CAN K HEADS EXPLORE BETTER THAN ONE IN ONLINE REINFORCEMENT LEARNING?

Anonymous authors
Paper under double-blind review

ABSTRACT

In reinforcement learning, generative policies can represent diverse behaviors, yet effective exploration remains a challenge in online reinforcement learning. Ensembles offer a natural approach, but training policy heads against a shared objective can reduce the diversity needed for exploration. We introduce Ensemble Sampling Actor-Critic (ESAC), a framework that combines generative policy populations with critic guided training and action selection. Circular critic pairing supplies head specific value weightings for diffusion policy regression, while pooled proposals allow the agent to explore actions from the full population. For fixed regression objectives, we prove contraction in output space under a shared objective and characterize when critic dependent targets induce distinct update operators. We further introduce Time Indexed Disagreement Exploration (TIDE), which uses policy disagreement across denoising times to guide action selection and calibrates its bonus through correlation with critic disagreement. Experiments with diffusion and flow matching policies on eight dense reward continuous control benchmarks over five seeds show improved returns over single policy controls on several locomotion tasks. Controlled comparisons and behavioral analyses examine the contribution of the policy population, while ablations establish the task-dependent benefit of calibrated disagreement. These results demonstrate the potential of generative policy ensembles for directed exploration and characterize the trade-off between improved returns and additional computation.

1 INTRODUCTION

In reinforcement learning, a policy is a probability distribution over actions given the current state. Standard actor-critic methods use a Gaussian policy, whose fixed shape concentrates probability around a single action (Haarnoja et al., 2018). Generative policies instead learn a flexible action distribution with no fixed shape: they start from random noise and transform it into an action through a sequence of learned refinement steps, as in diffusion (Ho et al., 2020, Song et al., 2020b) and flow matching (Lipman et al., 2022, Liu et al., 2022). This flexibility lets them express the complex, multimodal behavior that arises in real-world settings, and they have become a leading choice for learning such behavior, particularly in robotics (Chi et al., 2025). Current state-of-the-art methods train generative policies online, from the agent's own interaction with the environment (Ma et al., 2025, Ding et al., 2024, Wang et al., 2024, McAllister et al., 2026).

In online reinforcement learning, however, being able to represent several modes does not mean the agent will discover them. The agent learns only from the experience it collects, so it must decide which untried behaviors are worth trying. This is the exploration problem, one of the central challenges of RL, and it is especially hard in continuous control, where there are infinitely many actions to try. Existing online methods for generative policies explore the same way Gaussian methods do, by adding random noise to actions or by rewarding randomness through an entropy bonus (Ma et al., 2025, Wang et al., 2024, Dong et al., 2025). This keeps the agent close to its current behavior. The policy concentrates on the modes that have already paid off, and a promising alternative can remain unlikely to be chosen even though the policy could represent it.

Ensembles offer a way to keep alternatives alive. If each policy in an ensemble settles on a different mode, the ensemble as a whole proposes a wider range of behaviors than any single policy. Disagreement between the policies also shows the agent where it is uncertain, and therefore where exploring is most informative. Ensemble disagreement already guides exploration for value-based agents (Osband et al., 2016) and Gaussian actor-critic methods (Lee et al., 2021). Current methods learn several behaviors within a single diffusion policy (Li et al., 2024) or use an ensemble of critics to filter the candidate actions of a pretrained policy (Luo et al., 2026).

Building such an ensemble exposes a problem. Ensembles usually rely on random initialization to make their models different (Lakshminarayanan et al., 2017), but for generative policies this difference does not last. Online methods train a generative policy with the help of a critic, a network that estimates how much future reward an action will bring. The policy proposes candidate actions,

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.

the critic scores them, and the policy learns to imitate the candidates, giving more weight to those with higher scores (Ma et al., 2025, Ding et al., 2024). If every policy in the ensemble uses the same critic, every policy learns to imitate the same weighted candidates, and training pulls all of them toward the same behavior, which causes collapse. On the harder locomotion tasks like Ant, policies that share a critic produce nearly identical actions early in training, even though their network weights remain different, which shows that collapse follows directly from how the policies are trained.

We propose Ensemble Sampling Actor-Critic (ESAC), which prevents collapse by giving each policy its own training signal. Instead of one shared critic, ESAC trains several critics, and each policy scores its candidate actions with its own pair of critics. The critics start from different random initializations, so they score the same candidates differently, and each policy learns to imitate a different weighting of them. Each policy uses two critics rather than one because a single critic tends to overestimate how good actions are, and taking the lower of the two scores counteracts this (Fujimoto et al., 2018). We arrange the pairs in a circle: policy k uses critics k and $k+1$, and the last policy wraps around to the first critic. With this arrangement, K critics are enough for K policies, and no two policies share the same pair. Figure 1 summarizes the full framework: ESAC trains the policies and critics, and Time Indexed Disagreement Exploration (TIDE) chooses which candidate action to execute.

A diverse ensemble helps only if the agent acts on its diversity. At each step, every policy proposes candidate actions, and the agent must choose one to execute. If it simply picks the candidate with the highest critic score, it will rarely try an untested alternative, because the critics have not yet seen that alternative pay off. We want candidates that the critics consider promising and that the policies disagree about. Standard ensemble methods do not capture this. Upper Confidence Bound (UCB) rewards disagreement among critics (Lee et al., 2021), and Thompson sampling follows a single policy for a whole episode (Osband et al., 2016). Neither accounts for how generative policies produce actions. Policies may agree at one generation step and disagree at another, and it is not clear in advance which step matters. We therefore gather the candidates from all K policies and score each one by its average critic score plus a bonus for how much the actions predicted by policies differ, measured at several generation steps. We call this approach Time-Indexed Disagreement Exploration (TIDE). Measuring at a single generation step instead lowers performance on most tasks we test. Policy disagreement does not always reflect real uncertainty, so TIDE applies the bonus only if that policy disagreement agrees with critic disagreement. Otherwise, it picks the candidate with the highest critic score.

Our contributions are summarized as follows:

- We identify collapse in ensembles of generative policies that share a critic, and show theoretically and empirically that it follows from how the policies are trained.
- We propose ESAC, which gives each policy its own pair of critics, so that each policy learns from a different training signal while the ensemble needs only K critics.
- We introduce TIDE, which chooses actions that the critics score highly and the policies disagree about, measuring disagreement across generation steps and trusting it only when it agrees with critic disagreement.

To the best of our knowledge, this work is the first to use an ensemble of generative policies, trained online, for exploration.

2 PRELIMINARIES

We consider a discounted infinite-horizon Markov decision process $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ with continuous states and actions, transition kernel P, reward r, and discount $\gamma \in [0, 1)$. Let d_a be the action dimension. The objective is expected discounted return, $J(\pi) = \mathbb{E}_\pi[\sum_{t\geq 0} \gamma^t r(s_t, a_t)]$. Although the learning objective uses an infinite-horizon formulation, the experiments use finite episodes that end at environment termination or a task time limit. MuJoCo evaluation reports undiscounted episodic return. An actor proposes actions and a critic estimates their long-term value. We build on the candidate-based policy optimization of Soft Diffusion Actor-Critic (SDAC) (Ma et al., 2025), using a temperature to control action weighting and exploration noise. Critics are trained through Temporal-Difference (TD) regression with slowly updated target networks.

Diffusion-based policies: SDAC (Ma et al., 2025) parametrizes the policy as a Denoising Diffusion Probabilistic Model (DDPM) (Ho et al., 2020, Nichol & Dhariwal, 2021). Candidates are generated from Gaussian noise $x_T \sim \mathcal{N}(0, I)$ by iterating a learned denoiser ϵ_θ over T steps,

$$x_{k-1} = \frac{1}{\sqrt{\alpha_k}} \left(x_k - \frac{1 - \alpha_k}{\sqrt{1 - \bar{\alpha}_k}} \epsilon_\theta(x_k, k, s) \right) + \sigma_k z, \quad z \sim \mathcal{N}(0, I). \tag{1}$$

Here k indexes reverse steps, $\alpha_k \in (0, 1)$ is the signal-retention factor, $\bar{\alpha}_k = \prod_{i=1}^{k} \alpha_i$, and σ_k is the reverse-noise standard deviation. For a clean action a_0, the forward process is $x_k = \sqrt{\bar{\alpha}_k} a_0 +$

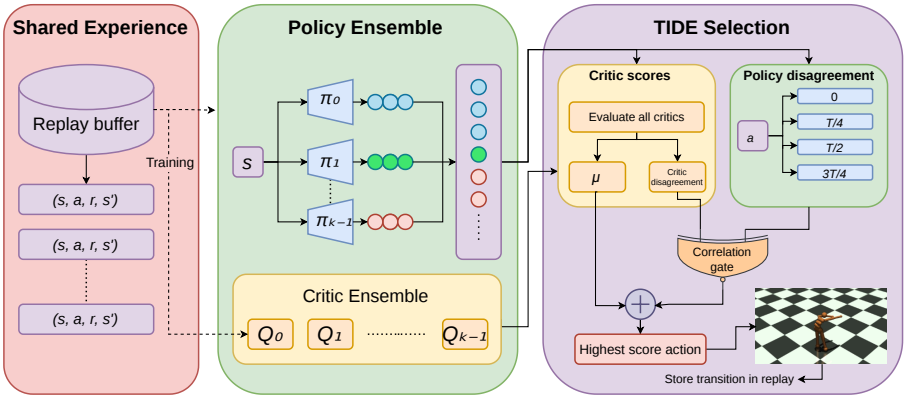

Figure 1: ESAC training and pooled action selection. Critic-guided updates train the generative policies, and TIDE scores the population's proposals; dataset symbols denote scalar actor-loss multipliers.

$\sqrt{1-\bar{\alpha}_k}\epsilon$, with $\epsilon \sim \mathcal{N}(0, I)$. The deterministic Denoising Diffusion Implicit Model (DDIM) sampler provides an alternative reverse update (Song et al., 2020a).

Q-weighted regression favors actions with larger critic values. On a finite candidate set, let $w_i = \mathrm{softmax}_i(\psi Q(s, \hat{a}_i)/\tau_{\mathrm{ent}})$, where $\psi > 0$ scales Q-values and $\tau_{\mathrm{ent}} > 0$ is the weighting temperature. The softmax normalizes over candidate index i. We write the regression objective as

$$\mathcal{L}(\theta) = \sum_i w_i \| f_\theta(x_i, t_i, s) - u_i \|^2. \tag{2}$$

Here f_θ is the policy predictor with parameters θ, and x_i, t_i, u_i are its noised input, time label, and regression target associated with the proposal construction. SDAC uses a reverse-sampling construction to obtain these quantities without drawing from the target policy (Ma et al., 2025). Appendix G specifies the construction used in our implementation.

Flow-matching policies: Conditional flow matching (Lipman et al., 2022, Tong et al., 2023) replaces the diffusion noise schedule with straight-line interpolation, $x_t = (1-t)\, a_0 + t\, \epsilon$ for $t \in [0, 1]$, and trains a velocity network v_θ to regress the conditional velocity $v^\star = \epsilon - a_0$:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, \epsilon\sim\mathcal{N}(0,I)} \left[\| v_\theta(x_t, t, s) - (\epsilon - a_0) \|^2 \right]. \tag{3}$$

Sampling uses Euler steps $x_{t-\Delta t} = x_t - v_\theta(x_t, t, s)\, \Delta t$ for the ordinary differential equation (ODE) from $t = 1$ to $t = 0$, and is deterministic given the initial noise.

3 Methodology

An ensemble of generative policies helps exploration only if two conditions hold. First, the policies must propose different actions; otherwise, the ensemble offers no more alternatives than a single policy. Second, the agent needs a rule for choosing which proposed action to execute; otherwise the alternatives are never tried. Neither condition holds by default. Policies trained against a shared critic learn to imitate the same weighted candidates and collapse to the same behavior, even when they start from different random initializations (Section 1). And even a diverse ensemble does not help if the agent always executes the action its critics already score highest, because untested alternatives rarely receive high scores.

ESAC addresses the first condition during training and the second during interaction. During training, each policy gets a different training signal so that the policies stay distinct (Section 3.1). Each diffusion policy weights the same candidate actions by the scores of its own pair of critics, so each policy learns to imitate a different weighting of those candidates. Flow-matching policies instead all learn from a single high-value action selected by one critic pair; Section 4 discusses why these policies nonetheless remain distinct. During interaction, ESAC gathers the candidate actions of all policies and uses TIDE to choose one (Section 3.2). TIDE favors candidates that the critics score highly and that the policies disagree about, and it trusts policy disagreement only when it agrees with critic disagreement. Algorithm 1 summarizes the full training loop.

3.1 ESAC: POLICY POPULATIONS AND CRITIC-GUIDED TRAINING

ESAC maintains K separately initialized policies and K critics, indexed from 0 to $K-1$. Policy and critic parameters are θ_k and ϕ_k, respectively; $\bar{\phi}_k$ denotes target-critic parameters. They share replay data and a common target-action distribution. Circular pairs $S_k = \{k, (k+1) \bmod K\}$ provide head-specific value estimates for diffusion regression. Each critic participates in two pairs, so the construction uses K critics.

One fixed policy head, indexed by 0 for notation, supplies the actions used for actor training and Bellman targets. Given a state, it generates N proposals b_i. A critic pair selects the highest-valued proposal, which is clipped to the action bounds and perturbed by Gaussian noise,

$$i^* = \arg\max_{i \le N} \min_{l \in S_0} Q_{\phi_l}(s, b_i), \qquad a^\dagger(s) = \mathrm{clip}(b_{i^*}, -1, 1) + \sigma_{\mathrm{act}} \tau_{\mathrm{ent}} \xi, \quad \xi \sim \mathcal{N}(0, I). \tag{4}$$

$a^\dagger(s)$ denotes the selected, perturbed training action. $\sigma_{\mathrm{act}} > 0$ is the fixed coefficient scaling the Gaussian noise. The training distribution therefore favors actions with higher estimated value before either regression loss is formed. Both variants inherit this candidate-selection step from the SDAC implementation (Ma et al., 2025). The selected training action is held fixed during the actor update; its construction has no final clipping after noise (Appendix G.2).

