Look-ahead Variational Flow for Generative Online Reinforcement Learning

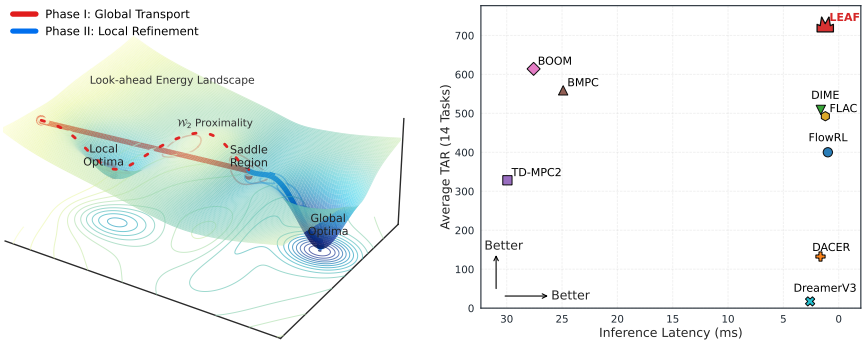
Tianyi Zhang[1†] **Likun Wang**[3†] **Tianze Zhu**[1†] **Guojian Zhan**[1] **Yinuo Wang**[4]
Feihong Zhang[1] **Yang Guan**[1] **Yao Lyu**[1*] **Shengbo Eben Li**[1,2*]

[1]School of Vehicle and Mobility, Tsinghua University
[2]College of Artificial Intelligence, Tsinghua University
[3]School of Computing, National University of Singapore
[4]2030Lab, Yinwang Intelligent Technology Co., Ltd.

[†]Equal contribution. [*]Corresponding authors.

Abstract

Generative policies have emerged as a compelling paradigm in reinforcement learning (RL) as they can represent complex and multimodal action distributions. However, this high expressivity leaves them particularly vulnerable to the inherent target distribution non-stationarity in RL, rendering the generative process severely volatile. To address this issue, we propose **L**ook-ah**E**ad v**A**riational **F**low (LEAF), a generative policy optimization framework that stabilizes both the target and the generative process. First, we replace the critic-guided target with a look-ahead target induced by predictive dynamics, which we theoretically prove to exhibit a tighter target drift bound. Second, we formulate policy improvement as a proximal variational free-energy optimization, which enforces a distributional trust region to prevent overfitting to transient target shifts. Finally, we prove that this optimization induces an optimal transport flow, which we instantiate as a two-phase coarse-to-fine generative process to achieve broad multimodal coverage and pinpoint accuracy. Extensive experiments on the DeepMind Control Suite and Humanoid Bench demonstrate that LEAF consistently outperforms strong baselines in both sample efficiency and final performance. The code is available at `https://github.com/TianyiMehdi/LEAF`.



(a) The LEAF Mechanism (b) Performance and Inference Efficiency

Figure 1: **Overview of LEAF.** (a) LEAF combines global transport and local refinement under a proximal variational flow. (b) LEAF attains the best performance with competitive inference latency across strong RL baselines.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Reinforcement learning (RL) has shown considerable potential across diverse domains such as embodied control [18, 10, 22] and large language model post-training [32, 26, 45] by enabling policy optimization through trial-and-error interaction. Generative policies, as an emerging class of policy representations in RL, parameterize the action sampling process with expressive generative models, thereby enabling effective modeling of multimodal action distributions in complex environments and significantly improving final performance [41, 28, 50]. Yet, this expressive advantage concurrently renders their optimization substantially more susceptible to target non-stationarity. While such non-stationarity is a known pathology in traditional actor-critic methods, it is drastically exacerbated in generative actors: their high-capacity nature causes them to aggressively fit a critic-induced action distribution that inherently fluctuates throughout training [9, 28, 31]. Consequently, rather than converging toward a relatively stable objective, generative actors suffer from a severe optimization mismatch as they continuously chase a moving distributional target, ultimately preventing their expressive capabilities from being fully translated into robust policy performance.

A variety of existing methods have attempted to mitigate this difficulty, but largely through indirect stabilization mechanisms. In conventional actor-critic training, target non-stationarity is typically weakened by slowly updated target networks [24], delayed policy updates and clipped double-Q estimation [9], and ensemble or distributional critic designs that suppress overestimation bias [4, 21, 8, 48]. In more recent generative policy learning, stabilization is often sought by regularizing the actor toward a relatively static reference distribution (e.g., behavior-constrained objectives or related distributional anchoring) [28, 17, 31]. While these strategies can partially alleviate training instability, they often compromise asymptotic performance and fail to directly address the critic-induced drift of the target action distribution itself. Consequently, they leave two fundamental questions unresolved: **(1)** how to construct a policy target with inherently reduced inter-iteration drift, and **(2)** how to execute a principled policy improvement and action sampling when that target inevitably evolves throughout training. For expressive generative actors, rigorously addressing these two questions is the key to resolving the persistent optimization mismatch.

To tackle these unresolved issues, we propose **L**ook-ah**E**ad v**A**riational **F**low (LEAF), a generative policy optimization framework that explicitly targets distributional drift in RL. We first move beyond critic-induced targets by constructing look-ahead targets derived from high-fidelity predictive dynamics. This yields a policy target with reduced inter-iteration drift, thereby providing a more stable anchor for the generative actor throughout training. Second, we formalize policy tracking of the target distribution as a variational free-energy minimization problem. By exploiting the proximal nature of the Jordan–Kinderlehrer–Otto (JKO) scheme [19], we derive a look-ahead variational flow that induces a natural distributional trust region and prevents the actor from overfitting to transient target shifts. Finally, we instantiate this flow as a two-phase coarse-to-fine generative process that combines broad multimodal coverage with precise local refinement. Extensive experiments demonstrate that LEAF consistently improves both sample efficiency and final performance, substantially unlocking the expressive potential of generative policies by alleviating target drift.

Our main contributions are summarized as follows:

- We identify target drift as a central bottleneck in generative RL and address it at its source by constructing a look-ahead target from predictive multi-step rollouts. This target has smoother temporal properties, mitigating the instability caused by critic-induced target updates.
- We recast policy improvement as free-energy minimization and exploit the JKO scheme to derive a proximal generative policy update with a natural distributional trust region. This proximal structure enables locally stable action generation under fluctuating target distributions.
- We show that the proximal optimization induces an optimal transport flow and instantiate it as a two-phase coarse-to-fine generative RL algorithm, LEAF, which combines global multimodal transport with local refinement and consistently achieves the strongest performance on DeepMind Control [39] and Humanoid Bench [35].

2 Preliminaries

2.1 Online Reinforcement Learning

Reinforcement learning (RL) can be formulated as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the continuous state and action spaces, $\mathcal{P}(s' \mid s, a)$ is the transition dynamics, $r(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor [38, 23]. At each time step t, the agent observes the current state $s_t \in \mathcal{S}$, samples an action $a_t \sim \pi(\cdot \mid s_t)$ from a stochastic policy π, receives a reward $r_t = r(s_t, a_t)$, and transitions to the next state $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, a_t)$. The objective of RL is to find an optimal policy π^* that maximizes the expected cumulative discounted return: $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. A central quantity is the action-value function $Q^\pi(s, a)$, which measures the expected return obtained by taking action a at state s and thereafter following policy π: $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) \mid s_t = s, a_t = a\right]$.

To balance the trade-off between exploration and exploitation, Maximum Entropy RL augments the standard objective $J(\pi)$ with an entropy regularizer $\mathcal{H}(\pi(\cdot|s))$ and has become a widely adopted framework for continuous-control policy optimization [11]. Under this perspective, policy improvement can be interpreted as an inference problem that minimizes the KL divergence between the policy and a Q-guided Boltzmann distribution:

$$\min_\pi \mathcal{D}_{\mathrm{KL}}(\pi(a|s) \parallel p_Q(a|s)), \quad p_Q(a|s) \propto \exp\left(\frac{Q(s, a)}{\tau}\right), \tag{1}$$

where $\tau > 0$ is the temperature parameter. This view recasts policy learning as approximating a target action distribution. However, the temporal difference (TD) updates of the critic $Q(s, a)$ render this target distribution inherently non-stationary, inevitably leading to severe target drift and training instability during policy optimization [36, 9].

2.2 Flow Matching

Continuous Normalizing Flows (CNFs) define a continuous-time probability path p_t from a tractable initial distribution p_0 to a complex target distribution p_1 [25, 27]. At the microscopic level, this evolution is governed by an ordinary differential equation (ODE):

$$\frac{dx_t}{dt} = v_t(x_t), \quad x_0 \sim p_0, \tag{2}$$

where $v_t(x)$ is a time-dependent vector field. Macroscopically, the corresponding evolution of the probability density $p_t(x)$ is strictly governed by the continuity equation [1]:

$$\partial_t p_t(x) + \nabla \cdot [p_t(x) v_t(x)] = 0. \tag{3}$$

Conditional Flow Matching (CFM) provides a simulation-free paradigm to train a neural vector field $v_\theta(x, t)$ to approximate the intractable marginal vector field v_t. By conditioning on individual target samples $x_1 \sim p_1$, CFM constructs optimal transport (OT) displacement paths $x_t = (1 - t)x_0 + tx_1$. The model is trained by regressing v_θ towards the conditional velocity $u_t(x_t|x_1, x_0) = x_1 - x_0$ via a simple regression objective [27]:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t\sim U[0,1], x_0\sim p_0, x_1\sim p_1} \left[\|v_\theta(x_t, t) - (x_1 - x_0)\|_2^2\right]. \tag{4}$$

2.3 Jordan-Kinderlehrer-Otto Scheme in Wasserstein space

The evolution of a probability distribution ρ that minimizes a free-energy functional $\mathcal{F}(\rho)$ can be interpreted as a steepest descent on an energy landscape within the Wasserstein space. The Jordan-Kinderlehrer-Otto (JKO) scheme [19] provides a principled time-discrete proximal approximation for this variational evolution. Starting from ρ_0, the distribution is updated via

$$\rho_{k+1} = \arg\min_{\rho\in\mathcal{P}_2} \left\{\mathcal{F}(\rho) + \frac{1}{2h}\mathcal{W}_2^2(\rho, \rho_k)\right\}, \tag{5}$$

where $\mathcal{P}_2$ is the space of probability measures with finite second moments, $\mathcal{W}_2$ is the 2-Wasserstein distance, and $h > 0$ is the proximal step size. The $\mathcal{W}_2$ penalty serves as a natural distributional trust region, ensuring that the updated distribution remains in the vicinity of the previous one. As $h \to 0$, this discrete proximal update sequence converges to a continuous variational flow.

