DADC: DUAL-SOURCE ANCHOR DRIFT CORRECTION FOR ONLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative policies offer expressive multimodal action representations for online reinforcement learning (RL). However, many existing methods focus on fitting actions favored by the critic, using negative feedback mainly to downweight less promising actions. This provides limited explicit guidance for avoiding low-value actions and can restrict further policy improvement. To address this limitation, we propose *Dual-Source Anchor Drift Correction (DADC)*, an online RL algorithm that combines attractive guidance with explicit repulsion from low-value policy samples. Such guidance requires informative anchors, which are difficult to construct without fixed expert data. DADC therefore constructs attractive and repulsive anchors from distinct sources. For each replay state, we sample initial attractive anchors from an advantage-weighted conditional flow prior trained on replay data and apply several steps of Q-guided Langevin refinement. At the same state, we select current-policy candidates with the lowest critic scores as repulsive anchors, focusing repulsion on relatively low-value actions. The actor update combines direct Q-value maximization with drift guidance toward attractive anchors and away from repulsive anchors. The flow prior and anchor construction are required only during training, while deployment uses a single actor forward pass. Experiments on 14 DMControl and HumanoidBench tasks show that DADC achieves the highest mean return among the evaluated methods on all tasks, delivering substantial gains over the compared generative policy methods.

1 INTRODUCTION

Deep reinforcement learning (RL) has achieved strong performance on complex continuous-control tasks (Schulman et al., 2017; Fujimoto et al., 2018; Heess et al., 2017). Such tasks often require coordinating many degrees of freedom and may admit multiple effective action strategies, motivating policies that can represent diverse behaviors. Generative policies based on diffusion models and flow matching further expand the policy class beyond standard unimodal distributions, enabling expressive multimodal action generation (Janner et al., 2022; Chi et al., 2025; Lipman et al., 2022; Wang et al., 2022; Ren et al., 2024; Hansen-Estruch et al., 2023; Zhang et al., 2025; Zhu et al., 2023). However, effectively using negative feedback from online interaction to improve policy performance remains challenging. Many existing methods train their generators by fitting actions selected or reweighted according to critic estimates (Peng et al., 2019; Peters & Schaal, 2007; Lv et al., 2025), thereby prioritizing actions with higher estimated values. These objectives assign lower weights to low-value actions but pro-

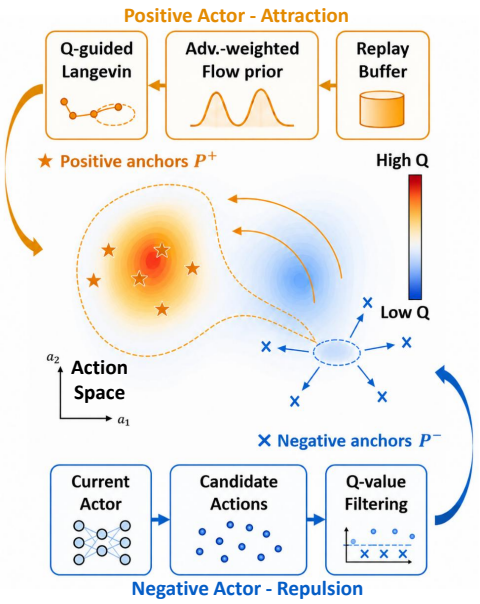


Figure 1: **Overview of DADC.** DADC combines attraction and repulsion in the action space via high- and low-value anchors.

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.

vide little explicit guidance for steering the policy away from them. Consequently, feedback from less successful interactions may remain underused, limiting the ability of policy to avoid repeated poor decisions and improve performance.

One way to better use this feedback is to treat poorly ranked actions as repulsive references, providing an explicit direction for moving policy samples away from them. Drift models offer geometric guidance by expressing distributional transport as a vector field with attractive and repulsive components (Deng et al., 2026). In their original formulation, however, repulsive samples are model-generated samples used for density correction, not semantically undesirable examples. Applying this mechanism to online RL requires constructing both attractive and repulsive anchors informed by the critic's value estimates.

Online nonstationarity makes reliable two-sided drift-field construction especially difficult. In image generation, real-data and model distributions provide distinct attraction and repulsion references. Online RL lacks a fixed high-quality target distribution, while the actor and critic evolve together. A naive adaptation partitions a single pool of current-policy actions into high- and low-Q anchors. However, critic ranking can only select among actions already present in that pool; promising actions absent from it cannot serve as attractive references. This limitation motivates drawing on replay experience to provide additional attractive references, while retaining current-policy candidates to identify relatively low-value actions for repulsion. Section 3.1 evaluates this design against the naive construction on Dog Run, where Naive Drift plateaus at a substantially lower return.

To address this limitation, we propose Dual-Source Anchor Drift Correction (DADC), an online RL algorithm that constructs attractive and repulsive anchors from complementary sources. For attraction, DADC samples initial anchors from an advantage-weighted conditional flow-matching (CFM) prior trained on replay actions and applies several steps of Q-guided Langevin refinement. For repulsion, it selects current-actor candidates with the lowest critic scores as relative-negative anchors. At each replay state, the two anchor sets define a drift correction that complements direct Q-value maximization, guiding actor outputs toward attractive anchors and away from repulsive anchors. The actor retains single-pass inference at deployment. Figure 1 illustrates the resulting dual-source construction. Across 14 DMControl and HumanoidBench tasks, DADC achieves the highest mean TAR among evaluated methods, and component ablations show complementary benefits from attraction and repulsion.

The main contributions of this paper are summarized as follows:

- We propose Dual-Source Anchor Drift Correction (DADC), an online RL algorithm that exploits negative feedback by using low-value policy samples as explicit repulsive anchors, while retaining single-pass inference.
- We propose a decoupled mechanism for generating attractive and repulsive anchors from distinct sources. Attractive anchors are generated using an advantage-weighted flow prior and Q-guided Langevin refinement, while repulsive anchors are selected from current-policy candidates with the lowest critic scores.
- Experiments on 14 DMControl and HumanoidBench tasks demonstrate average TAR improvements of 21.1% over FlowRL on the DMC Suite and 23.3% over SAC on HumanoidBench.