All critics regress to the common target

$$y = c_r r + \gamma(1-d) \min_{l \in \mathcal{M}} Q_{\bar{\phi}_l}(s', a^\dagger(s')), \tag{5}$$

where c_r is the fixed reward scale, s' is the next state, d indicates termination, and $\mathcal{M}$ is a random size-M subset of target critics (Chen et al., 2021). This estimates continuation under the rule for selecting training actions. The interaction policy can draw from all heads, making training off-policy. Target parameters follow an exponential moving average.

For the diffusion actor, the selected training action is noised once per state. Reverse-noise draws u_i then define N_{mc} candidate actions $\hat{a}_i$ at a shared input x_ℓ. Head k uses

$$\begin{aligned} w_{ki} &= \mathrm{softmax}_i\left(\frac{\psi \min_{l \in S_k} Q_{\phi_l}(s, \hat{a}_i)}{\tau_{\mathrm{ent}}}\right), \\ \mathcal{L}_k^{\mathrm{Diff}} &= \frac{\bar{m}_k}{N_{\mathrm{mc}} d_a} \mathbb{E}_{\mathcal{B},\ell}\left[\sum_i w_{ki} \|\epsilon_{\theta_k}(x_\ell, \ell, s) - u_i\|^2\right]. \end{aligned} \tag{6}$$

The fixed factor $\psi > 0$ scales Q-values independently of K. $\mathcal{B}$ is a replay minibatch of size B and ℓ a sampled time label. Expectations include proposal noises; the scalar multiplier $\bar{m}_k$ is defined below. The candidate-dependent targets allow the critic weights to change the regression target seen by each head. Appendix G gives the reverse construction and numerical settings.

For the flow actor, the selected action $a^\dagger$ defines the interpolation $x_t = (1-t)a^\dagger + t\epsilon$ and target $\epsilon - a^\dagger$. Its objective is

$$\mathcal{L}_k^{\mathrm{FM}} = \frac{c_{\mathrm{FM}} \bar{m}_k}{d_a} \mathbb{E}_{\mathcal{B},t,\epsilon}\left[\|v_{\theta_k}(x_t, t, s) - (\epsilon - a^\dagger)\|^2\right]. \tag{7}$$

Here $c_{\mathrm{FM}} > 0$ is a fixed loss scale. Each velocity network is trained on critic-selected actions through Eq. 7. The critic guides the objective through the selected training action $a^\dagger$.

The population loss averages $\mathcal{L}_k$ across heads. Each multiplier $\bar{m}_k$ is the minibatch mean of Bernoulli draws with probability p, providing random scalar loss scaling. Appendix A.1 gives the small head-0 adjustment when all heads receive a zero draw for a sample. The temperature controls diffusion regression weights and Gaussian action noise through a scalar noise-entropy update with a fixed target. Appendix G.4 describes its schedule and stationary value.

3.2 TIME-INDEXED DISAGREEMENT EXPLORATION

Section 3.1 keeps the policies distinct; TIDE decides how the agent uses that diversity when it collects data. If the agent simply executed the candidate with the highest critic score, it would rarely try an untested alternative, because the critics have not yet seen that alternative pay off. TIDE therefore adds a bonus for candidates the policies disagree about, since disagreement marks actions where trying them is most informative. TIDE is used only during data collection. At evaluation, the agent still pools candidates from all policies but executes the one with the highest average critic score, without the disagreement bonus or added noise, so TIDE improves performance through the data it collects.

Algorithm 1 ESAC training and action selection

```
Initialize K policies, K critics, target networks, temperature, and replay 𝒟.
Populate 𝒟 with random-action transitions.
for each interaction and update round do
    Pool N action proposals from each policy head.
    Select actions by Eq. 9, add Gaussian noise, and execute.
    Store transitions in 𝒟; sample minibatch ℬ and critic subset ℳ.
    Form y using Eqs. 4 and 5; update all critics.
    if an actor update is due then
        Generate training actions a†(s) and loss multipliers m̄_k.
        Update policies using K^{-1} Σ_k ℒ_k: Eq. 6 (diffusion) or Eq. 7 (flow).
        Update target parameters by exponential moving average.
    end if
    Update τ_ent when due.
end for
```

Each head proposes N actions, giving candidates $\{a_j\}_{j=1}^{KN}$ at state s; j indexes the pooled actions. For candidate a_j, TIDE evaluates every policy network at a fixed set of time labels $\mathcal{T}$. Its deployed score averages coordinate-wise population standard deviations across the finite ensemble,

$$D_j = \frac{1}{|\mathcal{T}| d_a} \sum_{\ell \in \mathcal{T}} \sum_{d=1}^{d_a} \sqrt{\frac{1}{K} \sum_{k=0}^{K-1} \left(f_{\theta_k}(a_j, \ell, s)_d - \bar{f}_{j\ell d} \right)^2}. \tag{8}$$

Here f_{θ_k} denotes the denoiser ϵ_{θ_k} or velocity predictor v_{θ_k}, d indexes action coordinates, and $\bar{f}_{j\ell d} = K^{-1} \sum_k f_{\theta_k}(a_j, \ell, s)_d$ is the head mean. The same candidate a_j is evaluated at every label, making D_j a measure of predictor disagreement on fixed action inputs. These probes differ from both the noised regression inputs and the states traversed during generation. Appendix A.3 gives finite-ensemble norm bounds and a separate coupled-trajectory result; the latter does not establish that the score used for action selection measures epistemic uncertainty.

Let $\bar{Q}(s,a) = K^{-1} \sum_k Q_{\phi_k}(s,a)$ and $U_j = \mathrm{std}_k\, Q_{\phi_k}(s, a_j)$, using variance divisor K. For the gate, j ranges over all candidate-state pairs in the current vectorized decision. $\bar{D}$ and $\bar{U}$ are their means. With $d_j = D_j - \bar{D}$ and $u_j = U_j - \bar{U}$, the gate and selection rule at interaction round t are

$$\rho_t = \frac{\sum_j d_j u_j}{\sqrt{\sum_j d_j^2 \sum_j u_j^2} + \epsilon_g}, \quad \beta_t = \beta_0 \max(0, \rho_t), \quad a^*(s) = \arg\max_{a_j} \left[\bar{Q}(s, a_j) + \beta_t D_j \right]. \tag{9}$$

Here $\epsilon_g > 0$ stabilizes the denominator and $\beta_0 > 0$ sets the base bonus scale. The maximization is over candidates at state s. It is stateless, with no sliding window. Scores use critic values and policy disagreement in their native scales. Nonpositive correlation suppresses the bonus, leaving ensemble-mean candidate selection. Training then adds Gaussian action noise, while evaluation omits the TIDE bonus and added noise. The UCB comparator instead adds $\beta\, \mathrm{std}_k\, Q_k$ to the ensemble mean, with $\beta > 0$, while Thompson selection commits to a policy head for an episode.

TIDE computes a single gate for all parallel environments at once, correlating policy and critic disagreement across every candidate from every environment. Pooling gives the correlation more candidates to estimate from, but it also means the bonus at one state depends on which other states the agent is acting in at the same time. A gate computed separately for each state, or averaged over time, could behave differently. The gate is best understood as a heuristic rather than a calibrated test of uncertainty. All critics train on the same data and the same temporal-difference target, so their disagreement does not directly measure how wrong their value estimates are. A positive correlation therefore tells TIDE that policy and critic disagreement point in the same direction, not that either one is accurate. Algorithm 1 summarizes how ESAC alternates between collecting data and learning from it. During interaction, the agent uses TIDE to choose among the candidate actions of all policies. When learning, the critics train on a shared temporal-difference target, and each policy trains with its diffusion or flow-matching objective. During evaluation, the agent executes the candidate with the highest average critic score, without the TIDE bonus or added action noise.

4 Shared objectives and policy diversity

Section 1 claimed that policies trained against a shared critic collapse, and that giving each policy its own pair of critics prevents this. This section explains why both claims hold. We first show that when policies regress toward the same weighted targets, every gradient step shrinks the distance between their predictions, so different random initializations cannot keep them apart. We then show

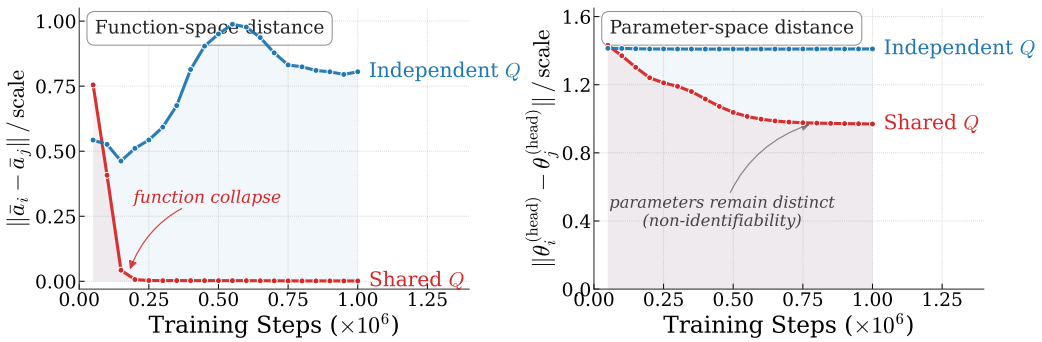

Figure 2: Shared-critic heads can produce nearly identical predictions while their parameters remain separated. Prediction distances (left) and final-layer parameter distances (right) are measured on Ant-v4 ($K = 5$), with each distance normalized by the corresponding mean head norm. Red denotes shared critics and blue denotes independent critics.

that different critic pairs change the regression target itself, which gives each policy a different point to converge toward. To isolate the effect of the training objective from the details of neural network training, the analysis assumes fixed targets and predictions that can be adjusted freely.

Let f collect freely adjustable predictions at distinct inputs, and let $J(f) = \sum_i w_i \|f_i - u_i\|^2$ for fixed targets and positive weights. With I_{d_a} the action-space identity and $W = \text{diag}(w_i) \otimes I_{d_a}$, one gradient step of size η (denoted by $+$) gives

$$f_k^+ - f_j^+ = (I - 2\eta W)(f_k - f_j), \qquad \|f_k^+ - f_j^+\| \le \varrho \|f_k - f_j\|, \tag{10}$$

where $\varrho = \max_i |1 - 2\eta w_i| < 1$ for $0 < \eta < 1/\max_i w_i$. Repeated inputs share one output coordinate and are grouped before applying this result. Appendix A.1 gives the proof and the output-space Polyak–Łojasiewicz condition.

The diffusion update illustrates how different critic estimates can change a grouped target. For a shared input and normalized weights, let $J_k(f) = \sum_i w_{ki} \|f - u_i\|^2$ and $\mu_k = \sum_i w_{ki} u_i$. Then

$$\nabla_f J_k(f) = 2(f - \mu_k), \qquad \nabla_f J_k(f) - \nabla_f J_j(f) = -2(\mu_k - \mu_j). \tag{11}$$

Circular pairing permits distinct μ_k when the critics rank candidates differently. Appendix A.2 states the exact conditions. These calculations concern fixed regression in output space. Neural updates also depend on each head's Jacobian and optimizer state, while targets evolve during training.

Figure 2 shows that different network parameters do not necessarily produce different predictions. In the left panel, the shared-critic heads (red) become nearly identical on the probe inputs as their prediction distance approaches zero. Their parameter distance nevertheless remains nonzero in the right panel. The independent-critic configuration (blue) retains larger prediction differences during later training. The takeaway is that parameter separation alone does not establish prediction diversity. This agrees with the shared-objective analysis, which describes convergence of outputs without requiring the network weights to become identical. Figures 4 and 5 in Appendix D provide additional diagnostics.

5 Experiments

5.1 Setup

The ESAC experiments use the MuJoCo (Todorov et al., 2012) tasks HalfCheetah, Hopper, Walker2d, Ant, Swimmer, Pusher, Reacher, and Humanoid. Training uses 30k random-action transitions followed by 1M interaction steps, or 5M for Humanoid. ESAC and the controls using the SDAC implementation use five vectorized environments, one critic update per five transitions (UTD 0.2), a batch size of 256, and policy and target updates every second critic update. The ESAC settings include $T = 20$ sampling steps, $K = 5$ ensemble heads, $N = 32$ candidates per head, and a size-2 target-critic subset. Table 4 in Appendix G gives the remaining hyperparameters and distinguishes this schedule from algorithm-specific baseline updates.

Table 1 summarizes these scores as mean $\pm$ standard deviation across five seeds. ESAC checkpoints are evaluated over 20 episodes every 50k training steps (250k for Humanoid), using mean undiscounted episode return. The action with the largest ensemble-mean Q-value is selected from 160 proposals across the five heads. Sampling seeds are determined by the observation, and the TIDE bonus and added action noise are disabled. Ensemble training uses more network evaluations and time, motivating the update-budget comparisons in Appendix B. Green cells and boldface mark the

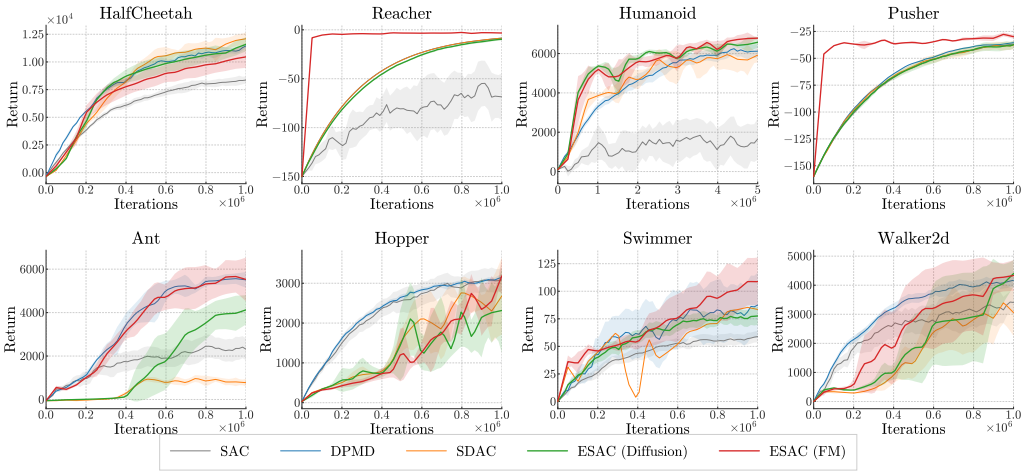

Figure 3: Training learning curves on five MuJoCo locomotion tasks trained for 1M steps (200k iterations across 5 vectorized environments) for top 5 baseline.

largest reported mean, including ties, without a significance claim, pale blue rows identify ESAC. Appendix G.6 gives the evaluation protocol.