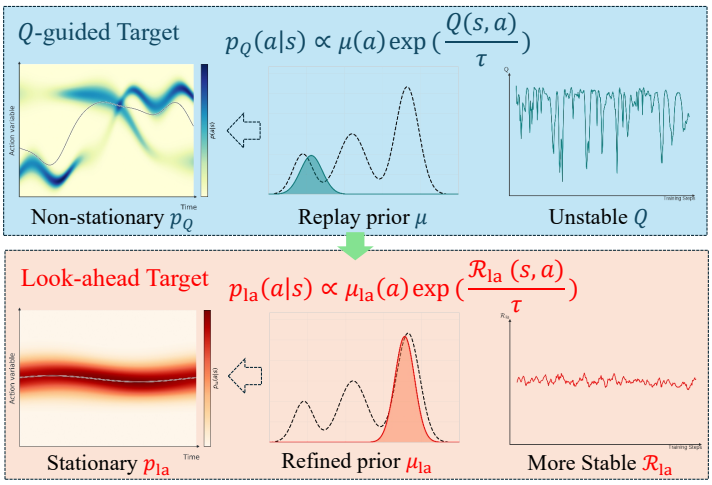


Figure 2: Q**-guided target v.s. look-ahead target.** The conventional target suffers from severe drift due to its fluctuating Q-landscape, while the replay prior μ only provides a limited local anchor. LEAF replaces Q with the more stable look-ahead return $\mathcal{R}_{\text{la}}$ and constructs a refined prior μ_{la}, yielding a more stable target distribution.

3 Method

3.1 Amplified Target Drift in Online Generative Policy Optimization

Policy optimization can be interpreted as shaping an action distribution toward high-value regions. As discussed in Sec. 2.1, maximum entropy RL further regularizes this distributional policy improvement process and can be viewed as approximating a Q-guided Boltzmann distribution. In this sense, existing generative RL can be understood as using expressive generative models to match a general critic-guided target Boltzmann distribution:

$$p(a|s) \propto \exp\left(\frac{-E(s,a)}{\tau}\right), \tag{6}$$

where $E(s,a)$ is a scalar energy. According to how this energy function is constructed, existing generative RL methods can be approximately grouped into the following two classes. (Detailed discussion and derivations are provided in the Appendix A.1)

The *first* class adopts $E(s,a) = -Q(s,a)$ [41, 3, 34], which corresponds to the standard maximum entropy RL in (1). However, the resulting purely Q-guided target is highly non-stationary, leading to severe **target drift**. In particular, the divergence between two successive target distributions admits the Q-dependent upper bound: $\mathcal{D}_{\text{KL}}(p_Q^{(k)} \parallel p_Q^{(k+1)}) \leq \frac{2}{\tau}||Q^{(k+1)} - Q^{(k)}||_\infty$ (see Appendix B.2 for detail). Therefore, as the critic fluctuates during training, the induced target can also vary sharply across iterations. In generative RL, this issue is particularly severe, as high-capacity policies can rapidly overfit such a fluctuating target landscape. The *second* class introduces prior regularization to mitigate target drift, formulating the energy as $E(s,a) = -Q(s,a) - \tau \log \mu(a)$ [7, 31, 28], which induces a prior-weighted Boltzmann distribution $p_Q(a|s) \propto \mu(a)\exp(Q(s,a)/\tau)$. Typically, the prior $\mu(a)$ is derived from the replay buffer. While this anchors policy updates to a stable region, it fails to resolve the non-stationarity inherent to the shifting Q-guided target.

To fundamentally resolve the target drift issue, we propose constructing a look-ahead return-guided target. Specifically, we utilize a learned transition dynamics $s_{i+1} = \hat{f}(s_i, a_i)$, a reward predictor $\hat{r}(s,a)$, and a terminal state value function $V(s)$ to formulate an H-step look-ahead return, which replaces the highly oscillatory bootstrapped Q as

$$\mathcal{R}_{\text{la}}(s_t, a_t) = \sum_{i=0}^{H-1} \gamma^i \hat{r}(s_{t+i}, a_{t+i}) + \gamma^H V(s_{t+H}), \quad a_{t+i} \sim \text{sg}(\pi(a|s_{t+i})), i \geq 1 \tag{7}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. In this formulation, the early-step rewards are evaluated by $\hat{r}$, which is inherently stable due to explicit supervised signals. Furthermore, the reliance

on bootstrapped Q is shifted to a more stable terminal value function V [20, 11, 12], whose influence is further exponentially suppressed by the γ^H discount. Beyond this, we use this look-ahead return in Model Predictive Path Integral (MPPI) to construct a refined prior $\mu_{\text{la}}(a)$, which provides a more adaptive and return-aware local anchor for the resulting target distribution than the buffer-derived prior (see Appendix A.2 for detail). Combining the look-ahead return and the refined prior, we define the look-ahead energy and its induced target distribution as

$$E_{\text{la}}(s,a) = -\mathcal{R}_{\text{la}}(s,a) - \tau \log \mu_{\text{la}}(a), \qquad p_{\text{la}}(a|s) \propto \mu_{\text{la}}(a) \exp\left(\frac{\mathcal{R}_{\text{la}}(s,a)}{\tau}\right). \tag{8}$$

This yields the look-ahead target that is more stable across training iterations, as illustrated in Fig. 2. We theoretically show that the look-ahead target admits a tighter inter-iteration drift bound than the standard Q-guided target.

Theorem 3.1 (Tighter Upper Bound for Look-ahead Target Drift)**.** *Under Assumptions B.2–B.5, the drift of our look-ahead target is bounded by*

$$\mathcal{D}_{\text{KL}}\left(p_{\text{la}}^{(k)} \parallel p_{\text{la}}^{(k+1)}\right) \leq \frac{2}{\tau}\left[\frac{1-\gamma^H}{1-\gamma}\epsilon_r + \epsilon_f\left(L_r \sum_{i=0}^{H-1} \gamma^i \Lambda_i + \gamma^H L_V \Lambda_H\right) + \gamma^H \|V^{(k+1)} - V^{(k)}\|_\infty\right],$$

where $\Lambda_i = \sum_{j=0}^{i-1} L_f^j$ with $\Lambda_0 = 0$, and L_f, L_r, and L_V are the Lipschitz constants of the dynamics, reward predictor, and value function, respectively. Here, ϵ_r and ϵ_f denote the temporal fluctuations of the learned reward predictor and dynamics, respectively. In the stable training regime, ϵ_r and ϵ_f approach zero due to supervised learning, and the remaining value fluctuation is attenuated by γ^H, yielding a tighter drift bound than the Q-guided target whose drift scales with $\|Q^{(k+1)} - Q^{(k)}\|_\infty$.

Proof. See Appendix B.2. □

3.2 Look-ahead Proximal Variational Flow

While the look-ahead target derived above has a tighter inter-iteration drift bound than the standard Q-guided target, it inevitably evolves across RL iterations due to the policy optimization. Directly fitting a high-capacity generative policy to such a moving target via standard KL minimization can lead to overly aggressive updates and may overwrite previously learned modes. To systematically stabilize policy improvement, we reformulate target matching as a proximal optimization problem over probability measures. We first introduce the associated free-energy objective.

Proposition 3.2. *Finding the optimal policy $\pi^*(a|s)$ that minimizes the KL divergence to the look-ahead target distribution $p_{la}(a|s)$ is equivalent to minimizing a state-dependent free-energy functional $\mathcal{F}_{la}^s(\pi_s)$, as*

$$\pi^*(a|s) = \arg\min_{\pi_s} \mathcal{D}_{KL}(\pi_s \parallel p_{la}) = \arg\min_{\pi_s} \mathcal{F}_{la}^s(\pi_s). \tag{9}$$

Here, $\mathcal{F}_{la}^s(\pi_s)$ is defined as

$$\mathcal{F}_{la}^s(\pi_s) := \int_{\mathcal{A}} E_{la}(s,a) d\pi_s(a) + \tau \int_{\mathcal{A}} \log \pi_s(a) d\pi_s(a). \tag{10}$$

Proof. See Appendix B.3. □

This reformulation places policy improvement within the Wasserstein space [2]. To prevent the actor from overfitting to transient target shifts, we apply the JKO scheme introduced in Sec. 2.3 to minimize the look-ahead free energy defined in (10). This gives the proximal update as $\pi_s^{t+h} = \arg\min_\pi \{\mathcal{F}_{\text{la}}^s(\pi) + \frac{1}{2h}\mathcal{W}_2^2(\pi, \pi_s^t)\}$. Here, the free-energy term drives the policy toward the look-ahead target, while the Wasserstein proximal term constrains excessive deviation from the current policy distribution, thereby imposing a distributional trust region in action space. To solve this proximal optimization problem, we analyze its first-order optimality condition and show that the resulting proximal update of the policy distribution admits a continuous-time flow formulation.

Theorem 3.3 (Variational Flow induced by JKO)**.** *The discrete OT map governed by the JKO proximal step asymptotically converges to an ODE $da_t = v_t(a_t)dt$. The corresponding velocity field v_t that constructs this flow is the negative gradient of the first variation of the free energy:*

$$v_t(a) = -\nabla_a \frac{\delta \mathcal{F}_{la}^s}{\delta \pi_t} = -\nabla_a E_{la}(s,a) - \tau \nabla_a \log \pi_t(a). \tag{11}$$

Proof. See Appendix B.4 □

This theorem provides principled continuous-time generative dynamics for stable policy improvement. The energy-gradient term $-\nabla_a E_{\text{la}}(s,a)$ drives actions toward high-value regions, while the score term $-\tau\nabla_a \log \pi_t(a)$ controls the distributional evolution of the policy. By further generalizing the temperature coefficient τ to a time-varying schedule $\tau(t) \in [0,1]$, the flow can smoothly trade off global multimodal transport when $\tau(t) \to 1$ and local value-guided refinement when $\tau(t) \to 0$. This yields a flexible and controllable generative mechanism for policy evolution under a moving look-ahead target.