2 PRELIMINARY

Continuous-Control RL and Critic-Induced Energy References. We formulate sequential decision making as a Markov decision process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$ (Li, 2023). At timestep t, an agent in state $s_t \in \mathcal{S}$ samples an action $a_t \sim \pi(\cdot \mid s_t)$, receives reward $r(s_t, a_t)$, and transitions according to $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. Its objective is to maximize the expected discounted return:

$$J(\pi) = \mathbb{E}_{\pi,\mathcal{P}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{1}$$

Policy evaluation and improvement rely on the action-value function $Q^\pi(s, a)$. In continuous control, its landscape may contain multiple separated high-value action regions, motivating policies that can

represent diverse action modes. Given a critic Q_ψ held fixed during Langevin refinement, we define the critic-induced energy $\mathcal{E}_\psi(s,a) = -Q_\psi(s,a)$ and the auxiliary Gibbs reference:

$$p_{\psi,\beta}(a \mid s) = \frac{\exp\left(Q_\psi(s,a)/\beta\right)}{Z_{\psi,\beta}(s)}, \qquad Z_{\psi,\beta}(s) = \int_{\mathcal{A}} \exp\left(Q_\psi(s,\tilde{a})/\beta\right)\, \mathrm{d}\tilde{a}. \tag{2}$$

Here, $\beta > 0$ controls concentration, and $Z_{\psi,\beta}(s)$ normalizes the density over the bounded action space $\mathcal{A}$. This distribution serves as an auxiliary reference for action refinement, reflecting the value estimates of the current critic. Its action score, $\nabla_a \log p_{\psi,\beta}(a \mid s) = \beta^{-1}\nabla_a Q_\psi(s,a)$, provides the drift direction for Q-guided Langevin refinement without evaluating the partition function.

Drifting Models for Generative Modeling. Drifting Models (Deng et al., 2026) formulate generation as a one-step pushforward $q_\theta = (f_\theta)_{\#}p_\epsilon$, where f_θ maps noise $\epsilon \sim p_\epsilon$ to $x_\theta = f_\theta(\epsilon) \sim q_\theta$. Unlike ODE- or SDE-based generators, this formulation learns a direct particle displacement and supports 1-NFE generation. Given target samples $y^+ \sim p$, model samples $y^- \sim q$, and a kernel k_h, the drift field is

$$V_{p,q}(x) = \frac{\mathbb{E}_{y^+\sim p}[k_h(x,y^+)(y^+ - x)]}{Z_p(x)} - \frac{\mathbb{E}_{y^-\sim q}[k_h(x,y^-)(y^- - x)]}{Z_q(x)}. \tag{3}$$

Here, $Z_p(x) = \mathbb{E}_p[k_h(x,y)] + \varepsilon$ and $Z_q(x) = \mathbb{E}_q[k_h(x,y)] + \varepsilon$ are local normalizers, and $k_h(x,y) = \exp(-\|x-y\|_2^2/(2h^2))$ is a Gaussian kernel. Bandwidth h controls locality, while $\varepsilon > 0$ ensures numerical stability. The first term attracts particles toward target data, whereas the second repels them from local model density. Importantly, y^- denotes model-generated samples rather than semantically undesirable examples. Using the stop-gradient operator sg, the field defines the regression objective,

$$\mathcal{L}_{\text{drift}}(\theta) = \mathbb{E}_{\epsilon\sim p_\epsilon}\left[\left\|x_\theta - \mathrm{sg}(x_\theta + V_{p,q_\theta}(x_\theta))\right\|_2^2\right]. \tag{4}$$

With the field and reference samples detached, the per-sample semi-gradient is $\nabla^{\mathrm{sg}}_{x_\theta}\mathcal{L}_{\text{drift}} = -2V_{p,q_\theta}(x_\theta)$; gradient descent therefore moves samples along the field. This detached objective seeks a fixed point at which the prescribed displacement vanishes. Thus, it supplies both attractive and repulsive guidance. DADC adapts this mechanism to online RL by constructing value-informed attractive and repulsive anchors from distinct sources.

3 Method

In this section, we develop Dual-Source Anchor Drift Correction (DADC) for online policy optimization. We first formulate an actor objective that combines direct Q-value maximization with attractive and repulsive guidance, and examine a naive anchor construction based on current-policy samples. We then introduce a dual-source construction that uses replay experience for attractive references and current-policy candidates for repulsive references. Finally, we describe how the actor, critic, and auxiliary prior are trained together, while retaining single-pass action generation at deployment.

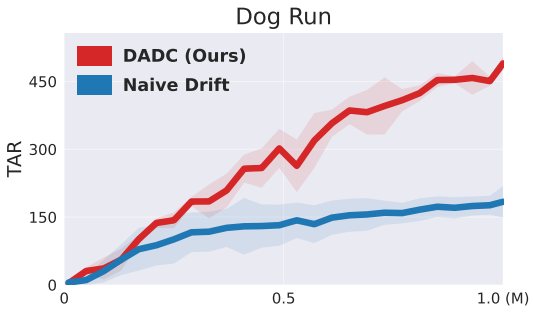


Figure 2: **Naive Drift vs. DADC on Dog Run.** Naive Drift constructs both anchor sets from current-actor samples and plateaus early, whereas DADC continues to improve.

3.1 Drift-Based Policy Optimization

We represent the policy by an implicit actor $a_\phi = \pi_\phi(s,z)$, where $z \sim \mathcal{N}(0,I)$. Direct Q-value maximization provides a local gradient signal at the generated action. To additionally guide the actor using reference actions, we introduce two anchor sets conditioned on each replay state $s \sim \mathcal{D}$: $\mathcal{Y}_s^+$ for attraction and $\mathcal{Y}_s^-$ for repulsion.

Using these sets as reference samples in Equation 3, we replace the expectations with empirical averages to obtain $V_s^+(a)$ and $V_s^-(a)$. We define the correction as $D_s(a) = V_s^+(a) - \omega V_s^-(a)$,

where $\omega > 0$ controls the relative strength of repulsion. To translate this correction into a policy update, we shift the current actor output a_ϕ by $D_s(a_\phi)$ to form a target action. Following Equation 4, we hold this target fixed using the stop-gradient operator and regress the actor output toward it. Combining this regression with direct Q-value maximization gives,

$$\mathcal{L}_{\text{Actor}}(\phi) = \mathbb{E}_{s\sim\mathcal{D},\, z\sim\mathcal{N}(0,I)}\Big[-Q_\psi(s, a_\phi) + \alpha \left\|a_\phi - \text{sg}(a_\phi + D_s(a_\phi))\right\|_2^2\Big]. \tag{5}$$

The first term provides local critic ascent. For each sampled action, the second term contributes an action-space gradient of $-2\alpha D_s(a_\phi)$ because the entire regression target is detached. Gradient descent therefore encourages the actor output to move along the prescribed correction field. Here, $\alpha > 0$ controls the strength of this additional guidance.