5.2 Main Results

The complete ESAC configurations improve over the tested SDAC implementation on Ant, Walker2d, and Humanoid with both generative parameterizations (Table 1). Diffusion Policy Mirror Descent (DPMD) has nearby reported means on these tasks, although the differences are small relative to seed variability and do not establish equivalence. On Humanoid, ESAC and DPMD also have less variation across seeds than SAC and DACER. These comparisons concern the full ensemble system at a fixed interaction budget, including its additional proposals and computation.

In the inspected Swimmer seed, flow heads separate into slow and fast gait-frequency groups, with each head exceeding the single-policy rollout return (Figure 6, Appendix E.1). Flow matching also has the stronger ensemble return on Swimmer, while diffusion is below SDAC. This illustrates behavioral diversity in one trained population.

HalfCheetah favors single-policy diffusion methods, and its single-policy flow control slightly exceeds the flow ensemble. Hopper favors DPMD and several conventional baselines. Reacher gives similar means across several methods, with larger ESAC variability, while TD3 leads on Pusher. The ensemble benefit is therefore task dependent. Figure 7 provides complementary learned-value diagnostics. The following controls examine the policy architecture, update budget, and population (Table 2). Figure 3 is illustrative and is excluded from conclusions about measured convergence rates.

5.3 Controlled Comparisons

The single-policy flow control (Table 2a) tests whether flow matching alone accounts for the ensemble's improvement (Appendix B). The largest difference is on Ant, where the five-head system more than doubles the single-policy mean. Walker2d and Hopper have smaller increases, while HalfCheetah slightly favors the single policy. These comparisons evaluate the combined effect of a larger policy population, more candidate proposals, and additional critic computation under the same interaction schedule.

The update-rate controls (Table 2b) show that the training budget is consequential. On Ant, SAC at critic UTD 1.0 reaches 5477 ± 625, close to ESAC (FM)'s 5646 ± 1001. Increased updates therefore provide a strong alternative to ensembling in this comparison. SDAC responds differently, dropping to 21 ± 9 at the higher rate under the tested settings. This result motivates examining its optimization separately from the benefit of a policy population. Equal update rounds also involve different numbers of trained networks across methods.

Widening individual networks produces mixed gains across tasks, showing that additional parameters alone do not consistently improve returns (Table 5). Widened SAC has more parameters than ESAC and exceeds the diffusion ensemble on Ant, while remaining below the flow ensemble in mean. Widening only SDAC's actor gives little change. These outcomes support evaluating how parameters are used, through larger individual networks or multiple proposal distributions, alongside the interaction and update budgets.

Table 1: ESAC (FM) has the largest reported mean on five of eight tasks, while the strongest method varies across environments. MuJoCo returns are reported as mean ± standard deviation across five seeds. Green bold cells mark the largest mean per task; Section 5.1 describes the evaluation protocol.

Method	HalfCheetah	Reacher	Humanoid	Pusher
PPO (Schulman et al., 2017)	4715 ± 810	-7.56 ± 10.68	1015 ± 280	-27.52 ± 3.40
TD3 (Fujimoto et al., 2018)	8345 ± 645	-3.19 ± 0.04	5705 ± 405	$\mathbf{-25.23 \pm 0.96}$
SAC (Haarnoja et al., 2018)	8745 ± 415	-61.58 ± 58.11	2945 ± 2415	-34.28 ± 1.16
QSM (Psenka et al., 2023)	10495 ± 510	-4.45 ± 0.45	5520 ± 480	-83.18 ± 2.20
DIPO (Yang et al., 2023)	9285 ± 720	-3.28 ± 0.03	4790 ± 985	-32.11 ± 0.44
DACER (Wang et al., 2024)	11055 ± 285	-3.12 ± 0.07	2880 ± 3210	-31.02 ± 0.18
QVPO (Ding et al., 2024)	7195 ± 1015	-35.89 ± 13.25	445 ± 95	-134.21 ± 1.08
DPPO (Ren et al., 2025)	1245 ± 415	-5.62 ± 1.67	510 ± 75	-95.45 ± 11.87
DPMD (Ma et al., 2025)	11750 ± 670	-3.18 ± 0.08	6710 ± 525	-30.18 ± 0.33
SDAC (Ma et al., 2025)	$\mathbf{11949 \pm 659}$	-3.43 ± 0.41	5766 ± 386	-32.28 ± 4.98
ESAC (Diff)	11310 ± 728	-3.84 ± 1.96	6493 ± 672	-31.52 ± 5.88
ESAC (FM)	10328 ± 1119	$\mathbf{-3.03 \pm 0.87}$	$\mathbf{6773 \pm 510}$	-28.90 ± 4.53

Method	Ant	Hopper	Swimmer	Walker2d
PPO (Schulman et al., 2017)	3320 ± 945	3155 ± 220	84.5 ± 12.4	4220 ± 740
TD3 (Fujimoto et al., 2018)	3690 ± 1180	1855 ± 985	71.9 ± 15.3	2640 ± 1190
SAC (Haarnoja et al., 2018)	2480 ± 820	3110 ± 345	63.5 ± 10.2	3110 ± 950
QSM (Psenka et al., 2023)	870 ± 195	2715 ± 510	57.0 ± 7.7	2480 ± 805
DIPO (Yang et al., 2023)	1020 ± 145	1245 ± 690	46.7 ± 2.9	2050 ± 1380
DACER (Wang et al., 2024)	4080 ± 615	3145 ± 145	103.0 ± 45.8	3275 ± 1640
QVPO (Ding et al., 2024)	745 ± 290	2810 ± 555	53.4 ± 5.0	2415 ± 1095
DPPO (Ren et al., 2025)	95 ± 35	2245 ± 580	106.1 ± 6.5	1085 ± 720
DPMD (Ma et al., 2025)	5520 ± 245	$\mathbf{3220 \pm 85}$	84.33 ± 34.56	4280 ± 320
SDAC (Ma et al., 2025)	798 ± 190	2472 ± 993	83.84 ± 16.9	3176 ± 782
ESAC (Diff)	4043 ± 617	2284 ± 1147	77.05 ± 16.04	4172 ± 483
ESAC (FM)	$\mathbf{5646 \pm 1001}$	2759 ± 437	$\mathbf{107.83 \pm 23.82}$	$\mathbf{4305 \pm 529}$

5.4 Exploration rules and diagnostics

Among the tested selection rules, TIDE has the highest mean on Ant, Walker2d, and Hopper, while UCB slightly leads on HalfCheetah (Table 3). The larger differences occur on Ant and Walker2d. Hopper's Thompson, UCB, and TIDE results are closer, indicating that much of its performance is shared across ensemble selection rules. The existing comparison establishes task dependence within this family of exploration methods.

The benefits of the correlation gate and multiple time labels are task dependent (Table 3). Removing either component reduces Hopper's mean, and the full rule also has the highest mean on Walker2d. On Ant the component variants remain closer to full TIDE, while HalfCheetah gains little from either component.

Prediction and critic dispersion persist over the measured training interval, and the inspected Swimmer seed exhibits distinct gait frequencies. Statewise gate probes show positive late correlations on Ant and Walker2d, near-zero correlation on HalfCheetah, and negative correlation on peg insertion (Table 7). These probes differ from the pooled training gate and do not measure value-error calibration.

5.5 Computational cost and manipulation

The ensemble's return gains come with substantially higher computational cost. At equal interactions, ESAC (FM) takes approximately three times SDAC's training time on Ant, 7.9 versus 2.6 hours for 1M steps. Humanoid takes 38.6 versus 14.3 hours for 5M steps. These measurements use two concurrent runs per RTX A6000. ESAC has 1.45M parameters against SDAC's 0.43M and pools 160 candidates against 32 (Table 6). The relevant trade-off is higher computation for a larger set of learned proposals at a fixed interaction budget.

On dense-reward MetaWorld tasks (Table 8; Figures 12 and 11), the ensemble adds little where SDAC already approaches full success. Drawer closing and button pressing are nearly saturated, and door opening also reaches high success with either method. Reach has substantial seed variability. Peg insertion favors SDAC, with a mean success rate roughly twelve percentage points above both ESAC variants. The per-head diagnostics show limited useful behavioral separation in the inspected peg-insertion checkpoint (Figures 8, 9, and 10). These results locate the present benefit primarily in locomotion, and motivate evaluating reward density and behavioral diversity as separate experimental factors.

6 Related work

Diffusion Policy established diffusion models for expressive behavior cloning in robotics (Chi et al., 2025). Online policy improvement includes Q-score matching (Psenka et al., 2023), action-gradient improvement in Diffusion Policy (DIPO) (Yang et al., 2023), entropy regulation in Diffusion Actor-Critic with Entropy Regulator (DACER) (Wang et al., 2024), reweighted score matching in SDAC and DPMD (Ma et al., 2025), and Q-weighted Variational Optimization in QVPO (Ding et al., 2024). MaxEntDP explicitly targets maximum-entropy RL with a diffusion policy (Dong et al., 2025). TIDE instead changes interaction-time candidate selection through disagreement among policy predictors, so a matched online comparison with MaxEntDP would test a different exploration mechanism.

Flow Policy Optimization (FPO) (McAllister et al., 2026) and Diffusion Policy Policy Optimization (DPPO) (Ren et al., 2025) connect generative policies to on-policy optimization. ReinFlow adds learnable noise for likelihood-based flow-policy fine-tuning (Zhang et al., 2026), while DSRL optimizes a pretrained diffusion policy's latent-noise inputs with its generator fixed (Wagenmaker et al., 2025). ESAC-FM learns a population from scratch through regression on critic-selected actions and selects proposals in action space.

Offline diffusion methods include diffusion Q-learning (Wang et al., 2022), Implicit Diffusion Q-Learning (IDQL) (Hansen-Estruch et al., 2023), and state-reconstruction objectives (Ada et al., 2024). Zhang et al. (2024) combine entropy-regularized diffusion policies with pessimistic Q-ensembles. Khan et al. (2026) use Q-weighted flow matching for offline discrete actions. ESAC studies the complementary setting of online continuous control with policy populations.

Deep Diffusion Policy Gradient (DDiffPG) (Li et al., 2024) learns multiple behaviors through intrinsic motivation, clustering, and mode-specific critics. ESAC instead indexes separately trained policies by population member and pools their proposals. DDiffPG remains relevant under matched native dense rewards, while paired dense and sparse conditions would additionally test reward discovery.

Diffusion Filtered Explorations via Ensembles (DF-ExpEnse) (Luo et al., 2026) filters diffusion candidates with critic-ensemble estimates of quality and exploration interest during fine-tuning, including coordinated fleet exploration. ESAC also uses candidate filtering and critic ensembles. TIDE's specific distinction is policy-predictor disagreement across time labels, gated by correlation with critic dispersion. A direct comparison must account for DF-ExpEnse's pretrained-policy setting.

Ensemble exploration builds on Thompson sampling (William et al., 1933, Daniel et al., 2018), bootstrapping (Breiman, 1996, Hesterberg, 2011), and randomized prior functions (Osband et al., 2018). Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning (SUNRISE) (Lee et al., 2021) and optimistic actor-critic (Ciosek et al., 2019) guide continuous actions with value uncertainty. ESAC combines ensemble values with generative proposals and a policy-disagreement bonus. Curiosity (Pathak et al., 2017), random network distillation (Burda et al., 2018), visitation bonuses (Bellemare et al., 2016), goal relabeling (Andrychowicz et al., 2017), and archive-based exploration (Ecoffet et al., 2021) provide complementary mechanisms for sparse reward discovery.

7 Conclusion and limitations

We asked whether ensembles, which help Gaussian policies explore, can do the same for generative policies trained online. We found that the answer depends on how the ensemble is trained. When all policies regress toward targets weighted by a shared critic, their predictions converge and the ensemble behaves like a single policy. ESAC keeps the policies apart by scoring each diffusion policy's candidates with its own pair of critics, and TIDE uses the remaining disagreement to choose which proposed action to execute. On locomotion tasks, this combination improves return over matched single-policy controls, most clearly on Ant. On Swimmer, the policies settle into distinct gaits that each outperform the single policy.

The evidence is limited to dense rewards, including native shaped MetaWorld rewards. It does not establish reward discovery before successful trajectories are observed. Additional policies cost approximately three times SDAC's training time, and increased-update SAC is a strong alternative on Ant. The gains are regime-dependent, ESAC plateaus on unimodal manipulation skills where coverage helps less than sharpness, pointing to multi-solution manipulation and adaptive K as natural future work, alongside extending ESAC to offline pretraining and offline-to-online finetuning where Q-uncertainty behaves differently.

DISCLOSURE OF AI ASSISTANCE

Generative AI tools were used to support three aspects of manuscript preparation. For writing assistance, they helped improve grammar, sentence structure, clarity, and consistency of terminology. For literature retrieval and discovery, they assisted in formulating search queries and identifying potentially relevant publications for the related-work discussion. For drafting, they helped develop preliminary text for selected manuscript sections using the author's research descriptions, notes, and existing material. These drafts served as starting points for further editing and refinement during manuscript preparation.

REFERENCES

Suzan Ece Ada, Erhan Oztop, and Emre Ugur. Diffusion policies for out-of-distribution generalization in offline reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3116–3123, 2024.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

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.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019.

J Russo Daniel, Van Roy Benjamin, Kazerouni Abbas, Osband Ian, and Wen Zheng. A tutorial on thompson sampling. *Foundations and trends® in machine learning*, 11(1):1–99, 2018.

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.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

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.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 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.

Tim Hesterberg. Bootstrap. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(6):497–526, 2011.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 795–811. Springer, 2016.