3.3 Look-ahead Flow Policy Improvement

The variational flow in Sec. 3.2 motivates a transport-to-refinement generation principle: first move particles toward the look-ahead target, then refine them using the look-ahead gradient. We implement this transition with a decreasing sigmoid temperature schedule over $t \in [0,1]$ as $\tau(t) = \frac{1}{1+\exp(\beta(t-t_c))}$, where $t_c \in (0,1)$ controls the transition time and $\beta > 0$ controls the transition sharpness. For efficient two-phase generation, we instantiate this schedule by its hard-switch approximation:

$$\tau(t) = \begin{cases} 1, & t \in [0, t_c] \\ 0, & t \in (t_c, 1] \end{cases}, \tag{12}$$

This yields a probability flow phase for target transport, followed by a gradient flow phase for local refinement.

Phase I: Probability Flow for Global Transport ($t \in [0, t_c]$). During this stage, with $\tau(t) = 1$, the flow performs density-aware transport toward the look-ahead target. While the theoretical velocity in (11) involves an evolving score term $\nabla_a \log \pi_t(a)$, we use a uniform base $a^0 \sim \mathcal{U}(\mathcal{A})$ to render a CFM approximation principled. Since the vanishing initial score ensures the transport is purely energy-driven at $t = 0$, the learned field $v_\theta(s,a,t)$ can implicitly capture the induced dynamics for unbiased global transport [2]. Because exact sampling from $p_{\text{la}}(a|s)$ is intractable, we further apply importance sampling with $\mu_{\text{la}}(a)$ as the proposal distribution, using candidate actions $\{a_{\text{la},k}\}_{k=1}^N$ to approximate the target expectation. The resulting CFM objective is

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t\sim U[0,t_c], a^0\sim\mathcal{U}(\mathcal{A})}\big[\sum_{k=1}^{N} \tilde{w}_k \| t_c v_\theta(s, a_k^t, t) - (a_{\text{la},k} - a^0)\|_2^2\big] \tag{13}$$

where $a_k^t = (1 - t/t_c)a^0 + t/t_c \cdot a_{\text{la},k}$ and $\tilde{w}_k \propto \text{softmax}(\mathcal{R}_{\text{la}}(s, a_{\text{la},k}))$. Once trained, the generation process efficiently steers initial noise particles toward the stable trust-region manifold by integrating this learned probability flow, where $[0, t_c]$ is discretized into K_1 steps with $h_1 = t_c/K_1$:

$$a^{t+h_1} = a^t + \int_t^{t+h_1} v_\theta(s, a^\xi, \xi)\mathrm{d}\xi, \quad t \in [0, t_c].$$

Phase II: Gradient Flow for Local Refinement ($t \in (t_c, 1]$). In the second stage, the system transitions from global transport to local refinement. As $\tau(t) = 0$, the entropic and prior components vanish. The optimal velocity field naturally degenerates into the look-ahead return gradient.

To avoid the prohibitive computational cost of Backpropagation Through Time (BPTT) during online inference, we also amortize this deterministic gradient flow into a vector field $v_\phi(s,a)$. We train it by directly matching the look-ahead gradient:

$$\mathcal{L}_{\text{GF}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[\|v_\phi(s,a) - \text{sg}(\nabla_a \mathcal{R}_{\text{la}}(s,a))\|_2^2\right], \tag{14}$$

where $\mathcal{B}$ is the replay buffer. The generation process then greedily exploits the high-return modes by flowing along this learned gradient field, where $(t_c, 1]$ is discretized into K_2 steps with $h_2 = (1 - t_c)/K_2$:

$$a^{t+h_2} = a^t + \int_t^{t+h_2} v_\phi(s, a^\xi)\mathrm{d}\xi, \quad t \in (t_c, 1] \tag{15}$$

Overall, Phase I performs global multimodal target transport, moving noise particles toward broad promising regions of the look-ahead landscape. Phase II then compensates for the approximation error

of the learned probability flow by locally refining particles toward higher-return modes. Together, the two phases combine broad mode coverage with precise local refinement within the proximal policy improvement framework.

Finally, the foundations of our look-ahead target rely on the transition dynamics $\hat{f}$, the reward predictor $\hat{r}$, and the terminal value function V. We jointly optimize them as

$$\mathcal{L}_{\text{dynamics}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[c_f\left\|\hat{f}(s,a) - \text{sg}(s')\right\|_2^2 + c_r\text{CE}(\hat{r}(s,a), r) + c_v\text{CE}(V(s), r + \gamma V_-(s'))\right], \tag{16}$$

where V_- is the target value network, and $\text{CE}(\cdot,\cdot)$ denotes the cross-entropy loss over categorical representations of rewards and values to robustly handle varying signal scales. The overall training objective is then given by the weighted combination:

$$\mathcal{L}_{\text{total}} = \alpha_1\mathcal{L}_{\text{CFM}} + \alpha_2\mathcal{L}_{\text{GF}} + \alpha_3\mathcal{L}_{\text{dynamics}}.$$

4 Experiments

4.1 Experimental Setup

Baselines. We benchmark **LEAF** against five strong online RL baselines: (1) **SAC** [11]: the standard maximum entropy off-policy RL algorithm for continuous control; (2) **DACER** [41]: a representative Class I ($E = -Q$) method explicitly aligning diffusion policies with Q-value landscapes; (3) **DIME** [3]: a recent SOTA diffusion-based Class I algorithm that optimizes the pure Q-guided maximum-entropy objective via a tractable variational bound; (4) **FlowRL** [28]: a strong flow-based Class II ($E = -Q - \tau\log\mu$) algorithm mitigating target drift via Wasserstein-2 regularization against historical priors; (5) **FLAC** [29] the latest flow-based Class II SOTA that penalizes distribution shifts via kinetic-energy-regularized bridge matching. Additional comparisons with strong model-based RL baselines are provided in Appendix C. To ensure a fair comparison, we train all algorithms for 1M environment steps.

Benchmarks. We evaluate across 14 high-dimensional continuous control tasks from two benchmarks: the DeepMind Control (DMC) Suite [39], and Humanoid Bench (H-Bench) [35]. The 7 DMC tasks cover the challenging humanoid (67/24 state/action dimensions) and dog (223/38) domains, which require precise balance control and coordinated whole-body motion. The remaining 7 H-Bench tasks are built on the Unitree H1hand robot (151/61) and further increase the difficulty through long-horizon locomotion and manipulation scenarios(e.g., grasping a spoon to trace a circular pattern). Detailed task specifications are provided in Appendix D.1.

Implementation details. The detailed hyperparameters and reproducibility statement of other baselines are documented in Appendix D.2.

4.2 Experimental Results

The training curves and detailed numerical comparisons are presented in Figure 3 and Table 1, respectively. Our LEAF consistently achieves the highest Total Average Return (TAR) across the full suite of 14 high-dimensional continuous control tasks.

Results on the DMC Suite. Our LEAF achieves an average TAR of **849.3**, surpassing the prior Class I SOTA DIME (744.3) by **+14.1%**, and significantly outperforming Class II strongest FLAC (645.7) by margins of **+31.5%**. The benefits of look-ahead target construction are particularly evident in highly dynamic locomotion tasks, where pure Q-guided targets suffer from severe drift, and prior-anchored methods tend to stagnate. For instance, in the most arduous *Humanoid-run* task, LEAF achieves a score of **509.7**, improving upon the second-best DIME (231.5) by a staggering **+120.2%** and establishing a considerable margin.

Results on the H-Bench. LEAF shows even stronger advantages on H-Bench, achieving an average TAR of **603.5**, outperforming the strongest baseline FLAC (339.5) by **+77.8%** and DIME (273.5) by **+120.7%**. Notably, LEAF achieves the best performance on all seven tasks, indicating strong robustness across both locomotion and manipulation scenarios. The improvement is especially

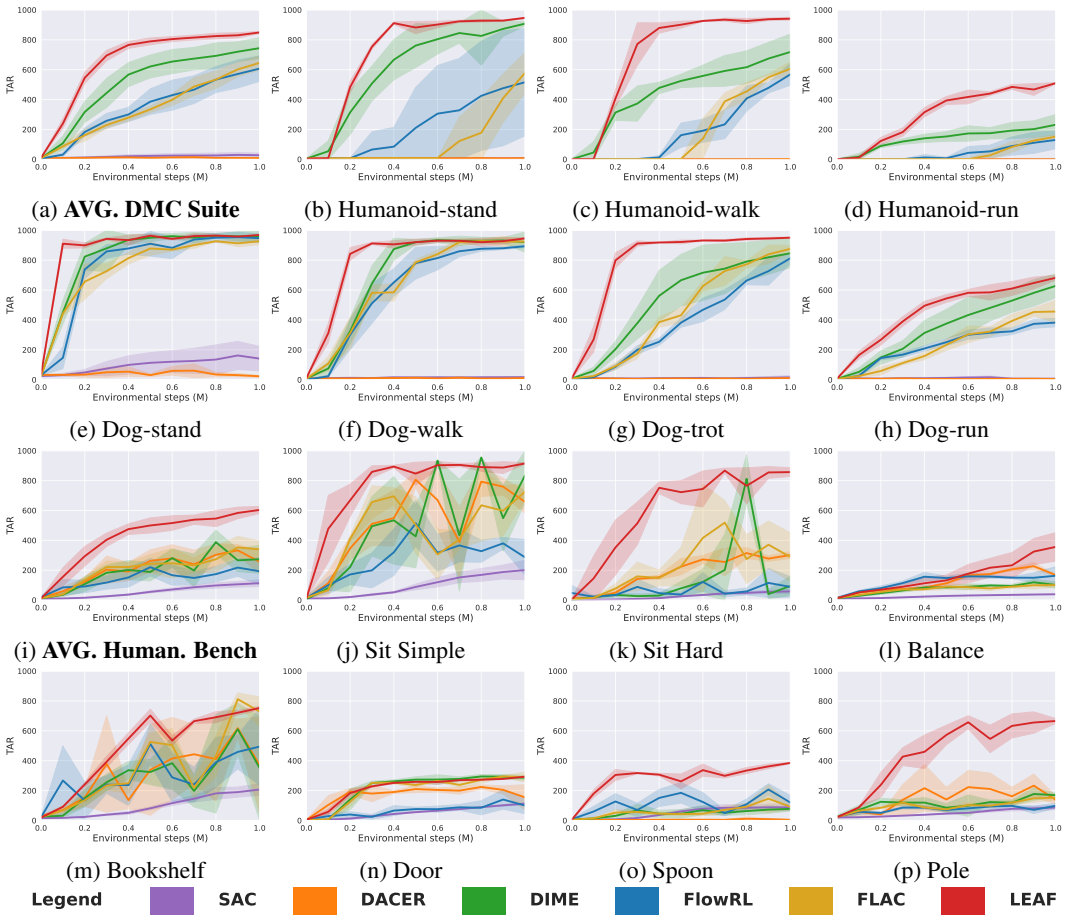


Figure 3: **Training curves on benchmarks.** The solid lines represent the mean, while the shaded regions indicate the confidence interval over five runs. The average performance curves for the two benchmarks appear at the left corner of the 1st and 3rd rows, respectively, highlighted in **bold**.

pronounced on long-horizon and precision-demanding tasks. For example, on *Sit hard*, LEAF reaches **856.7**, exceeding the second-best DACER score of 301.5 by **+184.1%**; on *Pole* and *Spoon*, LEAF improves over the second-best baselines by **+292.3%** and **+220.4%**, respectively. These results indicate that LEAF more effectively discovers high-return behaviors in rugged landscapes where standard generative RL methods tend to plateau.