A naive implementation constructs both anchor sets from a single candidate pool $\mathcal{C}_s$ sampled from the current actor. The critic ranks these candidates, and the highest- and lowest-scoring subsets form $\mathcal{Y}_s^+$ and $\mathcal{Y}_s^-$, respectively. We refer to this variant as Naive Drift. Both anchor sets consist directly of candidates from $\mathcal{C}_s$, and the actor is optimized using Equation 5.

Figure 2 shows that Naive Drift improves initially but soon plateaus at a substantially lower return than DADC on Dog Run. This result motivates examining the reference actions available under the naive construction. Selecting the highest-scoring candidates provides preferred actions within the current pool, but cannot supply attractive references absent from that pool. For repulsion, current-policy candidates remain useful because they identify relatively low-value actions that the actor currently generates. We therefore draw on replay experience to construct attractive references while retaining current-policy candidates for repulsion. The next section describes this dual-source construction under the same actor objective.

3.2 Dual-Source Anchor Construction

To obtain attractive references informed by replay experience, DADC introduces an auxiliary conditional flow prior $q_\theta(a \mid s)$. This prior provides initial actions for attractive-anchor construction during training, while π_ϕ remains the actor used for environment interaction and deployment. Repulsive anchors are selected from this actor's current samples. Both anchor sets are conditioned on the same replay state s and enter the correction D_s in Equation 5.

Attractive Anchor Construction. The auxiliary prior is trained on replay actions using advantage-weighted conditional flow matching, emphasizing actions with higher estimated advantages. Its training objective is described in Section 3.3. For each replay state s, we sample N_+ initial actions from $q_\theta(\cdot \mid s)$. To adjust these replay-informed proposals using the current critic, we apply K steps of Q-guided Langevin refinement to each initialization a_0:

$$a_{k+1} = a_k + \frac{\eta}{\beta}\nabla_a Q_\psi(s, a_k) + \sqrt{2\eta}\,\epsilon_k. \tag{6}$$

Here, $\eta > 0$ is the step size, $\beta > 0$ is the temperature, and $\epsilon_k \sim \mathcal{N}(0, I)$. The critic is held fixed during refinement. The gradient term directs the proposals toward increasing critic values, while the noise introduces stochastic variation. After K steps, the resulting actions are detached to form $\mathcal{Y}_s^+ = \{y_{s,i}^+\}_{i=1}^{N_+}$.

Repulsive Anchor Construction. For the same replay state s, we draw M candidate actions from the current actor $\pi_\phi(\cdot \mid s)$ and evaluate them using $Q_\psi(s, \cdot)$. We retain the m lowest-scoring candidates, where $1 \le m < M$, and detach them to form $\mathcal{Y}_s^- = \{y_{s,j}^-\}_{j=1}^m$. Sampling from the current actor keeps the repulsive references tied to actions the policy currently produces, while critic ranking directs their selection toward the lower-value part of that candidate pool. Their negative status is therefore relative to the sampled alternatives under the current critic. Together with $\mathcal{Y}_s^+$, these anchors define the attractive and repulsive components of the actor correction.

3.3 Training and Inference

DADC is trained off-policy using transitions collected by the implicit actor. The critic provides value estimates for policy optimization and anchor construction, while the flow prior is trained separately on replay actions. We alternate their updates with actor optimization using the detached dual-source anchors.

Critic Update. For each gradient step, we sample transitions from the replay buffer and update Q_ψ by minimizing the temporal-difference Bellman error:

$$\mathcal{L}_{\text{Critic}}(\psi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D},\, a'\sim\pi_\phi(\cdot|s')}\Big[\big(Q_\psi(s,a) - \big(r + \gamma Q_{\bar{\psi}}(s',a')\big)\big)^2\Big]. \tag{7}$$

Here, $\gamma \in [0,1)$ is the discount factor, and $\bar{\psi}$ denotes the target-critic parameters. The next-state action a' is sampled from the current actor, and the target critic supplies the bootstrap value.

Prior Update. Every d_{target} gradient steps, we update the auxiliary flow prior using advantage-weighted conditional flow matching. The prior is parameterized by a velocity field v_θ. Given $(s,a) \sim \mathcal{D}$, base noise $\epsilon \sim \mathcal{N}(0,I)$, and $t \sim \mathcal{U}(0,1)$, we define $a_t = (1-t)\epsilon + ta$ and minimize

$$\mathcal{L}_{\text{Flow}}(\theta) = \mathbb{E}\Big[w(s,a)\,\|v_\theta(s,a_t,t) - (a-\epsilon)\|_2^2\Big]. \tag{8}$$

The weight $w(s,a) \propto \exp(Q_\psi(s,a) - V(s))$ emphasizes replay actions with higher estimated advantages, where $V(s)$ is an empirical state-value baseline. These critic-derived weights are held fixed during each prior update. To sample from the trained prior, we integrate $\mathrm{d}x_t/\mathrm{d}t = v_\theta(s,x_t,t)$ from $x_0 \sim \mathcal{N}(0,I)$ at $t=0$ to $t=1$, taking $a_0 = x_1$ as the anchor initialization.

Actor and Target Updates. On the same scheduled step, we construct both anchor sets as described in Section 3.2 and update the actor using Equation 5. The critic parameters are held fixed, while the action gradient of $Q_\psi(s,a_\phi)$ is retained. The entire drift regression target is detached, so gradients pass through the predicted action a_ϕ without propagating through anchor generation, Langevin refinement, or candidate ranking. The actor loss therefore updates ϕ without updating the auxiliary prior or critic parameters. We then softly update the target critic as $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$, where $\tau \in (0,1]$.

Algorithm 1 summarizes the complete procedure. At deployment, the actor maps the state and latent noise directly to an action in a single forward pass. Flow integration, anchor refinement, and critic-based candidate selection are confined to training.