Fairoz Nower Khan, Nabuat Zaman Nahim, Ruiquan Huang, Haibo Yang, and Peizhong Ju. Flow matching for offline reinforcement learning with discrete actions. *arXiv preprint arXiv:2602.06138*, 2026.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International conference on machine learning*, pp. 6131–6141. PMLR, 2021.

Zechu Li, Rickmer Krohn, Tao Chen, Anurag Ajay, Pulkit Agrawal, and Georgia Chalvatzaki. Learning multimodal behaviors from scratch with diffusion policy gradient. *Advances in Neural Information Processing Systems*, 37:38456–38479, 2024.

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.

Calvin Luo, Chen Sun, and Shuran Song. Df-expense: Diffusion filtered exploration for sample efficient finetuning. *arXiv preprint arXiv:2606.19656*, 2026.

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

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Reginald McLean, Evangelos Chatzaroulas, Luc McCutcheon, Frank Röder, Tianhe Yu, Zhanpeng He, KR Zentner, Ryan Julian, Jordan Terry, Isaac Woungang, et al. Meta-world+: An improved, standardized, rl benchmark. *Advances in Neural Information Processing Systems*, 38, 2026.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PmLR, 2021.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

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 Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

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

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

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

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.

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.

R Thompson William et al. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B Schön, and Per Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in neural information processing systems*, 37:98871–98897, 2024.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 106282–106319, 2026.

A PROOFS AND SUPPORTING ANALYSIS

All vector norms below are Euclidean unless specified otherwise. Candidate values, regression targets, and importance weights are fixed during each analyzed regression update. Ensemble dispersion refers to the finite collection of heads, without a posterior-sampling assumption. We state the relationship to neural training after the relevant derivations.

A.1 SHARED-CRITIC REGRESSION

Let $f = (f_1, \dots, f_n) \in \mathbb{R}^{nd_a}$ be the unconstrained vector of predictions on a fixed set of regression inputs, with targets $u_i \in \mathbb{R}^{d_a}$ and fixed weights $w_i > 0$. Set $w_{\min} = \min_i w_i$ and $w_{\max} = \max_i w_i$. The input-output coordinates are treated as free variables. This is an idealization of training a neural network on that support, not a claim about its parameterization. Write

$$J(f) = \sum_{i=1}^{n} w_i \|f_i - u_i\|^2, \qquad W = \operatorname{diag}(w_1, \dots, w_n) \otimes I_{d_a}. \tag{12}$$

Proof. Differentiating the fixed quadratic gives

$$\nabla J(f) = 2W(f - u), \qquad \nabla^2 J(f) = 2W.$$

The gradient update and the difference between two heads therefore satisfy

$$\begin{aligned} f_k^+ &= f_k - 2\eta W(f_k - u), \\ f_k^+ - f_j^+ &= (f_k - f_j) - 2\eta W\big[(f_k - u) - (f_j - u)\big] \\ &= (I - 2\eta W)(f_k - f_j). \end{aligned}$$

Since $0 < \eta < 1/w_{\max}$, the diagonal matrix has spectral norm

$$\|I - 2\eta W\|_2 = \max_i |1 - 2\eta w_i| = \varrho < 1.$$

Iteration yields both pairwise contraction and convergence to the target:

$$\begin{aligned} \|f_k^{(t)} - f_j^{(t)}\| &\le \varrho^t \|f_k^{(0)} - f_j^{(0)}\|, \\ \|f_k^{(t)} - u\| &\le \varrho^t \|f_k^{(0)} - u\| \longrightarrow 0. \end{aligned}$$

Positive definiteness of W makes u the unique minimizer. □

For a fixed head-specific multiplier $a_k > 0$, the corresponding error update is

$$f_k^+ - u = (I - 2\eta a_k W)(f_k - u), \qquad 0 < \eta < \frac{1}{a_k w_{\max}}.$$

This also converges to u, although unequal multipliers need not satisfy the common pairwise recurrence.

The Hessian gives smoothness constant $2w_{\max}$ and strong-convexity constant $2w_{\min}$. In particular,

$$\begin{aligned} \tfrac{1}{2}\|\nabla J(f)\|^2 &= 2\sum_i w_i^2 \|f_i - u_i\|^2 \\ &\ge 2w_{\min} \sum_i w_i \|f_i - u_i\|^2 = 2w_{\min}\big(J(f) - J(u)\big). \end{aligned} \tag{13}$$

Thus a Polyak–Łojasiewicz inequality holds for this finite output-space objective (Karimi et al., 2016). These constants are conditional on the fixed batch. Positive softmax weights do not provide a uniform lower bound across changing batches or temperatures.

For independent Bernoulli masks with $0 < p \le 1$, multiplying a head's loss by $\bar{m}_k = B^{-1} \sum_{b=1}^{B} m_{kb}$ gives

$$\mathbb{E}_m[\nabla_\theta(\bar{m}_k J(f_\theta))] = p \nabla_\theta J(f_\theta), \qquad \operatorname{Var}(\bar{m}_k) = p(1-p)/B. \tag{14}$$

This equality requires the mask to be independent of the fixed batch and parameters. A mask applied separately to individual losses also preserves the expected objective under the same independence

assumption, but has different update noise. Equality of expected objectives does not establish equality of stochastic trajectories.

The implementation also assigns an otherwise unselected sample to head 0. For independent raw draws m_{kb}, the repaired bit is

$$m'_{0b} = m_{0b} + \prod_{l=0}^{K-1}(1 - m_{lb}), \qquad a_k := \mathbb{E}[\bar{m}'_k] = \begin{cases} p + (1-p)^K, & k = 0, \\ p, & k > 0. \end{cases}$$

The other bits satisfy $m'_{kb} = m_{kb}$ for $k > 0$. Independent samples give $\mathrm{Var}(\bar{m}'_0) = a_0(1 - a_0)/B$. At $K = 5, p = 0.8$, $a_0 = 0.80032$. The head-average loss adds the common factor $1/K$. These positive scalings preserve the fixed objective's minimizer, but yield unequal expected step sizes. Adaptive or stored masks require separate conditioning.

For a single head's neural objective, let $G_k = D_\theta f_{\theta_k}$. The chain rule gives $\theta_k^+ - \theta_k = -\eta G_k^\mathsf{T} \nabla_f J(f_{\theta_k})$. Taylor expansion then gives

$$\begin{aligned} f_{\theta_k^+} &= f_{\theta_k} + G_k(\theta_k^+ - \theta_k) + O(\|\theta_k^+ - \theta_k\|^2) \\ &= f_{\theta_k} - \eta H_k \nabla_f J(f_{\theta_k}) + O(\eta^2), \qquad H_k = G_k G_k^\mathsf{T}, \end{aligned} \tag{15}$$

provided the output map has locally bounded second derivatives. Head-dependent Jacobians, shared parameters, Adam preconditioning, and changing training batches prevent identifying this update with Eq. 10. Accordingly, this calculation explains a common-target mechanism in an idealized model, while the behavior of the implemented networks is measured in Appendix D. No global convergence rate for the coupled actor–critic algorithm is asserted.

The shared-candidate premise is essential. Independently drawing candidates from the same fixed proposal preserves the same expected loss functional. Drawing them from different head policies generally does not, because their proposal distributions differ. Fixed prior functions or other restrictions on the trainable function class can also change the attainable solutions, so this argument does not rule out those mechanisms.

Repeated regression inputs require grouping. If samples in G_b share one input, set $\omega_b = \sum_{i \in G_b} w_i$ and $\mu_b = \omega_b^{-1} \sum_{i \in G_b} w_i u_i$. Completing the square yields

$$\sum_{i \in G_b} w_i \|f_b - u_i\|^2 = \omega_b \|f_b - \mu_b\|^2 + \sum_{i \in G_b} w_i \|u_i - \mu_b\|^2.$$

The second term is constant in f_b. The fixed-objective contraction argument therefore applies to the distinct-input coordinates with weights ω_b and targets μ_b. Repeated copies of one input cannot be treated as independent output variables.

Lemma 1 (Outputs do not identify parameters)**.** *Let $F(\theta)$ map parameters to outputs on the sampled support. If F is non-injective and $F(\theta) = F(\theta') = u$ for distinct parameters, then both parameters minimize $J \circ F$. Consequently, convergence of output vectors to u does not imply convergence of parameter vectors to each other.*

Proof. The assumptions give

$$J(F(\theta)) = J(F(\theta')) = J(u) = 0, \qquad \|\theta - \theta'\| > 0.$$

Constant sequences at these two minimizers have identical outputs and a nonzero parameter distance, disproving the implication. Permuting hidden units with their corresponding adjacent-layer weights can produce equivalent parameterizations. This discrete symmetry does not establish a continuous manifold of minimizers. □

A.2 Circular critic pairing

Use zero-based head indices $k \in \{0, \ldots, K-1\}$ throughout this argument. Define $S_k = \{k, (k+1) \bmod K\}$, $q_{ki} = \min_{l \in S_k} Q_l(s, \hat{a}_i)$, $w_k = \mathrm{softmax}(cq_k/\tau_{\mathrm{ent}})$, and $J_k(f) = \sum_i w_{ki} \|f_{g(i)} - u_i\|^2$, where $c, \tau_{\mathrm{ent}} > 0$ and $g(i)$ identifies the distinct regression input of sample i. Let $\omega_{kb} = \sum_{i:g(i)=b} w_{ki}$ and $b_{kb} = \sum_{i:g(i)=b} w_{ki} u_i$.

Proof. Write $\lambda = c/\tau_{\mathrm{ent}} > 0$ and $Z_k = \sum_i \exp(\lambda q_{ki})$. For each candidate,

$$\log w_{ki} - \log w_{ji} = \lambda(q_{ki} - q_{ji}) - \log(Z_k/Z_j).$$

Thus $w_k = w_j$ implies $q_k - q_j = a\mathbf{1}$ with $a = \lambda^{-1}\log(Z_k/Z_j)$. Conversely, this constant shift gives $Z_k = e^{\lambda a} Z_j$ and hence identical weights.

Differentiating on a distinct-input group gives

$$\nabla_{f_b} J_k(f) = 2 \sum_{i:g(i)=b} w_{ki}(f_b - u_i) = 2(\omega_{kb} f_b - b_{kb}).$$

Subtracting the two operators yields

$$[\nabla J_k(f) - \nabla J_j(f)]_b = 2\big[(\omega_{kb} - \omega_{jb}) f_b - (b_{kb} - b_{jb})\big]. \tag{16}$$

If this expression is zero for every f, evaluating at $f_b = 0$ gives $b_{kb} = b_{jb}$, and varying f_b then gives $\omega_{kb} = \omega_{jb}$. The converse follows by substitution. Consequently,

$$\nabla J_k \equiv \nabla J_j \quad \Longleftrightarrow \quad (\omega_{kb}, b_{kb}) = (\omega_{jb}, b_{jb}) \quad \text{for every } b.$$

At a specified common f, Eq. 16 gives the condition for different gradients. □

For $K \geq 3$, different circular pairs overlap in at most one critic. This permits different pairwise minima, but does not guarantee them. A shared critic that is the smaller member of two overlapping pairs on every candidate makes their minima identical. Constant shifts of the candidate score vector also leave the weights unchanged. For $K = 2$, both pairs coincide. Independent initialization alone supplies neither a quantitative lower bound on weight separation nor a lower bound on parameter-gradient separation.

When all candidate inputs are distinct and their outputs freely adjustable, every J_k has minimizer u. Different optima can arise from grouped inputs with different targets or imperfect approximation. For a full-column-rank model $f = A\theta$, let $W_k = \mathrm{diag}(w_{k1}, \ldots, w_{kn}) \otimes I_{d_a}$. The normal equations give

$$A^\mathsf{T} W_k (A\theta_k^* - u) = 0 \quad \Longrightarrow \quad \theta_k^* = (A^\mathsf{T} W_k A)^{-1} A^\mathsf{T} W_k u.$$

This solution can depend on k, but need not when the targets are exactly realizable. The argument establishes conditional operator separation, with finite-training diversity examined in Appendix D.

For one state's reverse candidates in Eq. 24, the noise draws u_i are targets at a shared regression input. Since the weights sum to one, the grouped target is $\mu_k = \sum_i w_{ki} u_i$ and the gradient at its prediction f_b is $2(f_b - \mu_k)$. Circular critic pairs therefore change this regression update whenever they induce different weighted targets. Appendix G.1 gives the candidate construction.

A.3 Disagreement estimators and generated trajectories

For a fixed candidate, $m \geq 1$ probe labels, and $d_a \geq 1$ coordinates, let $s_{\ell d}^2 = K^{-1} \sum_k (f_{k\ell d} - \bar{f}_{\ell d})^2$. The implemented score is $D = (md_a)^{-1} \sum_{\ell,d} s_{\ell d}$. Define its root-mean-variance counterpart $R^2 = (md_a)^{-1} \sum_{\ell,d} s_{\ell d}^2$.

Lemma 2 (Finite-ensemble norm comparison)**.** *For any finite predictions and $K \geq 2$,*

$$\frac{R}{\sqrt{md_a}} \leq D \leq R. \tag{17}$$

If $\widetilde{R}$ uses $1/(K-1)$ instead of $1/K$ in each variance, then $R = \sqrt{(K-1)/K}\,\widetilde{R}$.

Proof. Let $s \in \mathbb{R}^{md_a}$ collect the nonnegative standard deviations. The norm inequalities give

$$\frac{R}{\sqrt{md_a}} = \frac{\|s\|_2}{md_a} \leq \frac{\|s\|_1}{md_a} = D \leq \frac{\sqrt{md_a}\|s\|_2}{md_a} = R.$$

Changing the variance divisor multiplies each coordinate variance by $K/(K-1)$, so

$$\widetilde{R}^2 = \frac{K}{K-1} R^2 \quad \Longrightarrow \quad R = \sqrt{\frac{K-1}{K}}\,\widetilde{R}.$$

□

No independence assumption is needed for this identity. Independent identically distributed head predictions with finite second moments, at an input fixed independently of those draws, are sufficient for the coordinate variance with divisor $K-1$ to be unbiased for its population variance. This does not make its square root unbiased. Shared training data, overlapping critics, and selection of a candidate by the same ensemble do not establish those sufficient conditions. We therefore use finite-ensemble dispersion without a population-unbiasedness claim.