Summary. Overall, Gaussian policies (SAC) and early value-driven generative actors (DACER) struggle on these high-dimensional benchmarks, highlighting the need for expressive policy representations with stable optimization. Stronger diffusion and flow-based baselines improve performance, but still rely on critic-induced targets or auxiliary regularization to stabilize policy learning. Such mechanisms do not directly reduce the target drift and may still plateau on precision-demanding tasks. By jointly constructing a more stable look-ahead target and tracking it through a proximal variational flow, **LEAF** consistently achieves higher final TAR across both DMC and H-Bench.

4.3 Ablation Study

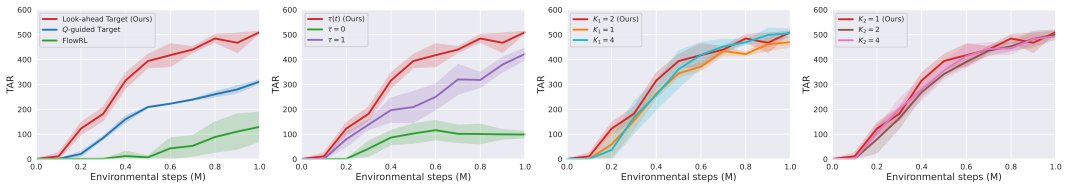
We conduct four ablations on the challenging *Humanoid-run* task, whose whole-body coordination makes it a stringent testbed for assessing each LEAF component and hyperparameter robustness.

Target Drift. Fig. 4a compares LEAF with two variants that progressively remove our drift-mitigation designs. Replacing the look-ahead target with a Q-guided target leads to a clear performance drop, showing that our look-ahead target provides a more stable optimization objective.

Table 1: Total Average Return (TAR) on 7 DMC Suite tasks and 7 H-Bench tasks. Mean ± Std over 5 seeds. **Bold** = best, underlined = second-best. Higher is better.

Task	SAC	DACER	DIME	FlowRL	FLAC	LEAF
Humanoid-stand	7.4 ± 0.9	7.6 ± 0.5	907.9 ± 15.9	515.9 ± 358.1	578.4 ± 134.7	**946.7** ± 4.8
Humanoid-walk	1.8 ± 0.4	2.1 ± 1.0	718.5 ± 119.4	568.6 ± 72.5	606.1 ± 46.7	**941.2** ± 9.2
Humanoid-run	1.2 ± 0.1	1.1 ± 0.3	231.5 ± 68.5	129.7 ± 59.0	151.5 ± 12.0	**509.7** ± 4.1
Dog-stand	141.2 ± 81.7	22.8 ± 6.0	957.6 ± 33.9	947.6 ± 29.0	927.1 ± 23.0	**968.9** ± 2.8
Dog-walk	17.1 ± 3.0	11.2 ± 3.9	920.4 ± 67.2	893.4 ± 9.5	924.4 ± 11.4	**945.9** ± 6.2
Dog-trot	16.0 ± 11.8	10.7 ± 2.9	846.1 ± 95.8	812.9 ± 37.0	876.0 ± 22.8	**950.5** ± 4.7
Dog-run	6.7 ± 2.5	4.7 ± 0.9	627.9 ± 79.7	382.1 ± 27.8	456.3 ± 75.2	**682.2** ± 16.8
AVG.DMC	27.3 ± 14.4	8.6 ± 2.2	744.3 ± 68.6	607.2 ± 84.7	645.7 ± 46.5	**849.3** ± 6.9
Sit simple	201.1 ± 64.7	657.6 ± 75.7	834.2 ± 180.5	286.3 ± 118.6	725.4 ± 52.8	**914.6** ± 5.1
Sit hard	58.2 ± 12.1	301.5 ± 22.2	94.6 ± 64.0	88.5 ± 68.9	288.2 ± 161.7	**856.7** ± 29.3
Balance simple	39.0 ± 1.5	167.4 ± 3.7	101.2 ± 21.4	162.9 ± 60.7	100.5 ± 13.0	**355.9** ± 95.2
Bookshelf simple	207.2 ± 38.9	370.9 ± 321.6	351.5 ± 348.4	494.6 ± 258.3	728.9 ± 98.1	**752.9** ± 12.0
Door	112.2 ± 7.1	154.9 ± 84.6	285.8 ± 15.7	100.3 ± 54.6	290.0 ± 34.2	**293.2** ± 2.6
Spoon	87.8 ± 21.9	4.5 ± 1.6	77.6 ± 14.3	120.2 ± 47.4	91.7 ± 17.2	**385.1** ± 4.3
Pole	86.9 ± 14.0	138.1 ± 25.9	169.7 ± 12.2	98.2 ± 16.9	152.0 ± 14.7	**665.8** ± 16.6
AVG.H-Bench	113.2 ± 22.9	256.4 ± 76.5	273.5 ± 93.8	193.0 ± 89.3	339.5 ± 55.9	**603.5** ± 23.6

Further replacing our proximal two-phase generative update with a direct KL-minimization-style flow update, i.e., FlowRL, causes an even larger degradation. These results indicate that both stable target construction and proximal generative tracking are essential for alleviating target drift.



(a) Target drift (b) Temperature schedule (c) Phase I steps (d) Phase II steps

Figure 4: **Ablation study curves.** We select the *Humanoid-run* task in DMC Suite to perform all ablation experiments.

Temperature schedule. Fig. 4b evaluates the two-phase generation controlled by $\tau(t)$. A constant $\tau = 1$ performs only probability-flow transport without final refinement, while $\tau = 0$ applies local refinement from poor initial particles. In contrast, our schedule first transports particles toward the look-ahead target and then refines them with the look-ahead gradient, achieving the best performance.

Phase-wise Steps. Fig. 4c and 4d study the sensitivity of LEAF to the discretization steps in both phases. Varying K_1 in Phase I and K_2 in Phase II yields similar final performance, showing that LEAF is robust to moderate changes in phase-wise step numbers. A very small K_1 slightly degrades performance, suggesting that sufficient global transport is helpful for reaching promising regions, whereas a larger K_2 provides only limited gains since local refinement converges quickly.

Finally, we provide additional experiments comparing LEAF with strong model-based RL algorithms, together with a detailed analysis of computational efficiency, in Appendix C and E.

5 Related Works

5.1 Generative Policy in RL

Generative models like diffusion [37] and continuous normalizing flows [25] have achieved remarkable success in imitation learning and offline RL by effectively modeling complex, multimodal action distributions [5, 44, 33]. However, extending them to online RL remains an open challenge.

Unlike offline settings with static expert datasets, online RL requires the high-capacity generative actor to continually track a moving target distribution induced by temporally bootstrapped critic [6]. This inherent susceptibility to a non-stationary target landscape inevitably leads to severe training instability. To address this optimization mismatch, existing online generative RL methods generally fall into two classes based on their target formulations. The *first* class adopts a purely Q-guided target. Methods in this category often explicitly optimize the generative process against maximum-entropy RL objectives. For instance, DACER [41] and DIME [3] perform this optimization directly in the action space, while GoRL [47] decouples it into a tractable latent space to preserve stability. Alternatively, algorithms like QSM [34] and DACERv2 [42] bypass explicit optimization by performing direct score-based sampling via Langevin dynamics. While theoretically intuitive, the resulting target is highly non-stationary; as the critic fluctuates during training, the generative actor rapidly overfits to the shifting landscape, leading to severe target drift.

To mitigate this drift, the *second* class introduces prior regularization, formulating a prior-weighted target distribution. Algorithms in this category, including QVPO [7], SDAC [31], FlowRL [28], and FLAC [29], typically implement this by evaluating a Q-weighted generative loss over historical transitions. This effectively anchors the policy to a static prior derived from the replay buffer. While this anchoring prevents catastrophic divergence, the historical prior inherently reflects past behaviors rather than currently promising actions. Consequently, it offers insufficient support for newly discovered high-return modes and fails to fundamentally resolve the non-stationarity of the Q-guided target. In contrast, LEAF constructs a lower-drift look-ahead target to mitigate Q-induced drift, while generating a high-return aligned prior to prevent historical staleness.

5.2 Model-based RL

Learning dynamics and reward models alongside policy optimization is commonly studied under the framework of model-based RL (MBRL). Existing MBRL methods can be broadly divided into *planning-driven* and *imagination-driven* paradigms according to how the learned model is used [49, 15, 14, 40]. Imagination-driven methods, represented by the Dreamer family [12–14], learn compact latent dynamics and improve the actor-critic entirely through imagined rollouts. These methods achieve strong sample efficiency by reusing synthetic experience, but their policy improvement can still be affected by compounding model errors over long imagined horizons. In contrast, planning-driven methods use the learned model for explicit online planning. The TD-MPC family [16, 15] performs short-horizon planning in a learned latent space with terminal value estimation, while recent methods such as BMPC [43] and BOOM [46] further improve planner-policy consistency to achieve strong performance in continuous control. However, these planning-driven methods typically rely on online MPPI optimization at inference time, leading to substantially higher latency. LEAF also learns predictive dynamics and rewards, but uses them to construct a more stable look-ahead target and prior during training. At deployment, action generation is fully amortized into the learned two-phase flow, avoiding online planning while retaining the benefit of model-informed policy improvement.

6 Conclusion

In this paper, we propose LEAF, a generative RL algorithm that mitigates target drift from both the target construction and policy generation perspectives. First, we replace the unstable critic-guided target with a look-ahead target, yielding a more stable objective with a tighter target drift bound. Second, we reformulate policy alignment as a proximal variational free-energy optimization problem and derive a JKO-induced transport flow that establishes a natural distributional trust region. Finally, we instantiate this flow as a two-phase generative process, combining global transport with local refinement. Extensive experiments and ablations on DMC Suite and H-Bench validate LEAF's SOTA performance, mechanistic efficacy, and hyperparameter robustness.

