.6**	80.0	78.3	81.0 (+2.7)	81.7 (+3.4)
Average	0.0	36.4	57.1	67.0	64.6	80.5	81.4	75.2	74.7	65.7	13.3	8.0	3.0	–	83.2	81.2	82.4 (+1.2)	**85.7** (+4.5)
Maze2D-U	3.8	5.7	47.4	73.9	–	140.6	137.6	57.9	–	–	113.9	135.1	–	128.4	136.6	129.2	142.3 (+13.1)	**145.7** (+16.5)
Maze2D-M	30.3	5.0	34.9	73.8	–	152.0	151.3	89.5	–	–	121.5	129.9	–	135.6	150.7	146.3	**155.5** (+9.2)	152.7 (+6.4)
Maze2D-L	5.0	12.5	58.6	74.4	–	186.8	185.8	90.1	–	–	123.0	167.9	–	155.8	203.6	203.1	**209.0** (+5.9)	203.7 (+0.6)
Average	13.0	7.7	47.0	74.0	–	159.8	158.2	79.2	–	–	119.5	144.3	–	139.9	163.6	159.5	**168.9** (+9.4)	167.4 (+7.8)

DPDP-inf combines the frozen DQL and inverse-dynamics denoisers at matched noise levels, applying M Langevin corrections after each reverse step to a positive noise level (M=1 by default). It requires no additional training. DPDP-ft instead fine-tunes the executor toward the same target using preference pairs: preferred actions come from the DQL policy, and dispreferred actions from the IDM's behavioral branch. The transition-conditioned IDM serves as the frozen reference, while the conditional-to-null adjustment retains the chosen level of planning guidance. We implement both methods in CleanDiffuser (Dong et al., 2024) with PyTorch (Paszke et al., 2019) and train on a single NVIDIA RTX 6000 Ada GPU. All diffusion models use Adam (Kingma & Ba, 2015) and a batch size of 128. We train the DV planner and DQL policy for 10^6 gradient steps each, and the IDM for 2×10^5 steps. DPDP-ft fine-tunes the executor for another 2×10^5 steps at the same batch size. We report results over five random seeds. Appendix C provides the full hyperparameters, including the value-transfer strength and corrector budget.

4.1 Experimental Results

Tables 1 and 2 report D4RL-normalized returns. The shaded DQL+ and DV+ columns denote our schedule-matched value policy and planner. Since our DPDP is still planning-based, we mark the signed differences for the proposed variants relative to DV+. Our reimplementations remain competitive with their reported counterparts and show no systematic performance degradation from schedule matching: some suite averages improve, while others decline modestly (e.g., DV+on Maze2D). We therefore evaluate value transfer against DV+ under the matched configuration.

On AntMaze and Maze2D (Table 1), the two bases perform similarly: DQL+ and DV+ score 81.4 and 81.2 on AntMaze, and 158.2 and 154.5 on Maze2D. Both proposed variants exceed both bases in suite-average return. DPDP-ft leads on AntMaze at 85.7 (+4.5 over DV+), with a particularly large gain on AntMaze-Large-Play (79.5→92.8). DPDP-inf leads on Maze2D at 168.9 (+9.4). The gains across both realizations support the benefit of combining planned transitions with critic-informed action selection, although the preferred realization differs by suite. Combining the planner's trajectory guidance with the reactive policy's value-informed action preferences yields higher average returns than either base model on AntMaze and Maze2D. Locomotion in Table 2 are dense-reward control tasks with no end goal to reach. Here DQL+ averages 88.8, ahead of DV+ at 83.4. Value transfer improves the planner to 84.5 with DPDP-inf and 84.3 with DPDP-ft, while the simple, reactive DQL+ retains the lead. DPDP-inf improves DV+ on every locomotion task, and DPDP-ft does so on the large majority. We observed that value-informed execution helps even where the reactive policy already performs better than the planner, but does not eliminate that gap.

Across both suites, the two variants improve the base planner on all but three of the thirty-two task entries, with two small regressions and one tie. They target the execution by implementing differently, DPDP-inf by inference-time score composition and DPDP-ft by amortized preference learning. Their returns track each other closely, so the choice between them is a deployment tradeoff: a single amortized pass with no critic at test time, or a training-free sampler that adds correction steps in place of a training stage. On end goal-oriented environments, DPDP-inf achieved better performance on Maze2D while DPDP-ft achieved better performance on AntMaze.

4.2 Ablation Studies

Q-reranking. A simple way to add value to the executor is to sample $\{a_1, \ldots, a_N\} \sim p_I(\cdot \mid s, g)$ and execute $a^\star = \arg\max_i Q(s, a_i)$. This tests whether our transfer improves on test-time critic

Table 2: **Gym-MuJoCo locomotion** on D4RL. Normalized return averaged over five seeds; columns as in Table 1. The blue DQL+ (value policy) and DV+ (planner) are our reimplemented schedule-matched base models. These are dense-reward, reactive tasks with no goal to reach: every step is scored on the spot, so they reward immediate control rather than foresight. The value policy DQL+ therefore leads and planning is unnecessary, yet the transfer still improves the base planner DV+ on nearly every task, narrowing the gap to the reactive frontier.