The bounds do not make the two scores proportional. For one probe and two coordinates, standard deviations $(1, 0)$ give $D = 0.5, R = 1/\sqrt{2}$, whereas $(0.6, 0.6)$ give $D = R = 0.6$. Their rankings reverse. A single change to β_0 cannot generally make their action-selection rules identical.

To isolate the trajectory argument, consider actual sampler states

$$x_{i+1}^k = g_i(x_i^k, f_k(x_i^k, t_i, s)) + \zeta_i, \qquad i = 0, \dots, L-1, \tag{18}$$

with a common initial state x_0 and common additive noise increments ζ_i. Suppose g_i is A_i-Lipschitz in its state argument and B_i-Lipschitz in its prediction argument. Set

$$V_i = \frac{1}{Kd_a}\sum_k \|x_i^k - \bar{x}_i\|^2, \qquad D_i^2 = \frac{1}{Kd_a}\sum_k \|f_k(x_i^k, t_i, s) - \bar{f}_i\|^2, \quad c_i = B_i \prod_{r=i+1}^{L-1} A_r. \tag{19}$$

The mean $\bar{f}_i$ is over predictions evaluated at each head's own sampler state.

Proposition 1 (Dispersion along coupled sampler trajectories)**.** *Under the preceding conditions,*

$$\sqrt{V_L} \le \sum_{i=0}^{L-1} c_i D_i, \qquad V_L \le \left(\sum_i c_i^2\right)\sum_i D_i^2. \tag{20}$$

The statement is pathwise and requires no independence among heads.

Proof. Write $z_i^k = f_k(x_i^k, t_i, s)$, $\delta x_i^{kj} = x_i^k - x_i^j$, and $\delta z_i^{kj} = z_i^k - z_i^j$. The finite-ensemble variance identity gives

$$V_i = \frac{1}{2K^2 d_a}\sum_{k,j} \|\delta x_i^{kj}\|^2, \qquad D_i^2 = \frac{1}{2K^2 d_a}\sum_{k,j} \|\delta z_i^{kj}\|^2.$$

The shared additive noise cancels in each pairwise difference. Lipschitz continuity then yields

$$\begin{aligned}\|\delta x_{i+1}^{kj}\| &= \|g_i(x_i^k, z_i^k) - g_i(x_i^j, z_i^j)\| \\ &\le A_i\|\delta x_i^{kj}\| + B_i\|\delta z_i^{kj}\|.\end{aligned}$$

Taking the Euclidean norm over all pairs and applying Minkowski's inequality gives

$$\begin{aligned}\sqrt{V_{i+1}} &\le \left[\frac{1}{2K^2 d_a}\sum_{k,j}\left(A_i\|\delta x_i^{kj}\| + B_i\|\delta z_i^{kj}\|\right)^2\right]^{1/2} \\ &\le A_i\sqrt{V_i} + B_i D_i.\end{aligned}$$

Since the initial state is shared, $V_0 = 0$. Unrolling the recurrence gives

$$\sqrt{V_L} \le \left(\prod_{r=0}^{L-1} A_r\right)\sqrt{V_0} + \sum_{i=0}^{L-1} B_i\left(\prod_{r=i+1}^{L-1} A_r\right) D_i = \sum_{i=0}^{L-1} c_i D_i.$$

The empty product is one. Finally, Cauchy–Schwarz gives

$$V_L \le \left(\sum_i c_i D_i\right)^2 \le \left(\sum_i c_i^2\right)\left(\sum_i D_i^2\right).$$

□

For flow Euler steps, $g_i(x, v) = x - h_i v$ with $h_i > 0$, so $A_i = 1, B_i = h_i$. With L equal steps over a unit interval, Eq. 20 gives $V_L \le L^{-1}\sum_i D_i^2$. For a DDPM mean update at reverse step t, valid constants are $A_i = 1/\sqrt{\alpha_t}$ and $B_i = (1-\alpha_t)/(\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t})$, with $0 < \alpha_t < 1$. The bound covers

ancestral sampling only under shared additive noise, conditionally on that noise and the shared initial draw, and expectation preserves the inequality. Independently sampled noise introduces additional pairwise noise differences even for identical heads.

The implemented DDPM mean clips the reconstructed clean action before combining it with the current state. Write it as

$$g_i(x, e) = a_t \operatorname{clip}_{[-1,1]}(r_t x - s_t e) + b_t x, \quad r_t = \bar{\alpha}_t^{-1/2}, \quad s_t = \sqrt{(1-\bar{\alpha}_t)/\bar{\alpha}_t}, \tag{21}$$

where $a_t = (1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}/(1-\bar{\alpha}_t)$ and $b_t = (1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}/(1-\bar{\alpha}_t)$ are nonnegative. Non-expansiveness of coordinate clipping gives state and prediction constants $a_t r_t + b_t = 1/\sqrt{\alpha_t}$ and $a_t s_t = (1-\alpha_t)/(\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t})$. Thus the same bounds cover this clipping operation.

For the unclipped deterministic DDIM update, the affine coefficients differ from the DDPM mean: $A_i = \sqrt{\bar{\alpha}_{t-1}/\bar{\alpha}_t}$ and $B_i = |\sqrt{1-\bar{\alpha}_{t-1}} - A_i\sqrt{1-\bar{\alpha}_t}|$ (Song et al., 2020a). Suppressing the noise term in a DDPM update alone does not produce that DDIM update.

The deployed TIDE score evaluates a fixed candidate action at four time labels. It does not evaluate the D_i on the generated trajectories in Eq. 18. Relating those quantities would require additional control of input mismatch and unobserved sampler steps. Proposition 1 motivates measuring disagreement during generation but is not a certified uncertainty bound for the deployed score. Dropping all intermediate terms does not yield a valid terminal-only bound, and critic-standard-deviation UCB is a different quantity from terminal policy disagreement.

A.4 Calibration and sparse rewards

Flatten the current decision batch into $M \geq 1$ candidate–state pairs. Let D_j be the nonnegative policy-disagreement score and U_j the critic standard deviation. Write $d_j = D_j - \bar{D}$, $u_j = U_j - \bar{U}$, $\epsilon_g > 0$ for the denominator stabilizer, and $\beta_0 \geq 0$ for the fixed bonus scale. The implementation uses

$$\rho_\epsilon = \frac{d^\mathsf{T} u}{\|d\|\|u\| + \epsilon_g}, \quad \beta = \beta_0 \max(0, \rho_\epsilon), \quad S_j = \bar{Q}_j + \beta D_j. \tag{22}$$

Here $\epsilon_g = 10^{-8}$. Define $\operatorname{osc}(z) = \max_j z_j - \min_j z_j$.

Proposition 2 (Suppression by the regularized gate)**.** *For any finite candidate batch,* $0 \leq \beta \leq \beta_0$*. If* $d^\mathsf{T} u \leq 0$*, including either centered vector being zero, then* $\beta = 0$*. More generally,*

$$\operatorname{osc}(S) \leq \operatorname{osc}(\bar{Q}) + \beta_0 \operatorname{osc}(D) \min\left\{1, \frac{\|d\|\|u\|}{\epsilon_g}\right\}. \tag{23}$$

For a fixed finite pool size, uniformly bounded disagreement scores D*, and fixed* $\epsilon_g > 0$*, critic dispersions converging to a common constant imply* $\beta \to 0$*.*

Proof. Write $z = \|d\|\|u\|$ and $x_+ = \max(0, x)$. Cauchy–Schwarz gives

$$0 \leq \beta = \beta_0 \frac{(d^\mathsf{T} u)_+}{z + \epsilon_g} \leq \beta_0 \frac{z}{z + \epsilon_g} \leq \beta_0 \min\{1, z/\epsilon_g\}.$$

The numerator is zero whenever $d^\mathsf{T} u \leq 0$. For any two candidates,

$$\begin{aligned} |S_j - S_l| &\leq |\bar{Q}_j - \bar{Q}_l| + \beta |D_j - D_l| \\ &\leq \operatorname{osc}(\bar{Q}) + \beta \operatorname{osc}(D). \end{aligned}$$

Maximizing over j, l and substituting the bound on β proves Eq. 23.

For the limit, index decision batches by n and set $C = \sup_n \|d^{(n)}\| < \infty$. If $U_j^{(n)}$ approach a common constant, then $\|u^{(n)}\| \to 0$, and

$$0 \leq \beta^{(n)} \leq \frac{\beta_0}{\epsilon_g} \|d^{(n)}\| \|u^{(n)}\| \leq \frac{\beta_0 C}{\epsilon_g} \|u^{(n)}\| \longrightarrow 0.$$

□

A flat ensemble mean alone is insufficient. Two critics $Q_1(a) = h(a)$ and $Q_2(a) = -h(a)$ have mean zero and dispersion $|h(a)|$. Taking $D(a) = |h(a)|$ nonconstant gives a positive correlation and, when $\beta_0 > 0$, a positive bonus coefficient. Moreover, $\beta = 0$ selects the largest ensemble-mean value in the pooled candidates, followed by the implemented exploration noise. It does not implement per-episode Thompson commitment.

Lemma 3 (An independent-sample null model)**.** *Suppose (D_j, U_j) are i.i.d. draws from a fixed joint distribution with independent coordinates, positive marginal variances, and finite fourth moments. Then, for fixed β_0, $\mathbb{E}\beta = O(\beta_0/\sqrt{M})$ as $M \to \infty$.*

Proof. Let $X_j = D_j - \mathbb{E}D_j$ and $Y_j = U_j - \mathbb{E}U_j$. With I_M the identity matrix and $\mathbf{1}_M = (1, \dots, 1)^\mathsf{T} \in \mathbb{R}^M$, define the centering matrix $H = I_M - M^{-1}\mathbf{1}_M\mathbf{1}_M^\mathsf{T}$. Thus $HX = X - \bar{X}\mathbf{1}_M$. For the column vectors X, Y, define

$$C_M = M^{-1}X^\mathsf{T}HY, \qquad V_D = M^{-1}X^\mathsf{T}HX, \qquad V_U = M^{-1}Y^\mathsf{T}HY.$$

The vectors X and Y are independent, with covariance matrices $\sigma_D^2 I_M$ and $\sigma_U^2 I_M$. Since $H^2 = H$ and $\operatorname{tr}(H) = M - 1$,

$$\begin{aligned}\mathbb{E}C_M^2 &= M^{-2}\operatorname{tr}\big(H\mathbb{E}[YY^\mathsf{T}]H\mathbb{E}[XX^\mathsf{T}]\big)\\ &= \frac{\sigma_D^2\sigma_U^2}{M^2}\operatorname{tr}(H^2) = \frac{M-1}{M^2}\sigma_D^2\sigma_U^2.\end{aligned}$$

To control the denominator, write $V_D = M^{-1}\sum_j X_j^2 - \bar{X}^2$ and $\mu_{4,D} = \mathbb{E}X_j^4 < \infty$. Independence and $(a-b)^2 \le 2a^2 + 2b^2$ give

$$\begin{aligned}\mathbb{E}(V_D - \sigma_D^2)^2 &\le 2\mathbb{E}\left(M^{-1}\sum_j X_j^2 - \sigma_D^2\right)^2 + 2\mathbb{E}\bar{X}^4\\ &= \frac{2(\mu_{4,D} - \sigma_D^4)}{M} + \frac{2\big(\mu_{4,D} + 3(M-1)\sigma_D^4\big)}{M^3} = O(M^{-1}).\end{aligned}$$

The same bound holds for V_U. For $E_M = \{V_D \ge \sigma_D^2/2,\ V_U \ge \sigma_U^2/2\}$, Markov's inequality applied to the squared deviations and a union bound yield

$$\Pr(E_M^c) \le \frac{4\mathbb{E}(V_D - \sigma_D^2)^2}{\sigma_D^4} + \frac{4\mathbb{E}(V_U - \sigma_U^2)^2}{\sigma_U^4} = O(M^{-1}).$$

The implemented normalization satisfies

$$\rho_\epsilon = \frac{C_M}{\sqrt{V_D V_U} + \epsilon_g/M}, \qquad \rho_\epsilon^2\mathbf{1}_{E_M} \le \frac{4C_M^2}{\sigma_D^2\sigma_U^2}.$$

Here $\mathbf{1}_{E_M}$ is the indicator of the event E_M. Together with $|\rho_\epsilon| \le 1$, this gives

$$\begin{aligned}\mathbb{E}\rho_\epsilon^2 &\le \frac{4\mathbb{E}C_M^2}{\sigma_D^2\sigma_U^2} + \Pr(E_M^c) = O(M^{-1}),\\ \mathbb{E}\beta &= \beta_0\mathbb{E}(\rho_\epsilon)_+ \le \beta_0\sqrt{\mathbb{E}\rho_\epsilon^2} = O(\beta_0 M^{-1/2}),\end{aligned}$$

where the last inequality is Cauchy–Schwarz. □

These assumptions define an auxiliary null model. Candidates generated by related heads and evaluated by shared trained networks need not satisfy them. Exchangeability or zero correlation alone is insufficient for this argument. The lemma supplies neither a universal $1/\sqrt{2\pi M}$ constant nor a sparse-reward performance guarantee.