Acknowledgments

This work was supported by the National Science and Technology Major Project (No. 2025ZD1606200), the Tsinghua–Efort Joint Research Center for EAI Computation and Perception, and SunRising AI Lab. The authors declare no competing interests.

References

[1] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows: in metric spaces and in the space of probability measures*. Springer, 2005.

[2] Michal Balcerak, Tamaz Amiranashvili, Antonio Terpin, Suprosanna Shit, Lea Bogensperger, Sebastian Kaltenbach, Petros Koumoutsakos, and Bjoern Menze. Energy matching: unifying flow matching and energy-based models for generative modeling. *arXiv preprint arXiv:2504.10612*, 2025.

[3] Onur Celik, Zechu Li, Denis Blessing, Ge Li, Daniel Palenicek, Jan Peters, Georgia Chalvatzaki, and Gerhard Neumann. Dime: Diffusion-based maximum entropy reinforcement learning. In *International Conference on Machine Learning*, pages 6958–6977. PMLR, 2025.

[4] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.

[5] 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.

[6] Wonhyeok Choi, Shutong Ding, Minwoo Choi, Jungwan Woo, Kyumin Hwang, Jaeyeul Kim, Ye Shi, and Sunghoon Im. A review of online diffusion policy rl algorithms for scalable robotic control. *arXiv preprint arXiv:2601.06133*, 2026.

[7] 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.

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

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

[10] Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15665–15672. IEEE, 2025.

[11] 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*, pages 1861–1870. Pmlr, 2018.

[12] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[13] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

[14] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[15] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

[16] Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, pages 8387–8406. PMLR, 2022.

[17] 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.

[18] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[19] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

[20] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022.

[21] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *International conference on machine learning*, pages 5556–5566. PMLR, 2020.

[22] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, et al. Simplevla-rl: Scaling vla training via reinforcement learning. *arXiv preprint arXiv:2509.09674*, 2025.

[23] Shengbo Eben Li. Reinforcement learning for sequential decision and optimal control. *Springer*, 2023.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[25] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[26] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[27] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[29] Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, and Xiao Ma. Flac: Maximum entropy rl via kinetic energy regularized bridge matching. *arXiv preprint arXiv:2602.12829*, 2026.

[30] Yao Lyu, Xiangteng Zhang, Shengbo Eben Li, Jingliang Duan, Letian Tao, Qing Xu, Lei He, and Keqiang Li. Conformal symplectic optimization for stable reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):11049–11063, 2025.

[31] Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. In *International Conference on Machine Learning*, pages 41837–41853. PMLR, 2025.

[32] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[33] Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Forty-second International Conference on Machine Learning*, 2025.

[34] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning*, pages 41163–41182. PMLR, 2024.

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

[36] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[37] 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*, 2020.

[38] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

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

[40] Likun Wang, Xiangteng Zhang, Yinuo Wang, Guojian Zhan, Wenxuan Wang, Haoyu Gao, Jingliang Duan, and Shengbo Eben Li. Off-policy reinforcement learning with model-based exploration augmentation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[41] 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.

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

[43] Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *The Thirteenth International Conference on Learning Representations*, 2025.

[44] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[45] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[46] Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[47] Chubin Zhang, Zhenglin Wan, Feng Chen, Xingrui Yu, Ivor Tsang, and Bo An. Gorl: An algorithm-agnostic framework for online reinforcement learning with generative policies. *arXiv preprint arXiv:2512.02581*, 2025.

[48] Feihong Zhang, Guojian Zhan, Bin Shuai, Tianyi Zhang, Jingliang Duan, and Shengbo Eben Li. Distributional soft actor-critic with harmonic gradient for safe and efficient autonomous driving in multi-lane scenarios. In *2025 IEEE Intelligent Vehicles Symposium (IV)*, pages 1964–1970. IEEE, 2025.

[49] Tianyi Zhang, Likun Wang, Guojian Zhan, Feihong Zhang, Yang Guan, Yao Lyu, and Shengbo Eben Li. Langevin rollout optimization for modelic reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026.

[50] Tianze Zhu, Yinuo Wang, Wenjun Zou, Tianyi Zhang, Likun Wang, Letian Tao, Feihong Zhang, Yao Lyu, and Shengbo Eben Li. Real-time generative policy via langevin-guided flow matching for autonomous driving. *arXiv preprint arXiv:2603.02613*, 2026.

A Detailed Derivation in Methods

A.1 Classification of Generative RL Algorithms

(i) $E(s,a) = -Q(s,a)$**.**

Under this formulation, the target distribution is a purely Q-guided Boltzmann distribution:

$$p_1(a|s) = \frac{1}{Z_1(s)} \exp\left(\frac{Q(s,a)}{\tau}\right). \tag{17}$$

Depending on how the generative actor interacts with this target energy landscape, existing methods in this class can be broadly divided into two paradigms.

Paradigm 1: Explicit Policy Optimization. To train a parameterized generative policy $\pi_\theta(a|s)$ to approach this target, a standard approach is to minimize the Kullback-Leibler (KL) divergence from the policy to the target:

$$\begin{aligned} D_{\text{KL}}(\pi_\theta(\cdot|s) \parallel p_1(\cdot|s)) &= \mathbb{E}_{a\sim\pi_\theta}\left[\log \pi_\theta(a|s) - \log p_1(a|s)\right] \\ &= \mathbb{E}_{a\sim\pi_\theta}\left[\log \pi_\theta(a|s) - \frac{Q(s,a)}{\tau}\right] + \log Z_1(s). \end{aligned} \tag{18}$$

Since the partition function $Z_1(s)$ is constant with respect to the policy parameters θ, minimizing this KL divergence is mathematically equivalent to maximizing the standard maximum entropy RL objective:

$$\max_\theta \mathbb{E}_{a\sim\pi_\theta}\left[Q(s,a)\right] + \tau\mathcal{H}(\pi_\theta),$$

where $\mathcal{H}(\pi_\theta) = -\mathbb{E}_{a\sim\pi_\theta}[\log \pi_\theta(a|s)]$ is the policy entropy. This optimization paradigm is represented by methods such as DACER [41] and DIME [3], where π_θ is instantiated as a diffusion generation process and directly optimized against this RL objective via BPTT.

Paradigm 2: Direct Score-Based Sampling. Instead of explicitly fitting a parameterized policy network π_θ, another paradigm, represented by QSM [34], bypasses the optimization objective entirely. It draws actions directly from $p_1(a|s)$ during inference using Langevin dynamics:

$$a^{t+1} = a^t + \eta\nabla_a \log p_1(a^t|s) + \sqrt{2\eta}\epsilon,$$

where η is the step size and $\epsilon \sim \mathcal{N}(0, I)$. By substituting the target distribution from (17), the transient score function evaluates exactly to the scaled gradient of the action-value function (i.e., $\nabla_a \log p_1(a|s) = \frac{1}{\tau}\nabla_a Q(s,a)$). Thus, the practical generation process elegantly degenerates into flowing along the Q-gradient with injected noise:

$$a^{t+1} = a^t + \frac{\eta}{\tau}\nabla_a Q(s,a^t) + \sqrt{2\eta}\epsilon.$$

(ii) $E(s,a) = -Q(s,a) - \tau\log\mu(a)$**.**

Under this formulation, the target distribution becomes a prior-weighted Boltzmann distribution:

$$p_2(a|s) = \frac{1}{Z_2(s)}\mu(a)\exp\left(\frac{Q(s,a)}{\tau}\right). \tag{19}$$

Typically, the prior $\mu(a)$ is implicitly defined by the historical action distribution stored in the replay buffer. To train a generative policy to capture this target, the optimization objective is generally formulated as a behavior cloning (BC) loss over the target distribution $p_2(a|s)$:

$$\mathcal{L}_{\text{policy}} = \mathbb{E}_{a\sim p_2(a|s)}\left[\mathcal{L}_{\text{gen}}(a)\right], \tag{20}$$

where $\mathcal{L}_{\text{gen}}(a)$ represents the training objective for a specific generative model. Depending on the choice of the generative policy, $\mathcal{L}_{\text{gen}}(a)$ can take different forms. For Diffusion Models (e.g., DDPM): $\mathcal{L}_{\text{gen}}(a)$ is typically the noise-prediction loss:

$$\mathcal{L}^{\text{diff}}_{\text{gen}}(a) = \mathbb{E}_{t\sim U[0,1],\epsilon\sim\mathcal{N}(0,I),a\sim p_2(a|s)}\left[\|\epsilon_\theta(s,a_t,t) - \epsilon\|_2^2\right],$$

where $a_t = \sqrt{\bar{\alpha}_t}a + \sqrt{1-\bar{\alpha}_t}\epsilon$. For Flow Matching Models (e.g., CFM): $\mathcal{L}_{\text{gen}}(a)$ is the vector field regression loss:

$$\mathcal{L}_{\text{gen}}^{\text{flow}}(a) = \mathbb{E}_{t\sim U[0,1], a_0\sim\mathcal{N}(0,I), a_1\sim p_2(a|s)}\left[\|v_\theta(s,a_t,t) - (a-a_0)\|_2^2\right],$$

where $a_t = (1-t)a_0 + ta$.

However, evaluating (20) requires drawing exact samples from $p_2(a|s)$, which is computationally intractable. To resolve this, importance sampling (IS) is employed by using the replay buffer prior $\mu(a)$ as the proposal distribution. For a batch of historical transitions $\{a_k\}_{k=1}^N \sim \mu(a)$ sampled from the replay buffer, the raw importance weights are given by the density ratio:

$$w_k = \frac{p_2(a_k|s)}{\mu(a_k)} = \frac{\frac{1}{Z_2(s)}\mu(a_k)\exp\left(\frac{Q(s,a_k)}{\tau}\right)}{\mu(a_k)} = \frac{1}{Z_2(s)}\exp\left(\frac{Q(s,a_k)}{\tau}\right).$$

To eliminate the intractable partition function $Z_2(s)$, self-normalized IS over the sampled batch is applied. The normalized weight $\tilde{w}_k$ is computed as

$$\tilde{w}_k = \frac{w_k}{\sum_{j=1}^N w_j} = \frac{\exp\left(\frac{Q(s,a_k)}{\tau}\right)}{\sum_{j=1}^N \exp\left(\frac{Q(s,a_j)}{\tau}\right)} := \text{softmax}\left(\frac{Q(s,a_k)}{\tau}\right).$$

Consequently, the objective in (20) is transformed into a tractable Q-weighted generative loss evaluated on replay buffer samples:

$$\mathcal{L}_{\text{policy}} = \sum_{k=1}^N \tilde{w}_k \mathcal{L}_{\text{gen}}(a_k). \tag{21}$$

As a result, this mechanism conceptually places more probability mass on actions in high-value regions while still preserving the support of the prior distribution, thereby combining the stability of prior-anchoring with the optimality of Q-guidance. Most recent strong generative RL methods, such as QVPO [7], SDAC [31], and FlowRL [28], can be mathematically understood under this unified framework.