Algorithm 1 Dual-Source Anchor Drift Correction (DADC)

Require: Actor π_ϕ, Flow Prior q_θ parameterized by v_θ, Critic Q_ψ, Replay Buffer $\mathcal{D}$

Initialize ϕ, θ, ψ, and target critic $\bar{\psi} \leftarrow \psi$
while training is not finished **do**
  Interact with the environment using π_ϕ and store transitions (s,a,r,s') in $\mathcal{D}$
  for each gradient step **do**
    Sample a mini-batch of transitions from $\mathcal{D}$
    1. Update Critic
    Update ψ by minimizing $\mathcal{L}_{\text{Critic}}(\psi)$ via Eq. 7
    if every d_{target} steps **then**
      2. Update Flow Prior
      Compute $w(s,a) \propto \exp(Q_\psi(s,a) - V(s))$ and update θ via Eq. 8
      3. Construct Dual-Source Anchors
      For each s, sample from $q_\theta(\cdot \mid s)$ and apply Eq. 6 to form detached $\mathcal{Y}_s^+$
      Sample M actor candidates and retain the critic-ranked bottom-m actions as detached $\mathcal{Y}_s^-$
      Construct the contrastive drift field $D_s = V_s^+ - \omega V_s^-$
      4. Update Actor
      Update ϕ by minimizing $\mathcal{L}_{\text{Actor}}(\phi)$ via Eq. 5
      5. Update Target Critic
      Soft update $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
    end if
  end for
end while

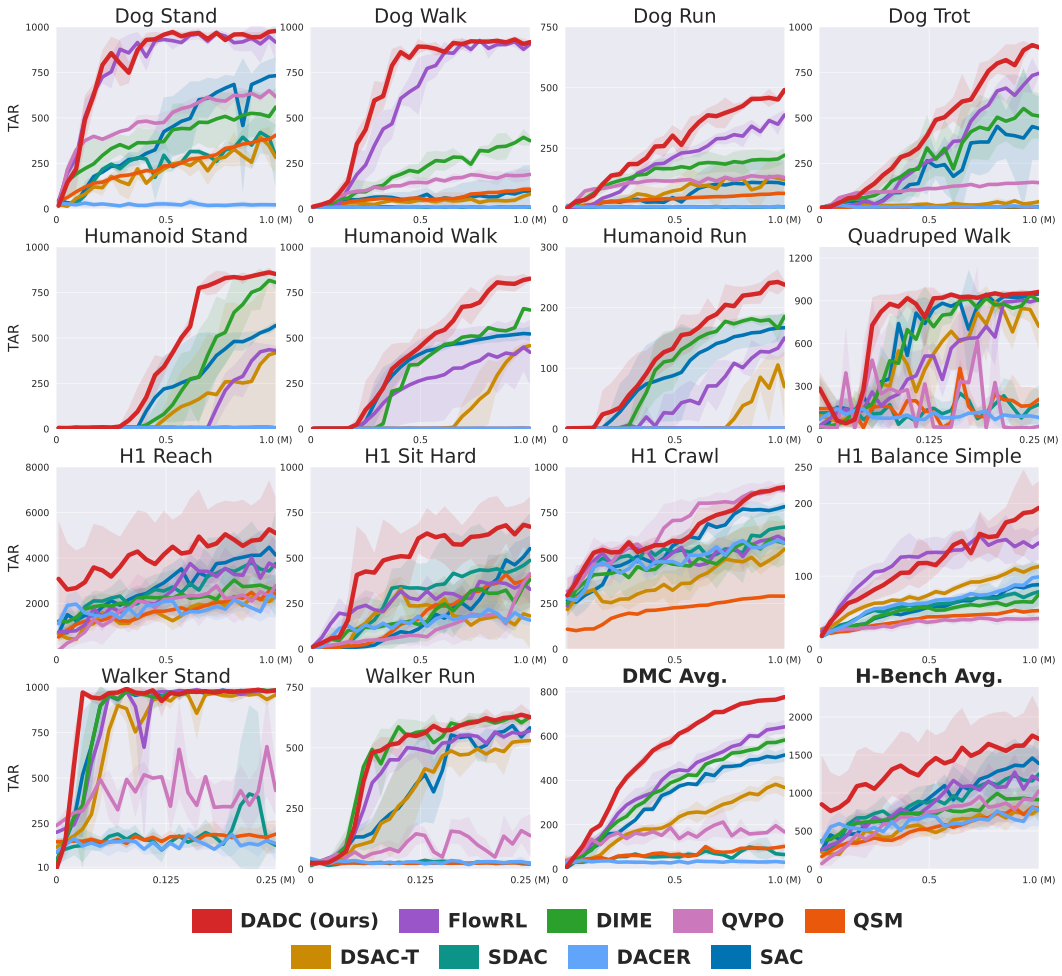


Figure 3: **Training curves on the DMControl and HumanoidBench tasks.** The solid lines depict the mean evaluation returns across 5 random seeds, while the shaded areas denote standard-deviation bands (± 1 std). The bottom-right panels report the aggregated average performance for DMC and H-Bench, respectively.

4 Experiment

4.1 Main Experiment

Benchmarks. We evaluate DADC on 14 continuous control tasks from the DeepMind Control Suite (DMC) (Tassa et al., 2018) and HumanoidBench (H-Bench) (Sferrazza et al., 2024). The 10 DMC tasks include dog, humanoid, quadruped, and walker agents, and cover standing, walking, trotting, and running. The four H-Bench tasks use the Unitree H1 humanoid and include *H1 Reach*, *H1 Sit Hard*, *H1 Crawl*, and *H1 Balance Simple*. These tasks cover different morphologies, action dimensions, and coordination requirements. Detailed task descriptions and environment settings are provided in Appendix A.1.

Baselines. We compare DADC with eight representative online RL baselines: (1) **SAC** (Haarnoja et al., 2018), a model-free off-policy method under the maximum-entropy framework; (2) **DACER** (Wang et al., 2024), which uses reverse diffusion as the policy and regulates exploration with a Gaussian mixture model; (3) **SDAC** (Ma et al., 2025), a max-entropy method that trains diffusion policies via reweighted score matching without optimal-policy samples; (4) **DSAC-T** (Duan

Table 1: Total Average Return (TAR) on 10 DMC Suite tasks and 4 H-Bench tasks. Mean ± Std over 5 seeds. **Bold** = best. Higher is better.