Dataset	Gaussian policies			Diffusion policies							Diffusion planners							
	BC	CQL	IQL	SfBC	DQL	DQL*	DQL+	IDQL	QGPO	SRPO	Diffuser	AD	DD	HD	DV	DV+	DPDP-inf	DPDP-ft
Walker2d-M	6.6	79.2	78.3	77.9	**87.0**	86.8	86.9	82.5	86.0	84.4	79.6	84.4	82.5	84.0	82.8	83.4	84.5 (+1.1)	85.0 (+1.6)
Walker2d-M-R	11.8	26.7	73.9	65.1	95.5	98.2	**98.4**	85.1	84.4	84.6	70.6	84.7	75.0	84.1	85.0	85.1	86.8 (+1.7)	86.1 (+1.0)
Walker2d-M-E	6.4	111.0	109.6	109.8	110.1	111.6	110.8	112.7	110.7	**114.0**	106.9	108.2	108.8	107.1	109.2	108.6	110.0 (+1.4)	110.7 (+2.1)
Hopper-M	29.0	58.0	66.3	57.1	90.5	96.5	97.0	65.4	98.0	95.5	74.3	96.6	79.3	**99.3**	83.6	83.2	84.3 (+1.1)	83.2 (+0.0)
Hopper-M-R	11.3	48.6	94.7	86.2	101.3	**101.6**	101.4	92.1	96.9	101.2	93.6	92.2	100.0	94.7	91.9	91.2	92.2 (+1.0)	91.9 (+0.7)
Hopper-M-E	111.9	98.7	91.5	108.6	111.1	111.1	108.7	108.6	108.0	100.1	103.3	111.6	111.8	**115.3**	110.0	109.8	110.7 (+0.9)	110.4 (+0.6)
HalfCheetah-M	36.1	44.4	47.4	45.9	51.1	52.3	52.4	51.0	54.1	**60.4**	42.8	44.2	49.1	46.7	50.4	50.8	50.9 (+0.1)	51.9 (+1.1)
HalfCheetah-M-R	38.4	46.2	44.2	37.1	47.8	47.9	48.0	45.9	47.6	**51.4**	37.7	38.3	39.3	38.1	45.8	46.0	46.8 (+0.8)	47.3 (+1.3)
HalfCheetah-M-E	35.8	62.4	86.7	92.6	**96.8**	95.5	96.0	95.9	93.5	92.2	88.9	89.6	90.6	92.5	92.7	92.7	94.6 (+1.9)	92.0 (-0.7)
Average	31.9	63.9	77.0	75.6	87.9	**89.1**	88.8	82.1	86.6	87.1	77.5	83.3	81.8	84.6	83.5	83.4	84.5 (+1.1)	84.3 (+0.9)

search over IDM proposals. Q-reranking improves locomotion performance but yields only modest gains on Maze2D, where both DPDP-ft and DPDP-inf outperform it on every layout (Appendix E.1). Its limitation is that value enters only after the candidates are generated. If a useful action has low probability under p_I, finding it may require many costly IDM samples and critic evaluations; if it lies outside the proposal support, reranking cannot find it at all. Searching more candidates can also amplify the notorious overestimation issue in the learned Q-function (Fujimoto et al., 2019). In short, reranking spends test-time compute to select from a fixed behavior-cloned proposal, whereas the transfer reshapes the executor's distribution itself.

Value strength. We sweep the value strength β for both realizations (Appendix E.2). On locomotion, where DQL+ already leads DV+, full value transfer (β=1) adds at most one or two points per task.

Langevin correction. We vary DPDP-inf's correction budget and measure inference latency alongside DPDP-ft (Appendix E.3). One correction step improves uncorrected composition on every suite. Small budgets provide the strongest gains; larger budgets reduce return on AntMaze while increasing latency. The amortized DPDP-ft remains the faster deployment option.

5 Discussion and Conclusion

Limitations. Because diffusion models do not compose exactly (Du et al., 2023a; Garipov et al., 2023), the two realizations also approximate their common clean-action target in different ways. For DPDP-inf, composing scores at each noise level does not generally produce the forward marginals of that target (Du et al., 2023a; Garipov et al., 2023), and finite Langevin correction does not remove all sampling error. Its theoretical characterization further assumes that the Q-trained denoiser scores the forward-noised distribution of its own generated actions, which Q-guided training does not guarantee. Repeated evaluations of the conditional IDM, null IDM, and DQL denoisers increase inference latency. For DPDP-ft, denoising-error differences substitute for intractable clean-action likelihood ratios, and preference fine-tuning adds a training stage.