Corollary 1 (An idealized zero-reward limit)**.** *Consider exact synchronous Bellman iteration for bounded critics in a zero-reward discounted problem, with $0 \le \gamma < 1$, a fixed next-action distribution, and a fixed distribution over nonempty critic subsets. Define $(\mathcal{T}Q)_k(s,a) = \gamma\mathbb{E}[\min_{l\in\mathcal{M}} Q_l(s',a')]$ for every k. Then its unique bounded fixed point is zero and $\|Q^{(t)}\|_\infty \le \gamma^t\|Q^{(0)}\|_\infty$. Under the bounded-score and finite-pool conditions of Proposition 2, its gate therefore vanishes in the limit.*

Proof. Equip the product of bounded critic functions with $\|Q\|_\infty = \sup_{k,s,a} |Q_k(s,a)|$. For two collections Q, Q', the same transition, action, and subset laws give

$$|(\mathcal{T}Q)_k(s,a) - (\mathcal{T}Q')_k(s,a)| \le \gamma \mathbb{E}\left|\min_{l\in\mathcal{M}} Q_l(s',a') - \min_{l\in\mathcal{M}} Q'_l(s',a')\right| \le \gamma\|Q - Q'\|_\infty.$$

Taking the supremum proves the contraction. Since $\mathcal{T}0 = 0$, any bounded fixed point Q^* satisfies

$$\|Q^*\|_\infty \le \gamma\|Q^*\|_\infty \quad \Longrightarrow \quad Q^* = 0.$$

Iterating the contraction around zero gives

$$\|Q^{(t)}\|_\infty \le \gamma^t \|Q^{(0)}\|_\infty \longrightarrow 0.$$

For every candidate j, its critic dispersion therefore obeys

$$(U_j^{(t)})^2 = \frac{1}{K}\sum_k (Q_{kj}^{(t)})^2 - (\bar{Q}_j^{(t)})^2 \le \|Q^{(t)}\|_\infty^2 \longrightarrow 0.$$

The gate limit follows from Proposition 2 under its uniform boundedness and fixed-pool assumptions. □

The corollary assumes a zero-reward problem with exact updates on its full domain. A neural critic fitted to a finite replay buffer before the first observed reward does not satisfy those premises merely because its observed rewards are zero. Bootstrapping outside observed states, approximation error, changing policies, and optimization can all preserve value variation. An explicit entropy term in the target would also change the zero fixed point. The result identifies a possible limiting case, while the behavior of ESAC on sparse tasks remains an experimental question.

B Controlled comparisons

The controls examine the effects of flow matching and additional updates. UTD counts critic update rounds per collected transition; equal UTD does not imply equal network evaluations. Table 2 separates the single-policy flow comparison from the Ant update-rate comparison. Appendix H examines larger individual networks.

B.1 Single-policy flow control

A K=1 flow-matching agent reaches 2555 ± 627 on Ant against 5646 ± 1001 for K=5, a $2.2\times$ difference. On HalfCheetah, the single-policy control has a slightly higher mean (10723 ± 337 versus 10328 ± 1119). Walker2d and Hopper show smaller increases, from 3498 to 4305 and from 2405 to 2759, respectively. The largest difference is on Ant. This comparison holds the policy architecture and interaction schedule fixed while increasing the policy count, candidate pool, parameter count, and critic computation; it does not isolate multiple learned proposal distributions.

B.2 Update count

Raising SDAC's critic UTD to 1.0 reduces its Ant return to 21 ± 9, versus 798 ± 190 at the inherited schedule. SAC at the same increased update rate reaches 5477 ± 625. SAC benefits strongly from the additional updates and approaches the flow ensemble's mean, while SDAC is sensitive to this change in optimization schedule. Appendix H reports parameter-scaling controls. An actor-widened SDAC with 0.84M parameters reaches 719 ± 168 on Ant. A widened SAC with 1.64M parameters reaches 4658 ± 940, compared with 5646 ± 1001 for ESAC (FM) with 1.45M parameters.

C Exploration and TIDE ablations

Table 3 compares exploration rules and TIDE components across multiple seeds. All entries use flow-matching policies, $K = 5$, $N = 32$, $p = 0.8$, and REDQ $M = 2$. The two component ablations remove the gate, leaving the fixed bonus coefficient, or use only time label zero. Results aggregate five seeds, except the two Hopper component ablations, which use three. The single-label variant evaluates policy disagreement at label zero, while UCB uses critic standard deviation.

TIDE has the highest mean on Ant, Walker2d, and Hopper, while HalfCheetah shows no advantage. Variation across seeds is substantial, especially for the three-seed Hopper component experiments. The largest separation among the three selection rules occurs on Ant and Walker2d. On Hopper, the full rule is close to Thompson and UCB but improves substantially over the two component variants.

These comparisons use the same flow-matching policy architecture and therefore examine action selection within the same actor-training scheme.

Table 2: The five-head flow configuration has a higher mean than the single-policy control on three of four tasks, increasing SAC's update rate also brings its Ant mean close to ESAC (FM). Results are mean ± standard deviation across five seeds. Green bold cells mark the largest mean within each comparison.

(a) Flow policies (UTD 0.2).

Environment	Flow $K = 1$	ESAC (FM) $K = 5$
Ant-v4	2555 ± 627	$\mathbf{5646 \pm 1001}$
Walker2d-v4	3498 ± 457	$\mathbf{4305 \pm 529}$
Hopper-v4	2405 ± 346	$\mathbf{2759 \pm 437}$
HalfCheetah-v4	$\mathbf{10723 \pm 337}$	10328 ± 1119

(b) Update rate (Ant-v4).

Method	Critic UTD	Return
SAC	0.2	2480 ± 820
SAC	1.0	5477 ± 625
SDAC	0.2	798 ± 190
SDAC	1.0	21 ± 9
ESAC (FM) $K = 5$	0.2	$\mathbf{5646 \pm 1001}$

Table 3: Full TIDE has the highest mean on Ant, Walker2d, and Hopper, while the one-label variant has the highest mean on HalfCheetah. Exploration ablations use flow-matching policies and report mean ± standard deviation. Results use five seeds, except the two Hopper component ablations, which use three. Green bold cells mark the largest mean per task.

Environment	Thompson	UCB	TIDE	TIDE, no gate	TIDE, one label
Ant-v4	5094 ± 416	4700 ± 872	$\mathbf{5646 \pm 1001}$	5068 ± 508	5155 ± 330
Walker2d-v4	3838 ± 714	3251 ± 794	$\mathbf{4305 \pm 529}$	3468 ± 558	3750 ± 228
Hopper-v4	2549 ± 654	2606 ± 607	$\mathbf{2759 \pm 437}$	1488 ± 1208	1308 ± 642
HalfCheetah-v4	10341 ± 652	10601 ± 1031	10328 ± 1119	10627 ± 715	$\mathbf{10773 \pm 312}$

D DIVERSITY DURING TRAINING

We examine action-mean cosine similarity, output distances, parameter distances, and critic dispersion. Together, these measurements distinguish separation between the predictions of the heads from separation between their parameters.

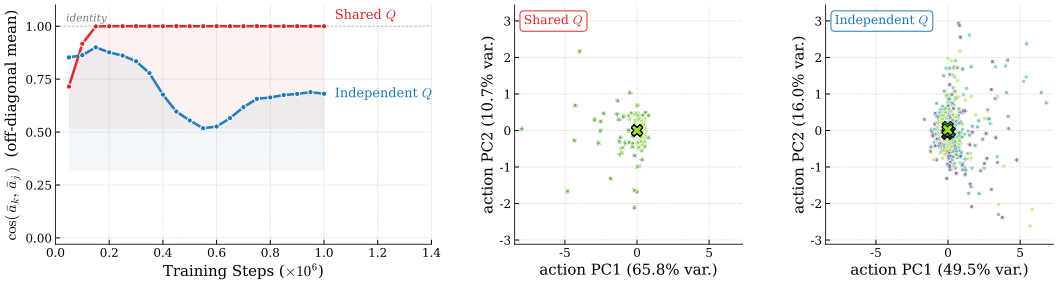

Figure 4: Shared critics align the heads' mean-action directions, while independent critics retain more distinct proposals in this Ant-v4 comparison ($K = 5$). Panels show head-mean cosine similarity (left) and candidate-action PCA under shared (middle) and independent (right) critics. Crosses mark head centroids.

In Figure 4, under shared critics, the mean-action directions become aligned and the prediction-distance diagnostic decreases. With separate critics, head predictions remain farther apart over the measured interval. The accompanying parameter distances stay separated even when predictions become similar, illustrating the distinction formalized in Lemma 1. Cosine similarity measures directional alignment, while the output-distance and candidate projections provide complementary information about the proposals.

In Figure 5, the policy diagnostic averages coordinate-wise standard deviations across heads, while the critic diagnostic takes the standard deviation of minibatch-mean values across critics. The temporal measurements show greater prediction disagreement for flow than for diffusion. They characterize the trained populations under the two actor objectives in Section 3.1. The behavioral analyses below examine whether this predictive separation corresponds to distinct actions over an episode.

E POLICY BEHAVIOR AND VALUE DIAGNOSTICS

A central question for an ensemble policy is whether differences between heads correspond to different behaviors. We examine action spectra on Swimmer and learned Q-values on locomotion benchmarks. Appendix F reports related measurements on peg insertion. These diagnostics complement the return comparisons and the fixed-regression analysis in Appendix A.

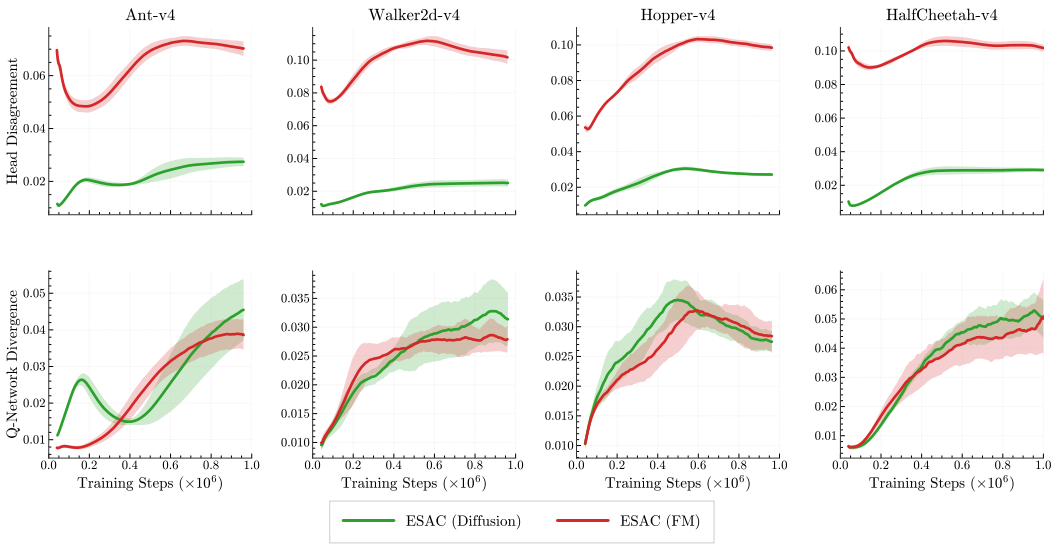

Figure 5: Flow policies show greater prediction disagreement than diffusion policies over the measured training interval. Panels show policy prediction disagreement (top) and critic dispersion (bottom). Policy scores use noised replay actions; critic scores use minibatch-mean Q-values.

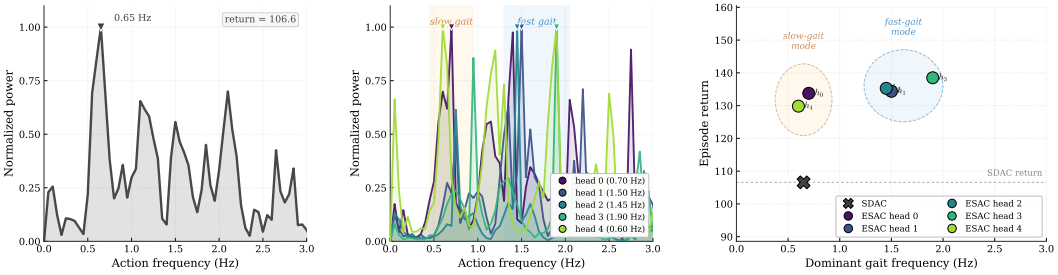

Figure 6: ESAC flow heads exhibit two gait-frequency groups, and every head exceeds the SDAC rollout return in the inspected seed. Panels show action spectra for SDAC (a) and ESAC heads (b), followed by dominant frequency versus return (c).

E.1 Gait spectrum analysis on Swimmer

Swimmer-v4 admits multiple high-value gait modes corresponding to different oscillation frequencies of the two body joints, and the following experiment compares the behavior reached by one trained policy with that reached by an ensemble. We trained an SDAC single-policy baseline and ESAC with flow-matching policies for 1M steps at the same seed, rolled each policy out deterministically for 1,000 steps, and computed the discrete Fourier transform of the action time series, where the dominant peak identifies the converged gait frequency.

In Figure 6, SDAC concentrates its action power near 0.65 Hz with a single dominant peak and return 106.58. The five ESAC heads form two separated clusters: heads 0 and 4 near 0.60–0.70 Hz, and heads 1, 2, 3 between 1.45 and 1.90 Hz. Every head has a return 23–32 above this SDAC rollout, and the highest-return head has a dominant frequency near 1.9 Hz. The population therefore retains both a gait near the SDAC solution and a faster family of gaits in this seed. This behavioral distinction complements the head-prediction distances by showing different temporal action patterns during interaction. It does not establish that these modes persist across training seeds or tasks.

E.2 Learned value distributions on locomotion benchmarks

We examine learned critics from five independently trained SDAC agents after 1M steps (5M for Humanoid). At ten reference states from a trained agent's rollout, we evaluate 3,000 uniformly sampled actions and plot their Q-values. A UMAP projection shows the sampled actions colored by normalized Q-value. These are measurements of fitted critics on a finite set of states.