Task	DACER	DSAC-T	SDAC	QSM	QVPO	DIME	SAC	FlowRL	DADC (Ours)
Dog Stand	22 ± 11	282 ± 86	289 ± 174	405 ± 11	616 ± 27	559 ± 105	732 ± 101	916 ± 105	**978 ± 9**
Dog Walk	11 ± 2	81 ± 14	8 ± 3	108 ± 17	189 ± 30	374 ± 26	98 ± 143	905 ± 16	**917 ± 7**
Dog Run	7 ± 5	120 ± 65	8 ± 3	63 ± 7	129 ± 16	221 ± 31	102 ± 96	387 ± 21	**490 ± 16**
Dog Trot	8 ± 6	39 ± 16	12 ± 3	10 ± 2	144 ± 9	510 ± 126	442 ± 175	745 ± 13	**888 ± 16**
Humanoid Stand	6 ± 0	418 ± 390	6 ± 0	6 ± 0	5 ± 4	805 ± 40	568 ± 23	431 ± 26	**851 ± 6**
Humanoid Walk	2 ± 0	457 ± 18	1 ± 0	2 ± 0	2 ± 0	695 ± 47	522 ± 41	421 ± 181	**826 ± 9**
Humanoid Run	1 ± 0	70 ± 59	1 ± 0	1 ± 0	1 ± 0	186 ± 7	167 ± 23	150 ± 29	**238 ± 13**
Quadruped Walk	80 ± 56	720 ± 168	172 ± 135	207 ± 70	18 ± 18	903 ± 25	947 ± 11	908 ± 18	**962 ± 14**
Walker Stand	140 ± 99	958 ± 13	130 ± 44	192 ± 76	431 ± 139	982 ± 4	982 ± 3	981 ± 4	**983 ± 19**
Walker Run	22 ± 6	530 ± 18	20 ± 4	24 ± 1	138 ± 87	627 ± 24	582 ± 30	570 ± 25	**628 ± 25**
AVG. DMC Suite	30 ± 19	367 ± 85	65 ± 37	102 ± 18	167 ± 33	586 ± 44	514 ± 65	641 ± 44	**776 ± 11**
H1 Reach	2274 ± 903	2307 ± 62	3757 ± 2327	2439 ± 640	2788 ± 579	2610 ± 642	4129 ± 625	3615 ± 1510	**5093 ± 1569**
H1 Sit Hard	158 ± 29	181 ± 187	487 ± 261	403 ± 153	413 ± 171	380 ± 158	552 ± 71	328 ± 79	**672 ± 167**
H1 Crawl	583 ± 28	549 ± 12	670 ± 125	290 ± 350	873 ± 17	582 ± 119	782 ± 24	602 ± 34	**889 ± 31**
H1 Balance Simple	99 ± 16	113 ± 4	79 ± 2	53 ± 2	42 ± 5	75 ± 22	88 ± 29	146 ± 22	**194 ± 36**
AVG. H-Bench	779 ± 244	788 ± 66	1248 ± 679	796 ± 286	1029 ± 193	912 ± 235	1388 ± 187	1173 ± 411	**1712 ± 451**

et al., 2025), a distributional RL method that improves Q-value estimation and training stability; (5) **QSM** (Psenka et al., 2023), which matches diffusion-policy scores to Q-function action gradients; (6) **FlowRL** (Lv et al., 2025), which parameterizes policies with flow models and optimizes them using a Wasserstein-2-regularized objective; (7) **QVPO** (Ding et al., 2024), which trains diffusion policies using a Q-weighted variational objective together with entropy regularization; and (8) **DIME** (Celik et al., 2025), which formulates maximum-entropy diffusion policy optimization through approximate inference. Implementation details, codebase references, and complete configurations are provided in Appendix A.2.

Table 2: **Component ablation on Dog Run and H1 Crawl.** Attraction denotes the refined attractive-anchor branch, while Repulsion denotes the critic-ranked repulsive-anchor branch. Results are reported as mean ± standard deviation over 5 seeds.

Setting	Local-Q	Attraction	Repulsion	Dog Run	H1 Crawl
Local-Q only	✓	–	–	415 ± 13	729 ± 110
Local-Q + Attraction	✓	✓	–	462 ± 27	837 ± 64
Local-Q + Repulsion	✓	–	✓	441 ± 32	790 ± 24
Full DADC	✓	✓	✓	**490 ± 16**	**889 ± 31**

Main results. The training curves and detailed numerical comparisons are presented in Figure 3 and Table 1, respectively. DADC consistently achieves the highest mean Total Average Return (TAR) among the evaluated methods across all 14 continuous control tasks. On the DMC Suite, DADC achieves an average TAR of **776**, exceeding the strongest aggregate baseline FlowRL (641) by **+21.1%** and the SAC baseline (514) by **+51.0%**. Its advantage is especially clear in complex multi-gait locomotion, where DADC maintains strong learning performance across different agent morphologies and control objectives. In *Dog Run*, DADC scores 490, outperforming the second-best FlowRL (387) by **+26.6%**. In *Humanoid Run*, DADC reaches 238, exceeding the second-best DIME (186) by **+28.0%**. On HumanoidBench, DADC also performs strongly on the whole-body control benchmark, achieving an average TAR of **1712**, surpassing SAC (1388) by **+23.3%** and FlowRL (1173) by **+46.0%**. In the challenging *H1 Reach* task, DADC achieves a mean return of 5093, compared with 4129 for the second-best SAC, corresponding to a **+23.3%** gain. This result, together with the other H-Bench tasks, shows that the improvements extend across different whole-body control objectives and coordination requirements. On *H1 Balance Simple*, DADC scores 194, compared with 146

for FlowRL, an improvement of **+32.9%**. In coordination-heavy *H1 Crawl*, DADC reaches 889, followed closely by QVPO at 873.

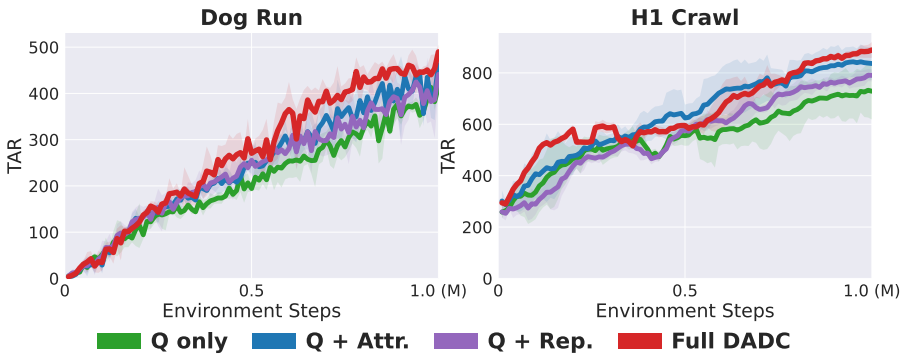


Figure 4: **Component Ablation.** Learning curves of four component configurations on (a) Dog Run and (b) H1 Crawl.