A.2 MPPI Implementation for Prior Construction

At each decision step, MPPI performs J iterations of trajectory optimization over a finite horizon H. The procedure at iteration j is structured as follows:

(i) Trajectory Sampling We independently sample N candidate action sequences from the current multivariate Gaussian proposal distribution:

$$a_{0:H-1}^i \sim \mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2 I), \quad i \in \{1,\dots,N\}. \tag{22}$$

(ii) Look-ahead Return Evaluation For each sampled action sequence, we simulate future states using the learned dynamics f. The look-ahead return $\mathcal{R}_{\text{la}}^i$ is evaluated by accumulating the predicted rewards $\hat{r}$ and the discounted terminal value V as

$$\mathcal{R}_{\text{la}}^i = \sum_{t=0}^{H-1} \gamma^t \hat{r}(s_t^i, a_t^i) + \gamma^H V(s_H^i). \tag{23}$$

This formulation employs the γ^H exponential weighting to suppress potential overestimation bias in the terminal value function V.

(iii) Return-Weighted Averaging To ensure numerical stability, we compute normalized importance weights w^i as

$$w^i = \frac{\exp\left((\mathcal{R}_{\text{la}}^i - \max_k \mathcal{R}_{\text{la}}^k)/\kappa\right)}{\sum_{m=1}^N \exp\left((\mathcal{R}_{\text{la}}^m - \max_k \mathcal{R}_{\text{la}}^k)/\kappa\right)}, \tag{24}$$

where κ is a temperature hyperparameter controlling the sharpness of the weighting.

(iv) Parameter Update The parameters of the action distribution are updated using the importance-weighted statistics:

$$\mu^j = \sum_{i=1}^{N} w^i a^i_{0:H-1}, \quad \sigma^j = \max\left(\sqrt{\frac{\sum_{i=1}^{N} w^i (a^i_{0:H-1} - \mu^j)^2}{\sum_{i=1}^{N} w^i}}, \epsilon\right), \tag{25}$$

where ϵ is a linearly decayed noise floor used to maintain essential exploration during training. After J iterations, the final prior is defined as $\mu_{\text{la}}(a) = \mathcal{N}(\mu_0^J, \sigma_0^J)$.

Equivalence to KL Minimization The MPPI update rule has a principled variational inference interpretation. Let the target exponential distribution induced by the return landscape be

$$p(a) \propto \exp(\mathcal{R}_{\text{la}}(a)/\kappa).$$

The goal is to fit a parameterized Gaussian distribution q_θ by minimizing the reverse KL divergence to p:

$$\min_\theta D_{\text{KL}}(p \parallel q_\theta) = \max_\theta \mathbb{E}_{a\sim p}[\log q_\theta(a)].$$

Since direct sampling from p is intractable, we rewrite the objective using importance sampling with respect to the current proposal distribution q_{old}:

$$\max_\theta \mathbb{E}_{a\sim q_{\text{old}}}\left[\frac{p(a)}{q_{\text{old}}(a)} \log q_\theta(a)\right] \approx \max_\theta \sum_{i=1}^{N} w^i \log q_\theta(a^i).$$

For a multivariate Gaussian q_θ, the solution to this weighted maximum-likelihood problem is given by the weighted sample mean and weighted covariance. Therefore, each MPPI update can be viewed as a variational step that iteratively shifts the Gaussian prior toward the high-return probability mass of the target distribution.

A.3 Weighted CFM and the JKO-Induced Transport

Endpoint distribution. The endpoint interpretation of the weighted CFM objective follows directly from the general importance-sampling derivation in Appendix A.1. Specifically, substituting $Q(s,a)$ with $\mathcal{R}_{\text{la}}(s,a)$, $\mu(a)$ with $\mu_{\text{la}}(a)$, and $\tau(t) = 1$ into (19) gives $\frac{p_{\text{la}}(a|s)}{\mu_{\text{la}}(a)} \propto \exp(\mathcal{R}_{\text{la}}(s,a))$. Consequently, the self-normalized importance weights derived in Appendix A.1 reduce exactly to the weights used in (13). Therefore, at the population level, the weighted CFM objective transports the uniform source distribution toward $p_{\text{la}}(\cdot|s)$, while (13) is its finite-sample self-normalized approximation.

Table 2: Comparison of the amortized CFM field and the explicit energy-driven field on Humanoid-run. Mean $\pm$ Std over 5 seeds. Higher is better.

Method	TAR
CFM (Ours)	509.7 ± 4.1
Energy-driven field	516.1 ± 9.2

Relation to the JKO-induced transport. Proposition 3.2 identifies $p_{\text{la}}(\cdot|s)$ as the minimizer of the look-ahead free energy, and Theorem 3.3 derives the corresponding JKO velocity in (11). At the beginning of Phase I, $\pi_0 = \mathcal{U}(\mathcal{A})$ and hence $\nabla_a \log \pi_0(a) = 0$ within its support. The initial JKO velocity therefore reduces to

$$v_0(a) = -\nabla_a E_{\text{la}}(s,a) = \nabla_a \mathcal{R}_{\text{la}}(s,a) + \nabla_a \log \mu_{\text{la}}(a). \tag{26}$$

Thus, the look-ahead return and MPPI-derived prior jointly determine the initial energy-driven JKO direction, while weighted CFM amortizes the subsequent short-horizon transport from the uniform source toward the same look-ahead target [2]. The learned CFM field is an endpoint-consistent approximation rather than a pointwise reconstruction of the complete JKO trajectory, since it does not explicitly evaluate the evolving policy score. This approximation is suitable for LEAF because Phase

I uses only a small number of integration steps, with $K_1 = 2$ by default. To evaluate it empirically, we replace the learned CFM field with the explicit energy-driven field $-\nabla_a E_{\text{la}}(s, a)$ during Phase I.

As shown in Table 2, the two implementations achieve comparable returns. This result supports the practical adequacy of weighted CFM as an efficient amortized approximation under the short Phase I transport horizon used by LEAF.

B Theoretical Analysis

B.1 Useful Lemmas

Lemma B.1. *Let $p_1(a) \propto \exp(-E_1(a)/\tau)$ and $p_2(a) \propto \exp(-E_2(a)/\tau)$ be two Boltzmann distributions defined over the same continuous action space $\mathcal{A}$. The KL divergence between them is bounded by the supremum norm of their energy difference:*

$$D_{KL}(p_1 \parallel p_2) \leq \frac{2}{\tau}\|E_1 - E_2\|_\infty \tag{27}$$

Proof. According to the definition of KL divergence, we have

$$\mathcal{D}_{\text{KL}}(p_1 \parallel p_2) = \int p_1(a) \log \frac{p_1(a)}{p_2(a)} da = \int p_1(a) \left(\frac{E_2(a) - E_1(a)}{\tau} \right) da + \log \frac{Z_2}{Z_1}. \tag{28}$$

The first integral term can be interpreted as the expectation of the energy difference under $p_1(a)$. Since for any $a \in \mathcal{A}$, the difference $E_2(a) - E_1(a)$ is strictly upper-bounded by its supremum norm $\|E_1 - E_2\|_\infty$, the expectation over any valid probability density must also be bounded by this supremum norm:

$$\int p_1(a) \left(\frac{E_2(a) - E_1(a)}{\tau} \right) da \leq \frac{1}{\tau} \int p_1(a) \|E_1 - E_2\|_\infty da = \frac{1}{\tau}\|E_1 - E_2\|_\infty. \tag{29}$$

For the log-partition ratio term, we apply a similar supremum bound technique:

$$\frac{Z_2}{Z_1} = \frac{\int \exp(-E_2(a)/\tau) da}{\int \exp(-E_1(a)/\tau) da} = \frac{\int \exp(-E_1(a)/\tau) \exp((E_1(a) - E_2(a))/\tau) da}{\int \exp(-E_1(a)/\tau) da}. \tag{30}$$

Substituting the supremum bound $(E_1(a) - E_2(a)) \leq \|E_1 - E_2\|_\infty$ into the exponent yields

$$\frac{Z_2}{Z_1} \leq \frac{\int \exp(-E_1(a)/\tau) \exp(\|E_1 - E_2\|_\infty/\tau) da}{\int \exp(-E_1(a)/\tau) da} = \exp\left(\frac{\|E_1 - E_2\|_\infty}{\tau} \right). \tag{31}$$

Taking the logarithm on both sides yields $\log(Z_2/Z_1) \leq \frac{1}{\tau}\|E_1 - E_2\|_\infty$. Summing both bounded terms completes the proof. □

B.2 Proof of Theorem 3.1

To rigorously compare the target drift, we evaluate the temporal fluctuations of the standard target and our look-ahead target between consecutive training steps k and $k+1$. We formalize the stability of the supervised reward predictor with the following assumption:

Assumption B.2 (Look-ahead Model Stability)**.** For any arbitrarily small $\epsilon_r > 0$ and $\epsilon_f > 0$, there exists a training step K such that for all $k > K$, the learned reward predictor and transition dynamics satisfy

$$\|\hat{r}^{(k+1)} - \hat{r}^{(k)}\|_\infty < \epsilon_r, \qquad \|\hat{f}^{(k+1)} - \hat{f}^{(k)}\|_\infty < \epsilon_f.$$

This assumption is reasonable, as both $\hat{r}$ and $\hat{f}$ are trained via explicit supervised learning and therefore tend to converge rapidly and stably.

Assumption B.3 (Prior Stationarity)**.** Because the replay buffer is static over short time scales and MPPI exhibits local consistency, the corresponding priors (μ and μ_{la}) change only negligibly across two consecutive optimization steps. As a result, their contributions to the temporal energy difference approximately cancel, and we omit these prior-dependent terms in the following analysis.