Figure 7 shows a dominant peak in the sampled Q-values for HalfCheetah and Ant, with two clearly separated peaks for Humanoid. The UMAP views locate regions with larger predicted values among the sampled actions. These measurements characterize the fitted value distribution, while action

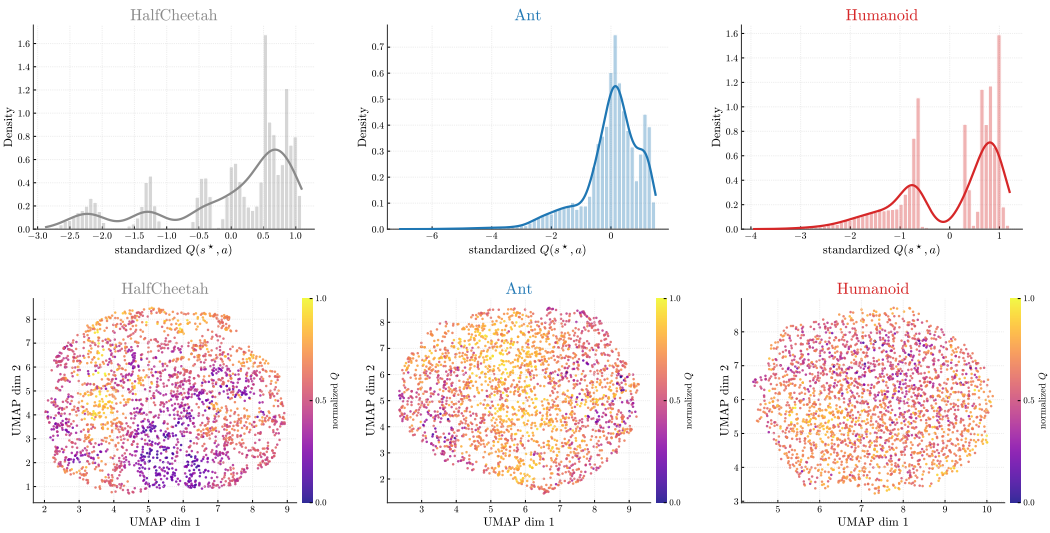

Figure 7: Sampled Q-value distributions (top) and action-space UMAP colored by normalized Q-value (bottom). Humanoid shows two separated peaks at the inspected states.

spectra and rollouts characterize behavior. Counting peaks in a scalar-value histogram alone does not determine the number of behavioral modes.

F PEG INSERTION DIAGNOSTICS

We retrained SDAC and ESAC on `peg-insert-side-v3` for 500k steps and examined Q-values over random actions, action spectra, and per-head rollouts. These single-seed measurements examine how the ensemble behaves on a task where its mean success is below the single-policy baseline.

F.1 VALUE DISTRIBUTIONS

Figure 8 applies the protocol of Appendix E.2. We evaluate Q-values over 3,000 uniformly random actions at ten on-policy reference states, using the SDAC $\min(Q_1, Q_2)$ critic and the ESAC ensemble-mean critic. Both give a single peak at every reference state, and the action-space UMAP shows a concentrated high-Q region. Both fitted critics concentrate predicted value in a similar part of the sampled action set.

F.2 ACTION SPECTRA

Figure 9 repeats the gait-spectrum analysis. Where Swimmer split the heads into two well-separated frequency clusters with every head beating the baseline, peg-insert produces the opposite pattern. Four of the five heads concentrate in the same low-frequency band as SDAC, with dominant frequencies of 0.08–0.16 Hz against SDAC's 0.16 Hz, and all four fall below the SDAC return of 597.2. The fifth head has a dominant frequency of 5.76 Hz and near-zero return, corresponding to high-frequency jitter. The spectral differences do not correspond to a higher-return alternative in these rollouts.

F.3 PER-HEAD ROLLOUTS

Figure 10 reports ten deterministic rollouts per head. Per-head success over those rollouts is $[0, 0, 0, 0.1, 0]$, and the ratio of between-head to within-head (across-episode) trajectory divergence is 0.73. The ratio places between-head separation below within-head variation under this diagnostic. Head identity consequently accounts for less of the measured trajectory variation than the differences between episodes from the same head.

The individual-head rollouts measure the behaviors available to the ensemble at this checkpoint. The main manipulation comparison evaluates the pooled interaction policy across five seeds. Taken together, the spectral and trajectory measurements show limited useful separation between the individual heads on peg insertion.

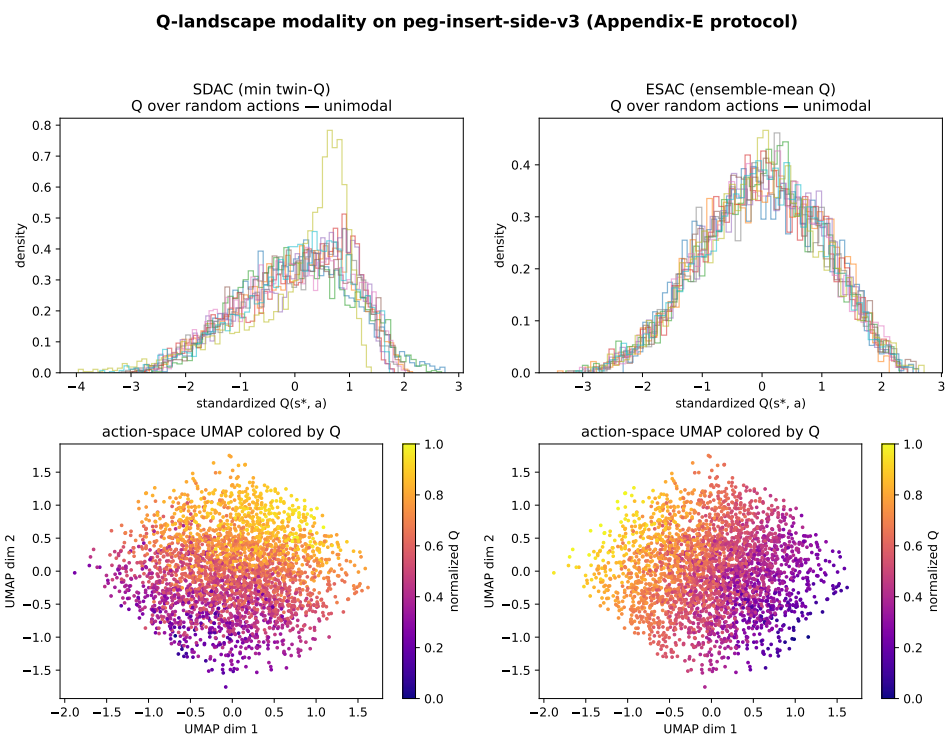

Figure 8: Peg-insertion Q-value distributions (top) and action-space UMAP (bottom) for SDAC (left) and ESAC (right). Both show one dominant peak in this single-seed diagnostic.

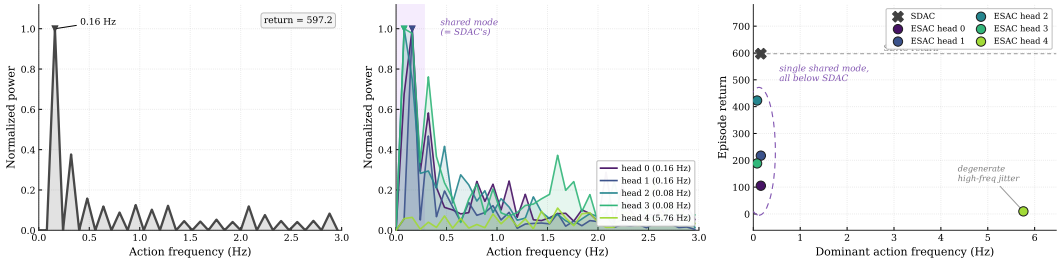

Figure 9: Peg-insertion action spectra for SDAC (a) and ESAC heads (b), with dominant frequency versus return (c). Every head falls below the SDAC return in this single-seed diagnostic.

G Implementation Details

ESAC is implemented in JAX (Bradbury et al., 2018) with dm-haiku for network definitions, and trained with Adam. The whole training step (critic loss, policy loss for all K heads, temperature loss, target updates) is compiled as a single JIT-traced function. The gate is stateless and recomputed from the current candidate pool at each decision, as in Eq. 9, without averaging over previous decisions.

G.1 Actor proposal construction

For each replay state, head 0 generates $N = 32$ proposals and the minimum of critics 0 and 1 selects the training action. Candidate values are evaluated before action clipping, and Gaussian noise is added after selection and clipping, as in Eq. 4. Actions for next-state Bellman targets use the same rule. Regression targets and importance weights use parameters fixed at the start of the update.

For diffusion, sample a time label ℓ and forward noise ϵ to obtain $x_\ell = \sqrt{\bar{\alpha}_\ell}a^\dagger + \sqrt{1-\bar{\alpha}_\ell}\epsilon$. Draw N_{mc} reverse-noise targets $u_i \sim \mathcal{N}(0, I)$ and construct

$$\hat{a}_i = \text{clip}\left(\frac{x_\ell - \sqrt{1-\bar{\alpha}_\ell}u_i}{\sqrt{\bar{\alpha}_\ell}}, -1, 1\right), \qquad \mu_k = \sum_i w_{ki}u_i. \tag{24}$$

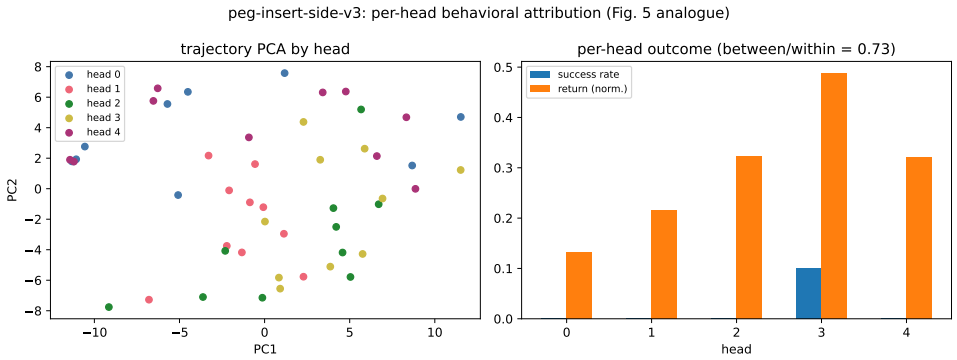

Figure 10: Peg-insertion trajectory PCA (left) and per-head success and return (right). Between-head separation is smaller than within-head variation in this checkpoint diagnostic.

Table 4: Default ESAC settings for the dense-reward MuJoCo experiments.

Hyperparameter	Value	Notes
Critic architecture	MLP [256, 256, 256]	Mish activation
Policy architecture	MLP [256, 256, 256]	Mish activation, sinusoidal time embedding
Learning rate (critic, policy)	3×10^{-4}	Linear decay to 3×10^{-5}
Temperature learning rate	7×10^{-3}	
Q-value scale ψ	5	Multiplier inside the regression softmax
Critic UTD ratio	0.2	1 critic update per 5 environment steps
Policy / target UTD ratio	0.1	policy and target update every 2nd critic update
Temperature update interval	250 critic updates	
Discount γ	0.99	
Reward scale c_r	0.2	Multiplier in the critic target
Soft target update τ	0.005	
Replay buffer capacity	10^6 transitions	
Batch size B	256	
Number of heads K	5	single-policy controls use the SDAC implementation
REDQ subset size M	2	subset minimum for the critic target
Loss-scaling probability p	0.8	
Particles per head N	32	TIDE and UCB pool $K \times N = 160$
Denoising / flow steps T	20	
UCB exploration bonus β	1.0	
TIDE base bonus β_0	2.0	modulated by $\max(0, \rho_t)$
TIDE probe timesteps $\mathcal{T}_{\text{TIDE}}$	$\{0, T/4, T/2, 3T/4\}$	fixed candidate input at four time labels
TIDE gate ρ_t	per decision, stateless	correlation over the current candidate pool
paired values per decision	800	$K \times N$ candidates $\times$ 5 vectorized envs
Exploration noise scale σ_{act}	0.1	multiplied by τ_{ent}
Beta schedule (diffusion only)	Linear, scale 0.8	
Warm-up steps (random policy)	30000	
Vectorized environments	5	
Monte Carlo candidates N_{mc}	64	reverse proposals shared across heads

Circular-pair values of $\hat{a}_i$ determine w_{ki}. All reverse candidates share x_ℓ, so the gradient is the gradient of regression toward μ_k. This is the grouped-input construction analyzed in Appendix A.2.

For flow matching, the selected training action and forward noise determine the input and conditional velocity target,

$$x_t = (1-t)a^\dagger + t\epsilon, \qquad u = \epsilon - a^\dagger. \tag{25}$$

Each flow head fits this velocity using Eq. 7, with critic guidance entering through the selection of training actions. Both variants collect experience through the same ensemble action-selection framework.

The diffusion loss averages over $N_{\text{mc}} = 64$ reverse candidates and d_a coordinates. The flow loss uses coordinate averaging and the fixed normalization $c_{\text{FM}} = 1/64$. Both also average over states and heads. These factors rescale gradients, and the step-size condition in Appendix A.1 refers to the unnormalized quadratic J used there.

Networks receive integer time labels. Flow interpolation maps ℓ to $(\ell+1)/T$, clipped to $[10^{-5}, 1-10^{-5}]$. Diffusion sampling initializes its latent at 0.5 times standard Gaussian noise, while flow sampling starts from a standard Gaussian. At evaluation, candidate seeds are determined by the observation, so the selection rule is deterministic for a given state and set of network parameters.

G.2 Action bounds and replay

The rule for selecting training actions scores the raw proposals before clipping, then adds Gaussian noise to the selected clipped action. Neither actions used for actor training nor Bellman next actions are clipped again. During TIDE and UCB interaction, proposals are instead clipped before critic scorin, the selected proposal then receives Gaussian noise, again with no final clipping.

The collector stores that post-noise command in replay and fits the critics on it. The environment wrapper converts its dtype and rescales if needed, without adding clipping. Environment-side actuator limits or task-specific clipping can affect the physical control: MetaWorld clips the Cartesian action components internally, while MuJoCo actuator limits need not remove a penalty computed from the original command, as in Ant's control cost. Replay therefore records the requested command, rather than a reconstructed actuator input. The separate evaluator clips the selected action to $[-1, 1]$ and adds no exploration noise. These distinctions matter near the action bounds and are part of the reported implementation.

G.3 Loss scaling and update schedules

The actor multiplier is the minibatch mean of independently drawn Bernoulli bits. Averaging before applying it gives a scalar loss scale for each head. Every critic uses the full shared minibatch. The head-0 adjustment and the expected multiplier are given in Appendix A.1.

After the random-action warm-up, the off-policy schedule performs one critic update round per five transitions and one actor update per two critic rounds. Each ESAC critic round updates all K critics. The policy update averages the K actor losses. PPO and DPPO use their own actor-update procedures. Comparisons therefore distinguish environment interactions, update rounds, trained-network counts, and elapsed time.