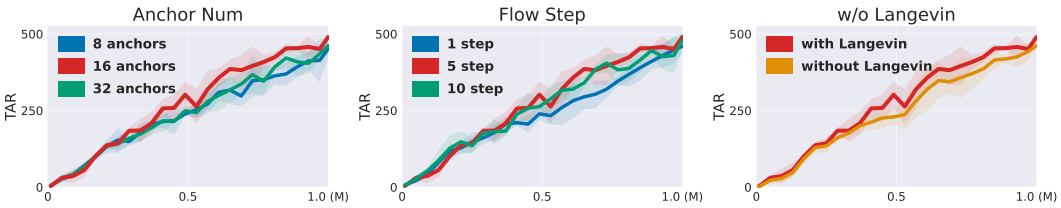


Figure 5: **Attractive Anchor Construction on Dog Run.** Left to right: positive-anchor count, flow integration steps, and Langevin refinement. All other settings remain fixed.

4.2 Component Ablation Study

We conduct component ablations on *Dog Run* from the DMC Suite and *H1 Crawl* from H-Bench. The two tasks involve different agent morphologies and control requirements, allowing us to examine whether the component trends extend beyond a single environment. All variants use the same actor and critic architectures, training budget, evaluation protocol, and five random seeds. Figure 4 presents the learning curves, while Table 2 reports the corresponding numerical results. We further examine the construction of attractive anchors on *Dog Run*, as shown in Figure 5.

Component analysis. The *Local-Q only* variant retains the direct critic objective and removes both anchor fields. *Local-Q + Attraction* adds the attractive branch constructed from the flow prior and critic-guided refinement, while *Local-Q + Repulsion* instead adds the repulsive branch constructed from critic-ranked actor samples. Full DADC combines the direct critic objective with both anchor branches. On *Dog Run*, adding attraction or repulsion to Local-Q increases the mean TAR from 415 to 462 and 441, respectively. On *H1 Crawl*, the corresponding return increases from 729 to 837 and 790. Combining both branches further improves performance, with Full DADC reaching 490 ± 16 on Dog Run and 889 ± 31 on H1 Crawl. Relative to Full DADC, removing repulsion reduces the mean return by 28 and 52 points on the two tasks, respectively. Removing attraction produces larger decreases of 49 and 99 points. Thus, attraction provides the larger improvement when each branch is added separately to Local-Q, while repulsion remains beneficial when attraction is already present. The same ordering of mean returns across both tasks supports the complementary contributions of the two anchor branches under different control requirements.

Attractive anchor construction. We vary the positive-anchor count, the number of flow integration steps, and the use of Q-guided Langevin refinement. Each experiment changes one factor while retaining direct Q-value optimization and critic-filtered repulsion, with all remaining settings fixed.

As shown in Figure 5, 16 positive anchors achieve the highest mean return among the tested counts of 8, 16, and 32. Increasing the flow integration steps from one to five improves learning performance, while ten steps yield similar performance to five. Langevin refinement also improves the final mean return over using unrefined flow-prior samples. These results support the default configuration of 16 positive anchors, five flow steps, and Langevin refinement on Dog Run under the evaluated training budget.

5 Related Work

Generative Models. Diffusion models (Janner et al., 2022; Chi et al., 2025) learn iterative denoising processes that transform noise into data through reverse-time sampling. To improve sampling efficiency and scalability, specialized solvers (Lu et al., 2022), consistency models (Song et al., 2023), and latent formulations (Rombach et al., 2022) have been developed. Flow matching (Lipman et al., 2022) instead learns continuous-time velocity fields, with Rectified Flow (Liu et al., 2022) encouraging straighter transport paths and MeanFlow (Geng et al., 2025) enabling efficient one-step generation. More recently, Drifting Models (Deng et al., 2026) formulate generation as the evolution of a pushforward distribution governed by a macroscopic drifting field. Their kernelized construction uses target samples as attractive anchors and generated samples as repulsive anchors to update a single-pass generator. This geometric mechanism provides the generative-modeling basis for DADC, which adapts attraction and repulsion to critic-conditioned online policy correction.

Generative Models in Online RL. Generative policies have increasingly been studied in online RL to represent complex and multimodal action distributions (Haarnoja et al., 2017; Mazoure et al., 2020). Unlike offline policy learning, online RL provides no fixed expert distribution, while both the replay buffer and critic evolve during training. Existing methods incorporate critic feedback through several objectives. DACER (Wang et al., 2024) and DACERv2 (Wang et al., 2025) optimize diffusion policies using critic gradients, while QSM (Psenka et al., 2023) and related score-based policy gradients (Zou et al., 2025; Sun et al., 2025) align policy scores with critic action gradients. SAC Flow (Zhang et al., 2026) directly trains flow policies through actor-critic updates. QVPO (Ding et al., 2024) uses a Q-weighted variational objective, SDAC (Ma et al., 2025) applies reweighted score matching, DIME (Celik et al., 2025) derives an approximate-inference objective for maximum-entropy policy optimization, and FlowRL (Lv et al., 2025) regularizes flow-policy updates through a Wasserstein-2 surrogate. MaxEntDP (Dong et al., 2025) further trains diffusion policies using Q-weighted denoising targets. DADC constructs attractive and repulsive anchors from distinct sources. A replay-trained flow prior and Q-guided Langevin refinement produce the attractive anchors, while current-policy candidates with the lowest critic scores serve as repulsive anchors. Together, these anchors provide explicit attraction and repulsion that complement direct Q-value maximization.

6 Conclusion

In this paper, we introduced Dual-Source Anchor Drift Correction (DADC) for online RL. DADC constructs attractive and repulsive anchors from distinct sources to incorporate negative feedback into policy optimization. For each replay state, an advantage-weighted flow prior trained on replay data generates initial attractive anchors, which undergo Q-guided Langevin refinement. Repulsive anchors are selected from current-policy candidates with the lowest critic scores. These two anchor sets define a drift correction that complements direct Q-value maximization. The prior and anchor construction are required only during training, preserving single-pass actor inference at deployment. Across 14 DMControl and HumanoidBench tasks, DADC achieves the highest mean TAR among the evaluated methods on every task. Component ablations on Dog Run and H1 Crawl support the complementary contributions of attraction and repulsion.

Limitations and Future Work. DADC relies on the critic for both anchor refinement and ranking. Inaccurate estimates during early training may therefore misdirect attractive anchors or select unreliable repulsive anchors. The flow prior, candidate evaluations, and refinement steps increase training complexity and computational cost, while deployment uses only the actor. Future work will study critic uncertainty, adaptive anchor selection and weighting, more efficient refinement, and applications to physical systems and tasks with longer horizons.

AI USE STATEMENT

We used generative AI tools to assist with manuscript drafting and language editing, to improve the organization and clarity of the method and experiment sections, and to discuss theoretical arguments and interpretations of experimental results. AI-assisted suggestions informed the authors' revisions. The authors are responsible for verifying the correctness and originality of all AI-assisted content and take full responsibility for the final manuscript, including its text, theoretical claims, experimental results, and conclusions.