Assumption B.4 (Lipschitz Rollout Components)**.** The learned transition model, reward predictor, and terminal value function are Lipschitz continuous with respect to the state input. Specifically, for all admissible states s_1, s_2 and actions a,

$$\|\hat{f}^{(k)}(s_1, a) - \hat{f}^{(k)}(s_2, a)\| \leq L_f\|s_1 - s_2\|, \quad |\hat{r}^{(k)}(s_1, a) - \hat{r}^{(k)}(s_2, a)| \leq L_r\|s_1 - s_2\|,$$

and

$$|V^{(k)}(s_1) - V^{(k)}(s_2)| \leq L_V\|s_1 - s_2\|.$$

Assumption B.5 (State-Value Stability)**.** Compared with the action-value function $Q(s, a)$, the state-value function $V(s)$ typically evolves more smoothly over training and exhibits lower temporal variance. This is because $Q(s, a)$ must capture action-specific fluctuations, whereas $V(s)$ aggregates over actions and therefore provides a smoother baseline. We thus assume that, throughout the optimization process,

$$\|V^{(k+1)} - V^{(k)}\|_\infty < \|Q^{(k+1)} - Q^{(k)}\|_\infty.$$

(i) Drift bound of the Q-guided target. The standard Q-guided target Boltzman distribution is $p_Q(a|s) \propto \mu(a)\exp(Q^{(k)}(s,a)/\tau)$. Given Assumption B.3, the prior term remains static, so the temporal energy difference evaluates exclusively to the fluctuation of the Q-network:

$$||E_Q^{(k+1)} - E_Q^{(k)}||_\infty = ||-Q^{(k+1)} - (-Q^{(k)})||_\infty = ||Q^{(k+1)} - Q^{(k)}||_\infty.$$

Applying Lemma B.1, the target drift for the Q-guided target is bounded by:

$$\mathcal{D}_{\text{KL}}(p_Q^{(k)} \parallel p_Q^{(k+1)}) \leq \frac{2}{\tau}||Q^{(k+1)} - Q^{(k)}||_\infty.$$

(ii) Drift bound of the look-ahead target. Our LEAF constructs the target using the look-ahead return: $p_{\text{la}}(a|s) \propto \mu_{\text{la}}(a)\exp(\mathcal{R}_{\text{la}}(s,a)/\tau)$. Given Assumption B.3, the temporal energy difference depends only on the look-ahead return:

$$||E_{\text{la}}^{(k+1)} - E_{\text{la}}^{(k)}||_\infty = ||\mathcal{R}_{\text{la}}^{(k+1)} - \mathcal{R}_{\text{la}}^{(k)}||_\infty.$$

We now bound the fluctuation of the look-ahead return. For a fixed initial state-action pair and a fixed candidate action sequence $a_{0:H-1}$, define the model rollout at training step k as $s_{i+1}^{(k)} = \hat{f}^{(k)}(s_i^{(k)}, a_i)$, with $s_0^{(k)} = s_0$. The corresponding H-step look-ahead return is

$$\mathcal{R}_{\text{la}}^{(k)} = \sum_{i=0}^{H-1} \gamma^i \hat{r}^{(k)}(s_i^{(k)}, a_i) + \gamma^H V^{(k)}(s_H^{(k)}).$$

Let $\Delta_i = \|s_i^{(k+1)} - s_i^{(k)}\|$. Since both rollouts start from the same initial state, $\Delta_0 = 0$. By Assumptions B.2 and B.4,

$$\Delta_{i+1} \leq \|\hat{f}^{(k+1)}(s_i^{(k+1)}, a_i) - \hat{f}^{(k+1)}(s_i^{(k)}, a_i)\| + \|\hat{f}^{(k+1)}(s_i^{(k)}, a_i) - \hat{f}^{(k)}(s_i^{(k)}, a_i)\| \leq L_f\Delta_i + \epsilon_f.$$

Unrolling this recursion gives

$$\Delta_i \leq \epsilon_f \Lambda_i, \qquad \Lambda_i := \sum_{j=0}^{i-1} L_f^j, \qquad \Lambda_0 = 0.$$

Next, the reward fluctuation at each rollout step is bounded by

$$\begin{aligned}|\hat{r}^{(k+1)}(s_i^{(k+1)}, a_i) - \hat{r}^{(k)}(s_i^{(k)}, a_i)| &\leq |\hat{r}^{(k+1)}(s_i^{(k+1)}, a_i) - \hat{r}^{(k+1)}(s_i^{(k)}, a_i)| + |\hat{r}^{(k+1)}(s_i^{(k)}, a_i) - \hat{r}^{(k)}(s_i^{(k)}, a_i)| \\ &\leq L_r\Delta_i + \epsilon_r \leq L_r\epsilon_f\Lambda_i + \epsilon_r.\end{aligned}$$

Similarly, the terminal value fluctuation satisfies

$$\begin{aligned}|V^{(k+1)}(s_H^{(k+1)}) - V^{(k)}(s_H^{(k)})| &\leq |V^{(k+1)}(s_H^{(k+1)}) - V^{(k+1)}(s_H^{(k)})| + |V^{(k+1)}(s_H^{(k)}) - V^{(k)}(s_H^{(k)})| \\ &\leq L_V\Delta_H + \|V^{(k+1)} - V^{(k)}\|_\infty \leq L_V\epsilon_f\Lambda_H + \|V^{(k+1)} - V^{(k)}\|_\infty.\end{aligned}$$

Combining the above bounds yields

$$\begin{aligned}|\mathcal{R}_{\text{la}}^{(k+1)} - \mathcal{R}_{\text{la}}^{(k)}| &\leq \sum_{i=0}^{H-1} \gamma^i(\epsilon_r + L_r\epsilon_f\Lambda_i) + \gamma^H(L_V\epsilon_f\Lambda_H + \|V^{(k+1)} - V^{(k)}\|_\infty) \\ &= \frac{1-\gamma^H}{1-\gamma}\epsilon_r + \epsilon_f\left(L_r\sum_{i=0}^{H-1}\gamma^i\Lambda_i + \gamma^H L_V\Lambda_H\right) + \gamma^H\|V^{(k+1)} - V^{(k)}\|_\infty.\end{aligned}$$

Taking the supremum over all admissible initial state-action pairs, we obtain

$$\|\mathcal{R}_{\text{la}}^{(k+1)} - \mathcal{R}_{\text{la}}^{(k)}\|_\infty \leq \frac{1-\gamma^H}{1-\gamma}\epsilon_r + \epsilon_f\left(L_r\sum_{i=0}^{H-1}\gamma^i\Lambda_i + \gamma^H L_V\Lambda_H\right) + \gamma^H\|V^{(k+1)} - V^{(k)}\|_\infty.$$

Applying Lemma B.1, we establish the drift bound for the look-ahead target:

$$\mathcal{D}_{\text{KL}}(p_{\text{la}}^{(k)} \parallel p_{\text{la}}^{(k+1)}) \leq \frac{2}{\tau}\left[\frac{1-\gamma^H}{1-\gamma}\epsilon_r + \epsilon_f\left(L_r\sum_{i=0}^{H-1}\gamma^i\Lambda_i + \gamma^H L_V\Lambda_H\right) + \gamma^H\|V^{(k+1)} - V^{(k)}\|_\infty\right].$$

Final Comparison. We now compare the derived upper bounds. The standard Q-guided target satisfies

$$\mathcal{D}_{\mathrm{KL}}(p_Q^{(k)} \parallel p_Q^{(k+1)}) \leq \frac{2}{\tau}\|Q^{(k+1)} - Q^{(k)}\|_\infty,$$

whereas the look-ahead target satisfies

$$\mathcal{D}_{\mathrm{KL}}(p_{\mathrm{la}}^{(k)} \parallel p_{\mathrm{la}}^{(k+1)}) \leq \frac{2}{\tau}\left[\frac{1-\gamma^H}{1-\gamma}\epsilon_r + \epsilon_f\left(L_r\sum_{i=0}^{H-1}\gamma^i\Lambda_i + \gamma^H L_V\Lambda_H\right) + \gamma^H\|V^{(k+1)} - V^{(k)}\|_\infty\right].$$

As training progresses into the stable regime, the supervised reward predictor and transition model converge, yielding $\epsilon_r \to 0$ and $\epsilon_f \to 0$. Furthermore, by Assumption B.5, the temporal fluctuation of the state-value function is smaller than that of the action-value function, and this remaining value fluctuation is further attenuated by the multi-step discount factor γ^H. Consequently, for sufficiently large $k > K$,

$$\frac{1-\gamma^H}{1-\gamma}\epsilon_r + \epsilon_f\left(L_r\sum_{i=0}^{H-1}\gamma^i\Lambda_i + \gamma^H L_V\Lambda_H\right) + \gamma^H\|V^{(k+1)} - V^{(k)}\|_\infty < \|Q^{(k+1)} - Q^{(k)}\|_\infty.$$

This implies

$$\sup\mathcal{D}_{\mathrm{KL}}(p_{\mathrm{la}}^{(k)} \parallel p_{\mathrm{la}}^{(k+1)}) < \sup\mathcal{D}_{\mathrm{KL}}(p_Q^{(k)} \parallel p_Q^{(k+1)}).$$

Therefore, in the stable training regime, the derived upper bound for the look-ahead target drift is tighter than the corresponding Q-guided upper bound, completing the proof.