Future Work. The transfer needs only a value-optimized action prior and a transition-conditioned executor that share a noise schedule, and value enters through the prior alone, which makes it easy to extend. A world-model planner whose inverse dynamics conditions on high-dimensional observations would carry value-aware execution to visual control (Hafner et al., 2020; Du et al., 2023b), recovering the transition likelihood from the executor's two branches exactly as here. Because value is confined to the prior, a sharper offline critic drops directly into that slot, turning progress on value estimation into better plan execution without changing the method. Finally, the value strength β is fixed per suite; learning a state-dependent β that trusts the critic more where its estimates are reliable (Wu et al., 2024) is a natural refinement of the single knob.

Conclusion. We presented a way to steer the execution of a diffusion planner by value-informed action preferences. The planner's proposed transition determines which actions are compatible with its plan, while a critic-trained policy changes their relative preference through a tempered prior substitution. We realize this target by composing frozen scores at inference with DPDP-inf or by training a conditional executor through preference alignment with DPDP-ft. On D4RL, both improve the base planner in average return across locomotion, AntMaze, and Maze2D; on the navigation suites, they exceed both base models on average. Returning to the question we opened with, the planner still answers *where to go*, and the executor uses value information from the policy to decide *what to do now to get there*.: an immediate choice made with deliberate foresight.

AI USE STATEMENT

We used generative AI tools only for language editing and polishing of author-written text. We have reviewed all AI-assisted work. All technical content, derivations, experiments, and analyses were produced by the authors.

ETHICS STATEMENT

This work studies offline reinforcement learning on standard, publicly available simulated benchmarks (Fu et al., 2020) and involves no human subjects, personally identifiable information, or private data. By reusing logged experience rather than collecting new interactions, offline methods such as ours can lower the cost and safety risk of online data gathering. At the same time, the executed behavior inherits the coverage limits and biases of the offline dataset and of the estimated critic; because the transfer deliberately steers execution toward the critic's preferences, a miscalibrated or biased value function could amplify undesirable actions. We therefore recommend auditing the underlying data and value estimates and validating the learned executor before any safety-critical or high-stakes deployment. We are not aware of ethical concerns specific to this work beyond those common to offline reinforcement learning.

REPRODUCIBILITY STATEMENT

We provide complete proofs of every claim in the appendices and full detail for reproduction. All experiments use the public benchmark and the open-source CleanDiffuser framework (Dong et al., 2024). Appendix C lists hyperparameters. The evaluation protocol is specified in Section 4, and per-task means with standard errors appear in Appendix D. Both realizations are given as explicit algorithms in Appendix B. To serve the machine learning community, we will release our code, configurations, and evaluation scripts together with detailed documentation and tutorials upon acceptance.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *ICLR*, 2023.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion. In *ICLR*, 2024a.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *ICLR*, 2023.

Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. In *ICLR*, 2024b.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Int. J. Robot. Res.*, 44(10-11):1684–1704, 2025.

Zibin Dong, Yifu Yuan, Jianye Hao, Fei Ni, Yi Ma, Pengyi Li, and Yan Zheng. CleanDiffuser: An easy-to-use modularized library for diffusion models in decision making. *NeurIPS*, 2024.

Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and MCMC. In *ICML*, 2023a.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *NeurIPS*, 2023b.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, 2019.

Timur Garipov, Sebastiaan De Peuter, Ge Yang, Vikas Garg, Samuel Kaski, and Tommi S. Jaakkola. Compositional sculpting of iterative generative processes. In *NeurIPS*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

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. *arXiv preprint arXiv:2304.10573*, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop on Deep Generative Models and Downstream Applications*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, 2022.

Daniel Kahneman, Patrick Egan, et al. *Thinking, fast and slow*, volume 1. Farrar, straus and Giroux New York, 2011.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *ICLR*, 2022.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-learning for offline reinforcement learning. In *NeurIPS*, 2020.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *ICML*, 2023.

Xiang Li, Yixuan Jia, Xiao Li, Jeffrey A Fessler, Rongrong Wang, and Qing Qu. MCLR: Improving conditional modeling via inter-class likelihood-ratio maximization and unifying classifier-free guidance with alignment objectives. In *ICML 2026 Workshop on Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning*, 2026.

Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. AdaptDiffuser: Diffusion models as adaptive self-evolving planners. In *ICML*, 2023.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models. In *ECCV*, 2022.

Cheng Lu, Huayu Chen, Jianfei Chen, Hang Su, Chongxuan Li, and Jun Zhu. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *ICML*, 2023.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? In *ICLR*, 2025.

Willie Neiswanger and Eric P. Xing. Post-inference prior swapping. In *ICML*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, David Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *ICLR*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *CVPR*, 2024.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Junkang Wu, Yuexiang Xie, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. β-DPO: Direct preference optimization with dynamic β. In *NeurIPS*, 2024.

Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Haoquan Guo, Tingting Chen, and Weinan Zhang. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*, 2023.