REFERENCES

Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. *arXiv preprint arXiv:2502.02316*, 2025.

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. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Xiaoyi Dong, Jian Cheng, and Xi Sheryl Zhang. Maximum entropy reinforcement learning with diffusion policy. *arXiv preprint arXiv:2502.11612*, 2025.

Jingliang Duan, Wenxuan Wang, Liming Xiao, Jiaxin Gao, Shengbo Eben Li, Chang Liu, Ya-Qin Zhang, Bo Cheng, and Keqiang Li. Distributional soft actor-critic with three refinements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

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.

Nicolas Heess, Dhruva Tb, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Shengbo Eben Li. Reinforcement learning for sequential decision and optimal control. 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *arXiv preprint arXiv:2506.12811*, 2025.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

Bogdan Mazoure, Thang Doan, Audrey Durand, Joelle Pineau, and R Devon Hjelm. Leveraging exploration in off-policy algorithms via normalizing flows. In *Conference on Robot Learning*, pp. 430–444. PMLR, 2020.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

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.

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. *arXiv preprint arXiv:2409.00588*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. 2023.

Mingyang Sun, Pengxiang Ding, Weinan Zhang, and Donglin Wang. Score-based diffusion policy compatible with reinforcement learning via optimal transport. *arXiv preprint arXiv:2502.12631*, 2025.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

Yinuo Wang, Likun Wang, Mining Tan, Wenjun Zou, Xujie Song, Wenxuan Wang, Tong Liu, Guojian Zhan, Tianze Zhu, Shiqi Liu, et al. Enhanced dacer algorithm with high diffusion efficiency. *arXiv preprint arXiv:2505.23426*, 2025.

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.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

Yixian Zhang, Shu-ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. Sac flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *International Conference on Learning Representations*, volume 2026, pp. 121983–122016, 2026.

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.

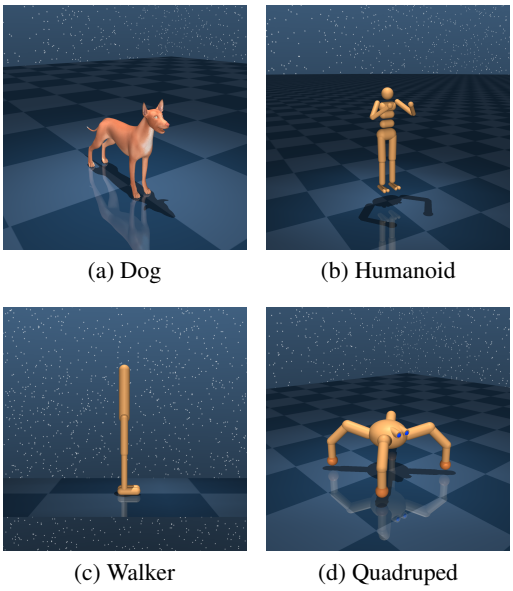
Wenjun Zou, Yinuo Wang, Tong Liu, Bin Shuai, Liming Xiao, Yinsong Ma, Jingliang Duan, and Shengbo Eben Li. Multimodal reinforcement learning with score-based policy. *IEEE Transactions on Automation Science and Engineering*, 22:24105–24119, 2025.

A Environmental Details

A.1 Benchmark Introduction

DeepMind Control Suite. We evaluate on 10 challenging continuous control tasks across four different agent morphologies: `dog`, `humanoid`, `walker`, and `quadruped` (see Figure 6). These tasks fall into two categories: (1) *Standing tasks*, where the agent must maintain an upright balance (e.g., Stand), and (2) *Moving tasks*, which additionally require the agent to move at a specific target velocity (e.g., Walk, Trot, Run).

For moving tasks, the reward is generally defined as the product of the standing reward and the forward velocity reward, i.e., ***Reward*** $=$ (***Standing reward***) $\times$ (***Forward velocity reward***).



(a) Dog (b) Humanoid

(c) Walker (d) Quadruped

Figure 6: Four different agent morphologies evaluated in the DeepMind Control Suite.

Dog: A high-dimensional quadruped agent designed for complex locomotion, featuring tasks like stand, walk, trot, and run.

Humanoid: A complex 3D bipedal agent with 21 degrees of freedom, requiring sophisticated balancing and coordination.

Walker: A planar bipedal agent restricted to 2D motion, serving as a fundamental benchmark for basic locomotion dynamics.

Quadruped: A generic 3D four-legged agent requiring the coordination of multiple limbs to achieve stable gaits.

Rewards: For moving tasks, the reward is generally defined as the product of a ***Standing reward*** (encouraging an upright, stable posture) and a ***Forward velocity reward*** (ensuring the agent moves at the task's target speed).

HumanoidBench. We consider 4 complex whole-body locomotion and manipulation tasks involving a Unitree H1 robot. The robot is initialized to a standing position, with random noise added to all joint positions during each episode reset. Their specific goals and reward structures are presented below.

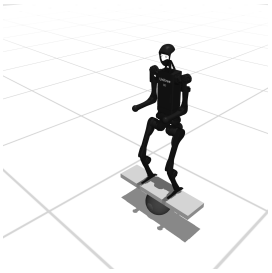


Figure 7: H1 Balance Simple

Objective. Balance on an unstable board. In the `simple` variant, the spherical pivot beneath the board remains fixed.

Reward: $R(s,a) = (e \times \texttt{still}) \times (\texttt{height}_{\text{robot}} \times \texttt{upright})$, where `still` encourages stationary balance, e rewards small control effort, and the remaining terms reward maintaining a high, vertical torso.

Termination. 1000 steps, or when $z_{\text{robot}} < 0.8$, the board collides with the floor, or the sphere collides with anything other than the floor and board.

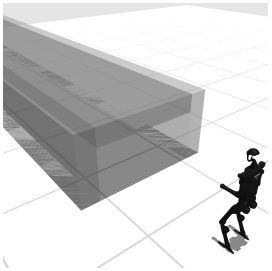


Figure 8: H1 Crawl

Objective. Keep forward velocity close to $1\,\mathrm{m/s}$ while passing inside a tunnel.

Reward: $R = \texttt{tunnel} \times \big(0.1 + 0.25\min(\texttt{height}_{\text{crawl}}, \texttt{height}_{\text{IMU}}) + 0.25\texttt{orientation} + 0.4\texttt{speed}\big)$. Here, `tunnel` rewards being inside the tunnel's lateral bounds, `orientation` matches the body to a specific crawling quaternion, and the `height` terms encourage a low, stable posture.