G.4 Temperature schedule

Let $\alpha = \log \tau_{\text{ent}}$. The analytic entropy of the added action noise $\mathcal{N}(0, (0.1e^{\alpha})^2 I)$ is

$$h_{\text{noise}}(\alpha) = d_a\big(\alpha + \log 0.1 + \tfrac{1}{2}\log(2\pi e)\big). \tag{26}$$

Every 250 critic updates, Adam updates α using the gradient

$$\nabla_\alpha \mathcal{L}_\alpha = h_{\text{noise}}(\alpha) - h_{\text{target}}, \qquad h_{\text{target}} = -0.9 d_a. \tag{27}$$

This scalar update depends on the current temperature and a fixed entropy target. Its trajectory is determined by initialization and the optimization schedule, independently of learned policy entropy or observed rewards. The stationary temperature is

$$\tau^*_{\text{ent}} = \frac{\exp(h_{\text{target}}/d_a)}{0.1\sqrt{2\pi e}} \approx 0.984. \tag{28}$$

Training initializes $\tau_{\text{ent}} = 5$, so the temperature varies during optimization. It controls the diffusion weight concentration and added action noise. The stationary value characterizes the scalar schedule rather than the entropy of the generative action distribution.

G.5 Gate sensitivity

The TIDE settings are $\epsilon_g = 10^{-8}$, $\beta_0 = 2$, and $\mathcal{T} = \{0, \lfloor T/4 \rfloor, \lfloor T/2 \rfloor, \lfloor 3T/4 \rfloor\}$.

The gate-off ablation replaces $\beta_0 \max(0, \rho_t)$ with the fixed coefficient β_0. The full rule computes ρ_t over the current candidate pool at each decision, without temporal averaging. Table 3 compares these variants over five seeds, with three seeds for the Hopper component ablations. Temporal smoothing remains unevaluated.

G.6 Evaluation protocol

Each ESAC checkpoint is evaluated in a separate environment for 20 episodes. The evaluation schedule contains 20 checkpoints, spaced every 50k training steps for the 1M runs and every 250k

steps for Humanoid. Training-step counts begin after the 30k random-action warm-up. Evaluation interactions are separate from training interactions. An episode ends at environment termination or the task time limit. HalfCheetah, Hopper, Walker2d, Ant, Swimmer, and Humanoid have a 1,000-step limit, Pusher has a 100-step limit, and Reacher has a 50-step limit. The checkpoint score is the mean undiscounted episode return over the 20 episodes, using native task rewards. The specified ESAC configurations and controls using the SDAC implementation match interaction and update schedules. Table 1 does not establish a fixed-compute comparison across all baselines.

Evaluation uses exponential moving averages of the actor parameters and the current critic ensemble. Each of the five heads generates 32 candidates, and every critic scores the resulting 160 actions. The action maximizing the mean of the five critic values is executed. Candidate seeds are determined by the observation, making the policy deterministic for fixed parameters and input. The TIDE bonus and additional exploration noise are disabled. At collection time, the moving-average actors also supply proposals and disagreement predictions, with the configured exploration rule and added Gaussian noise.

H PARAMETER SCALING

The parameter-scaling experiments compare larger individual networks with the policy population (Table 5). The widened SDAC actor has 0.84M parameters, ESAC has 1.45M, and widened SAC has 1.64M. Writing h_π and h_Q for actor and critic hidden-layer widths, scaled SDAC widens only the actor ($h_\pi = 512$, $h_Q = 256$), while scaled SAC uses width 512 throughout. Parameter counts use Ant-v4 dimensions. These configurations compare capacity without matching candidate search or network evaluations.

Table 5: ESAC (FM) has the highest mean on Ant and Walker2d in this comparison; base SAC and SDAC lead on Hopper and HalfCheetah, respectively. SDAC scaled widens only the actor ($h_\pi = 512$, $h_Q = 256$), while SAC scaled uses width 512 for both actor and critic. Parameter counts use Ant-v4 dimensions. Results are mean $\pm$ standard deviation across five seeds at 1M training steps. Base and ESAC rows are reproduced from Table 1. Green bold cells mark the highest mean per environment.

Method	Ant-v4	Walker2d-v4	Hopper-v4	HalfCheetah-v4
SAC, h=256 (0.43M)	2480 ± 820	3110 ± 950	$\mathbf{3110 \pm 345}$	8745 ± 415
SAC scaled, h=512 (1.64M)	4658 ± 940	3772 ± 1290	2574 ± 1103	7340 ± 3097
SDAC, h=256 (0.43M)	798 ± 190	3176 ± 782	2472 ± 993	$\mathbf{11949 \pm 659}$
SDAC scaled, h_π=512 (0.84M)	719 ± 168	3275 ± 1180	2584 ± 716	11478 ± 392
ESAC (Diff.), K=5 (1.45M)	4043 ± 617	4172 ± 483	2284 ± 1147	11310 ± 728
ESAC (FM), K=5 (1.45M)	$\mathbf{5646 \pm 1001}$	$\mathbf{4305 \pm 529}$	2759 ± 437	10328 ± 1119

Widening only the SDAC actor yields 719 ± 168 on Ant, compared with 798 ± 190 for the base configuration. This capacity increase does not improve the mean return, and the difference is small relative to seed variability. The widened SAC configuration achieves a higher mean than ESAC with diffusion on Ant but remains below ESAC with flow matching. These comparisons show that the effect of additional parameters depends on the algorithm and task.

I COMPUTATIONAL COST

Table 6: ESAC (FM) considers five times as many action candidates and takes approximately three times SDAC's training time at the reported interaction budgets. The table reports parameter counts, candidate budgets, and elapsed training time. Timings use two concurrent jobs per RTX A6000.

Method	Parameters	Candidates per decision	Ant-v4 1M steps	Humanoid-v4 5M steps
SDAC	0.43M	32	2.6 h	14.3 h
ESAC (FM), K=5	1.45M	160	7.9 h	38.6 h

Training uses two concurrent jobs per RTX A6000, so the end-to-end times reflect shared GPU execution. Parameter counts in Table 6 use Ant-v4 dimensions. SDAC requires approximately 2.2–2.6 hours per 1M locomotion interactions, compared with 7.7–11.2 hours for ESAC with diffusion and 7.9–8.2 hours for flow matching. Humanoid at 5M interactions takes approximately 14.3 hours for SDAC and 38.6 hours for ESAC flow. Run counts vary across these timing measurements.

Per environment, pooling N candidates from each of K heads requires TKN policy evaluations counted per candidate and denoising step, K^2N critic evaluations for the pool, and $|\mathcal{T}|K^2N$ policy evaluations for the TIDE probe. These are scalar-example counts, which the implementation batches. Their ratio is not a wall-clock or FLOP estimate. Update costs additionally include the K critics and actor heads.

J Dense and sparse rewards

The MuJoCo and MetaWorld experiments use dense rewards. Their intermediate feedback can guide candidate selection before a complete behavior is learned. Sparse rewards place greater emphasis on discovering successful behavior before this value information becomes available. The current experiments characterize the dense setting. Proposition 2 analyzes gate suppression for given critic predictions, and Corollary 1 describes exact zero-reward Bellman iteration under its stated assumptions.

J.1 Checkpoint measurements

We evaluate the gate at training checkpoints using 20 reference states collected from the final policy, held fixed across checkpoints. Each state is evaluated separately with 160 candidates. This statewise measurement isolates the relationship between disagreement and critic dispersion within each candidate pool. Training instead uses one correlation pooled across the five simultaneous states. Table 7 summarizes the statewise measurements.

For descriptive purposes define

$$r(s) = \frac{\mathrm{mean}_j\, \sigma_Q(s, a_j)}{\max_j \bar{Q}(s, a_j) - \min_j \bar{Q}(s, a_j) + 10^{-8}}. \tag{29}$$

Here $\sigma_Q(s, a_j) = U_j$ is the critic standard deviation defined in Section 3.2. This descriptive ratio measures critic dispersion relative to the value range of the available proposals. Larger values indicate greater disagreement compared with the spread in ensemble-mean scores.

Table 7 averages ρ over the reference states; "open" is the fraction with positive correlation. Locomotion uses three seeds (mean $\pm$ standard deviation), and peg insertion uses one. Early and late checkpoints are 50k and 1M interactions for locomotion, and 25k and 500k for peg insertion. The performance column gives TIDE minus UCB return for locomotion, and ESAC diffusion minus SDAC success in percentage points for peg insertion. The ratio r measures critic dispersion, not the TIDE bonus magnitude.

Table 7: Late-checkpoint mean correlations are positive on Ant and Walker2d, near zero on HalfCheetah, and negative in the peg-insertion diagnostic. Measurements use statewise gates at early and late checkpoints. Green/orange cells indicate positive/negative performance differences; definitions, seed counts, and units are given in the text.

Environment	ρ early	ρ late	open early	open late	r early / late	Performance difference
Ant-v4	-0.087 ± 0.033	$+0.148 \pm 0.056$	0.31	0.67	0.55 / 0.30	$\mathbf{+946}$
Walker2d-v4	$+0.026 \pm 0.073$	$+0.134 \pm 0.090$	0.51	0.71	1.42 / 0.34	$\mathbf{+1054}$
HalfCheetah-v4	-0.008 ± 0.185	-0.000 ± 0.113	0.56	0.47	2.45 / 0.19	$\mathbf{-273}$
peg-insert-side-v3	-0.183	-0.105	0.20	0.40	1.25 / 0.70	$\mathbf{-12.0}$ pt

Ant and Walker2d have positive late-checkpoint mean correlations in these measurements. HalfCheetah has a mean near zero, while peg insertion has a negative mean in its single diagnostic seed. Ant and Walker2d also favor TIDE over UCB in mean, giving a consistent task pattern in this retrospective comparison. The open-gate fraction shows that the sign varies across individual states. Peg insertion uses the ESAC-diffusion versus SDAC success difference, and therefore measures the population comparison rather than an exploration-rule difference.

K Manipulation tasks

We evaluate ESAC on five MetaWorld v3 manipulation tasks (Yu et al., 2020, McLean et al., 2026) to characterize the regime in which ensemble exploration provides a benefit. Performance varies across tasks, with lower mean success than SDAC on `peg-insert-side-v3`. Appendix F provides additional single-seed diagnostics.

K.1 Setup and results

The tasks are reaching, drawer closing, door opening, top-down button pressing, and side peg insertion (Table 8). The first three are trained for 0.2M steps and the latter two for 0.5M. We compare SAC (Haarnoja et al., 2018), SDAC (Ma et al., 2025), and ESAC with both policy types at $K=5$ with TIDE and $\beta_0=2.0$. All methods use a single environment and 5 seeds per method and task. The generative policies use 32 action candidates per policy head, so ESAC pools 160 candidates. Success rate is the binary per-episode indicator from the MetaWorld API averaged over the final 50 training episodes.

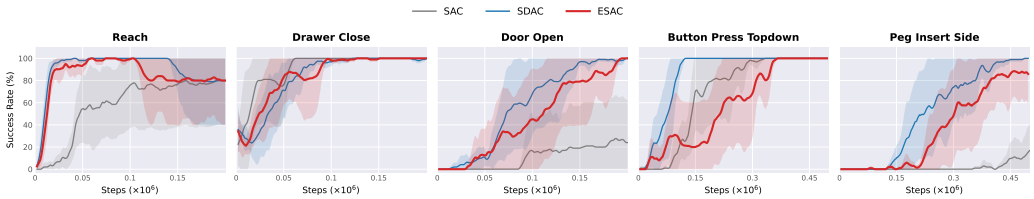

Figure 11: SDAC and ESAC diffusion approach full success on drawer closing and button pressing. Curves show dense-reward MetaWorld training success for SAC, SDAC, and ESAC diffusion, smoothed over 20 episodes. Shading shows one standard deviation across five seeds.

Table 8: SDAC and ESAC approach full success on drawer closing and button pressing, while SDAC has the highest mean success on peg insertion. Dense-reward MetaWorld success rates are reported as percentages, with mean ± standard deviation across five seeds. Green bold cells mark the largest mean per task, including ties.

Task	SAC	SDAC	ESAC (Diff.)	ESAC (FM)
reach-v3	78.8 ± 39.5	79.6 ± 39.8	$\mathbf{80.8 \pm 36.4}$	71.0 ± 40.8
drawer-close-v3	$\mathbf{100.0 \pm 0.0}$	99.2 ± 1.0	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
door-open-v3	25.2 ± 38.7	$\mathbf{98.8 \pm 2.4}$	94.4 ± 11.2	98.0 ± 4.5
button-press-topdown-v3	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$
peg-insert-side-v3	60.0 ± 49.0	$\mathbf{99.6 \pm 0.8}$	87.6 ± 24.8	88.0 ± 17.9

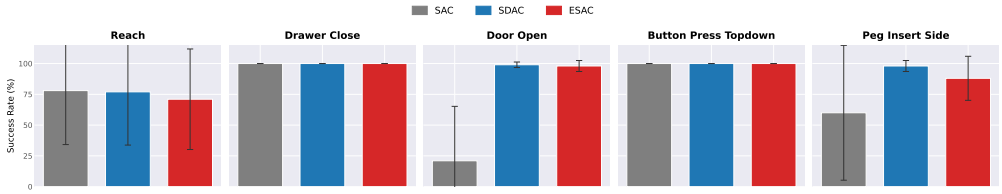

Figure 12: ESAC flow has similar mean success to SDAC on drawer closing and button pressing, but lower mean success on peg insertion. Bars show dense-reward MetaWorld success rates for SAC, SDAC, and ESAC flow, with means and standard deviations across five seeds.

SDAC and ESAC approach saturation on drawer-close and button-press-topdown, while reach has large variation across seeds. On peg-insert-side, SDAC reaches $99.6 \pm 0.8\%$ success, versus $87.6 \pm 24.8\%$ for ESAC diffusion and $88.0 \pm 17.9\%$ for ESAC flow. The ensemble is most costly on peg insertion, where lower success is accompanied by larger variation across seeds. The diagnostics in Appendix F show little useful behavioral separation at the inspected checkpoint. This makes precise manipulation an informative complement to the locomotion comparisons.