Termination. 1000 steps.

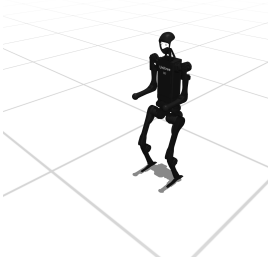


Figure 9: H1 Reach

Objective. Reach a randomly initialized 3D point with the left hand.

Reward: $R(s,a) = 10 - 4\texttt{penalty}_{\text{motion}} + \texttt{health} + \texttt{close} + \texttt{success}$. The `health` term $(5 - 2z_{\text{pelvis.proj}})$ rewards keeping the pelvis elevated , while `close` and `success` provide tiered bonuses as the hand approaches within specific distance thresholds to the target.

Termination. 1000 steps.



Figure 10: H1 Sit Hard

Objective. Sit onto a chair. In the `hard` setting, the robot's orientation $\alpha \in [-1.8, 1.8]$ rad and position are randomized, requiring active navigation and alignment before sitting.

Reward: $R(s,a) = ((0.5 \cdot \texttt{sitting_z} + 0.5 \cdot \texttt{sitting_x} \cdot \texttt{sitting_y}) \times \texttt{upright} \times \texttt{posture}) \times e \times \texttt{mean}(\texttt{still_x}, \texttt{still_y})$. The reward focuses on minimizing the distance between the robot's pelvis and the chair's seat while maintaining a proper seated posture.

Termination. 1000 steps, or when $z_{\text{pelvis}} < 0.5$.

A.2 Reproducibility Statement & Detailed Hyperparameters

We base our foundational experimental framework on the released official FlowRL codebase (https://github.com/bytedance/FlowRL). To facilitate reproducible research and ensure double-blind review compliance, we have included the core source code of our algorithm in the supplementary ZIP attachment. The complete repository will be fully open-sourced in the future. During the training and evaluation phases, we utilize Weights & Biases (WandB) for consistent experimental tracking, metric logging, and transparent visualization across all random seeds.

In this paper, we evaluate each algorithm across 5 random seeds to ensure statistical significance. The computational infrastructure utilized for our experiments consists of high-performance computation nodes equipped with 2 × NVIDIA RTX A6000 and 2 × NVIDIA A100 GPUs. Taking the highly demanding *Dog Run* task in the DMC Suite as a practical example, the computational time required to train for 1M iterations is approximately 9 hours.

For baseline comparisons, we implemented the respective algorithms directly within the FlowRL framework or adopted their official repository implementations to ensure the utmost fairness. Specifically, we adapt the official codebases for DACER (https://github.com/happy-yan/DACER-Diffusion-with-Online-RL), SDAC (https://

github.com/mahaitongdae/diffusion_policy_online_rl), and QSM (https://github.com/escontra/score_matching_rl). For SAC and DSAC-T, we utilize the robust implementations provided by the GOPS library (https://github.com/Intelligent-Driving-Laboratory/GOPS). QVPO and DIME use their official implementations at https://github.com/wadx2019/qvpo and https://github.com/ALRhub/DIME, respectively. We strictly align critical experimental settings, such as the total number of environment interactions and the evaluation protocols, across all algorithms.

The detailed hyperparameter configurations used for DADC are meticulously extracted from our training environment and summarized in Table 3. These values are kept consistent across all evaluated tasks unless otherwise specified in the task-specific sections.

Table 3: Detailed Hyperparameters for DADC.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Training			
Learning rate	3×10^{-4}	Optimizer	Adam
Replay buffer size	1_000_000	Discount factor (γ)	0.99
Batch size	256	Target update rate (τ)	0.05
Start steps	5000	Target update interval (d_{target})	2
Updates per step	1	Quantile	0.9
Drift & Flow Prior			
Drift imitation weight (α)	1.0	Number of positive anchors	16
Flow drift coefficient (ω)	0.1	Flow integration steps	5
Refinement step size (η)	0.02	Refinement steps	4
Refinement noise scale	0.01		
Architecture			
Hidden dimensions	512		

A.3 Implementation of Critic-Ranked Bottom-m Selection

For each replay state s, we sample $M = 16$ candidate actions from $\pi_\phi(\cdot \mid s)$, rank them by $Q_\psi(s, \cdot)$, and retain the bottom $m = 4$ actions as the detached repulsive-anchor set $\mathcal{Y}_s^-$. This retains the critic-relative lower 25% of each candidate pool without an additional absolute threshold and does not assume that the selected actions are globally suboptimal.

A.4 Efficiency Comparison

We evaluated the inference efficiency on the Dog Run task, specifically measuring the time required for 200 forward passes given environment observations. As shown in Table 4, DADC achieves a latency of 0.300 ms at a batch size of 256, demonstrating a significant speed advantage over the 0.536 ms observed for the single-step FlowRL baseline. These results suggest that our architecture provides a highly efficient profile for real-time inference.

This latency advantage fundamentally stems from their distinct action generation paradigms. DADC maps states to actions through a single network forward pass. In contrast, FlowRL generates actions by solving an ordinary differential equation (ODE). Even when constrained to a single integration step, the numerical solvers employed by FlowRL (such as the midpoint method) require multiple network evaluations. Combined with its wider input space (concatenating states, noisy actions, and time), FlowRL incurs inherent computational overheads that the direct mapping approach of DADC effectively avoids.

Table 5 illustrates the overall training throughput, measured as the rate of environment interaction and training updates per second. DADC achieves 26.7 steps/s, representing a moderate decrease compared to the 32.8 steps/s of FlowRL. This variation is primarily a consequence of the increased architectural complexity, specifically the 5-step flow prior, the generation of multiple anchors, and the iterative Langevin refinement. Nevertheless, this throughput remains within an acceptable range

Table 4: Inference latency comparison on the Dog Run task. Latency is evaluated over 200 forward passes across different batch sizes to reflect the real-time processing capability.

Batch Size	FlowRL (flow step=1) (ms)	DADC (Ours) (ms)
256	0.536	0.300
1024	0.584	0.360

for online reinforcement learning, serving as a reasonable trade-off for the enhanced multimodal modeling and the active suppression of suboptimal actions.

Table 5: Training throughput comparison on the Dog Run task, measured in steps per second (including environment interaction and gradient updates).

Method	FlowRL (flow steps=1)	DADC(Ours)
steps/s	32.8	26.7