B.3 Proof of Proposition 3.2

The look-ahead target distribution in (8) can be written as

$$p_{\mathrm{la}}(a|s) = \frac{1}{Z(s)}\exp\left(-\frac{E_{\mathrm{la}}(s,a)}{\tau}\right), \tag{32}$$

where $Z(s) = \int_{\mathcal{A}}\exp\left(-\frac{E_{\mathrm{la}}(s,a)}{\tau}\right)da$ is the partition function that normalizes the target distribution. The KL divergence from π_s to the target distribution p_{la} is defined as:

$$\mathcal{D}_{KL}(\pi_s \parallel p_{\mathrm{la}}) = \int_{\mathcal{A}}\pi_s(a)\log\left(\frac{\pi_s(a)}{p_{\mathrm{la}}(a|s)}\right)da.$$

Substituting (32) into the KL divergence yields

$$\begin{aligned}\mathcal{D}_{KL}(\pi_s \parallel p_{\mathrm{la}}) &= \int_{\mathcal{A}}\pi_s(a)\left[\log\pi_s(a) - \log p_{\mathrm{la}}(a|s)\right]da \\ &= \int_{\mathcal{A}}\pi_s(a)\left[\log\pi_s(a) + \frac{E_{\mathrm{la}}(s,a)}{\tau} + \log Z(s)\right]da \\ &= \frac{1}{\tau}\underbrace{\left[\int E_{\mathrm{la}}(s,a)d\pi_s(a) + \tau\int\log\pi_s(a)d\pi_s(a)\right]}_{\mathcal{F}_{\mathrm{la}}^s(\pi_s)} + \log Z(s).\end{aligned} \tag{33}$$

The bracketed component matches the definition of the free-energy $\mathcal{F}_{\mathrm{la}}^s(\pi_s)$. Thus, we have

$$\mathcal{D}_{KL}(\pi_s \parallel p_{\mathrm{la}}) = \frac{1}{\tau}\mathcal{F}_{\mathrm{la}}^s(\pi_s) + \log Z(s).$$

Since τ acts merely as a scaling factor, and the log-partition function $\log Z(s)$ is independent of π_s, the optimal policy that minimizes the KL divergence is identical to the one that minimizes the free-energy functional:

$$\arg\min_{\pi_s}\mathcal{D}_{KL}(\pi_s \parallel p_{\mathrm{la}}) = \arg\min_{\pi_s}\mathcal{F}_{\mathrm{la}}^s(\pi_s).$$

B.4 Proof of Theorem 3.3

Let π_{t+h} be the optimal minimizer. For any smooth, compactly supported vector field $\xi \in C_c^\infty(\mathcal{A}; \mathbb{R}^d)$, we construct a push-forward perturbation $\pi_\varepsilon = (\Phi_\varepsilon)_\# \pi_{t+h}$ via the map $\Phi_\varepsilon(a) = a + \varepsilon\xi(a)$. Since π_{t+h} is the minimizer, its first variation evaluated at $\varepsilon = 0$ vanish:

$$\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0} \left[\mathcal{F}^s_{\text{la}}(\pi_\varepsilon) + \frac{1}{2h}\mathcal{W}_2^2(\pi_\varepsilon, \pi_t) \right] = 0.$$

First, we analyze the variation of the free energy. Using the definition of the first variation $\frac{\delta \mathcal{F}}{\delta \pi}$ and the continuity equation of the push-forward perturbation $\frac{d}{d\varepsilon}\big|_{\varepsilon=0} \pi_\varepsilon = -\nabla_a \cdot (\pi_{t+h}\xi)$, applying integration by parts yields

$$\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0} \mathcal{F}^s_{\text{la}}(\pi_\varepsilon) = \int_{\mathcal{A}} \nabla_a \left(\frac{\delta \mathcal{F}^s_{\text{la}}}{\delta \pi}(\pi_{t+h}) \right) \cdot \xi(a)\, \pi_{t+h}(a) da.$$

Next, we evaluate the variation of the Wasserstein penalty. Assuming absolute continuity, Brenier's Theorem guarantees a unique optimal transport map $T : \mathcal{A} \to \mathcal{A}$ such that $T_\# \pi_{t+h} = \pi_t$, which represents the backward optimal transport map from the updated distribution π_{t+h} back to the previous distribution π_t. Using $T \circ \Phi_\varepsilon^{-1}$ as a candidate transport map for the perturbed measure, the variation of the transport cost at $\varepsilon = 0$ evaluates to

$$\frac{d}{d\varepsilon}\bigg|_{\varepsilon=0} \frac{1}{2h}\mathcal{W}_2^2(\pi_\varepsilon, \pi_t) = \frac{1}{h}\int_{\mathcal{A}} (a - T(a)) \cdot \xi(a)\, \pi_{t+h}(a) da.$$

Summing these two variations and recognizing that the equality holds for any arbitrary test vector field ξ, we obtain the point-wise first-order optimality condition for the JKO step:

$$T(a) - a = h\nabla_a \left(\frac{\delta \mathcal{F}^s_{\text{la}}}{\delta \pi}(\pi_{t+h})(a) \right) \tag{34}$$

This equation dictates the optimal displacement of particles returning from their new state to their prior state.

Finally, to construct the forward generative process, we analyze the forward optimal transport map $S = T^{-1}$, which pushes π_t to π_{t+h}. For a particle $a_t \sim \pi_t$, its subsequent position is $a_{t+h} = S(a_t)$. Substituting the backward map condition $T(S(a_t)) = a_t$ to (34) yields the forward particle displacement:

$$S(a_t) - a_t \approx -h\nabla_a \left(\frac{\delta \mathcal{F}^s_{\text{la}}}{\delta \pi}(\pi_t)(a_t) \right).$$

In the deterministic limit ($h \to 0$), the discrete optimal transport displacement scales linearly with h, defining an instantaneous velocity field as

$$v_t(a_t) = \lim_{h\to 0} \frac{S(a_t) - a_t}{h} = -\nabla_a \frac{\delta \mathcal{F}^s_{\text{la}}}{\delta \pi_t}. \tag{35}$$

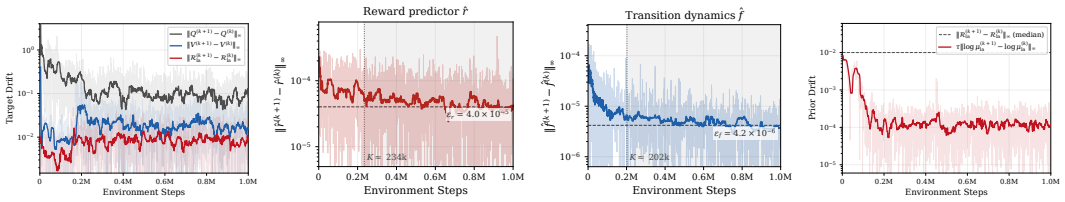
Consequently, the probability measure evolves along continuous trajectories governed by the ODE:

$$\frac{da_t}{dt} = v_t(a_t).$$

By substituting the free energy $\mathcal{F}^s_{\text{la}}(\pi_t)$ in (10) to (35), we explicitly recover the continuous velocity field:

$$v_t(a) = -\nabla_a E_{\text{la}}(s, a) - \tau\nabla_a \log \pi_t(a).$$

This rigorous derivation confirms that the discrete JKO proximal sequence asymptotically converges to an optimal transport flow, completing the proof.



(a) Target fluctuations (b) Reward stability (c) Dynamics stability (d) MPPI-prior stability

Figure 5: **Empirical validation of the temporal-stability assumptions on Humanoid-run.** (a) Consecutive-update fluctuations of the action value function Q, state value function V, and look-ahead return $\mathcal{R}_{\text{la}}$. (b)–(c) Temporal fluctuations of the learned reward predictor and transition dynamics. (d) The MPPI-prior fluctuation compared with the median look-ahead-return fluctuation. Dark curves show the smoothed trends, while light curves show the corresponding raw measurements.

B.5 Empirical Validation of the Assumptions Underlying Theorem 3.1

Theorem 3.1 establishes the target-drift bound under a set of sufficient conditions on the temporal stability of the learned rollout components, the prior distribution, and the terminal value function. To assess whether these assumptions are satisfied in practice, we empirically measure the corresponding consecutive-update fluctuations throughout training on the Humanoid-run task. In particular, we examine the quantities associated with Assumptions B.2, B.3, and B.5, including the early training stage in which target non-stationarity is most pronounced. Assumption B.4 is a standard regularity condition on the learned functions and is therefore not directly evaluated here.

Figure 5a compares the temporal fluctuations of the quantities that define the conventional Q-guided and look-ahead targets. Throughout training, including the early stage, the fluctuation of the look-ahead return remains consistently below that of the action-value function, typically by one to two orders of magnitude. Moreover, the state value fluctuation remains below the action-value fluctuation, empirically supporting the stability relation in Assumption B.5. These observations indicate that replacing the directly bootstrapped action-value landscape with a multi-step look-ahead return provides a substantially smoother target signal in the evaluated setting.

Figures 5b and 5c examine the temporal stability of the supervised rollout components in Assumption B.2. The consecutive-update fluctuation of the reward predictor decreases below $\epsilon_r = 4.0 \times 10^{-5}$ after approximately $K = 234$k environment steps, while the transition-dynamics fluctuation decreases below $\epsilon_f = 4.2 \times 10^{-6}$ after approximately $K = 202$k steps. Both quantities remain small thereafter. This behavior is consistent with the stable-training regime considered in Theorem 3.1, in which the supervised reward and dynamics models contribute only small temporal perturbations to the look-ahead return.

Finally, Figure 5d measures the temporal fluctuation of the MPPI-derived prior. After the initial training stage, the prior-dependent term $\tau|\log \mu_{\text{la}}^{(k+1)} - \log \mu_{\text{la}}^{(k)}|_\infty$ stabilizes around 10^{-4}, approximately two orders of magnitude below the median look-ahead-return fluctuation of 10^{-2}. This result empirically supports the short-timescale prior-stationarity condition in Assumption B.3 and indicates that the temporal variation of the look-ahead target is dominated by its return component rather than by abrupt changes in the refined prior.

Overall, these measurements provide empirical evidence that the sufficient temporal-stability conditions used in Theorem 3.1 hold in the evaluated online RL setting. They also corroborate the central mechanism underlying LEAF: the look-ahead construction produces a substantially smoother target-defining signal than the conventional Q-guided alternative.

C Experimental Results with Model-based RL algorithms

To further evaluate the performance of LEAF, we compare it with four representative model-based reinforcement learning (MBRL) baselines: (1) **DreamerV3** [14], a SOTA imagination-driven MBRL algorithm; (2) **TD-MPC2** [15], a scalable planning-driven MBRL method that performs online planning via MPPI; (3) **BMPC** [43], an improved TD-MPC2 variant that enhances policy learning through planner-guided imitation; and (4) **BOOM** [46], a recent SOTA planning-driven MBRL algorithm that couples online planning with off-policy learning via bootstrap alignment between the planner and policy.

As shown in Fig. 6 and Table 3, LEAF still achieves the best average performance on both benchmarks when compared with these strong MBRL baselines. On the DMC Suite, LEAF obtains an average TAR of **849.3**, outperforming the strongest MBRL baseline BOOM (803.1) by **+5.8%**. On H-Bench, the advantage is more pronounced: LEAF reaches an average TAR of **603.5**, substantially exceeding BOOM (425.3) by **+41.9%** and BMPC (425.1) by **+42.0%**. These results show that LEAF remains highly competitive even against planning-based MBRL methods that explicitly leverage learned dynamics for online decision making.

Moreover, LEAF is significantly more efficient at inference. Planning-driven methods such as BMPC and BOOM require repeated online look-ahead optimization and take around 25 ms per action, whereas LEAF amortizes the look-ahead improvement into a generative policy and only requires about **1.2 ms** per action. This demonstrates that LEAF achieves stronger performance while avoiding the high inference cost of online planning.

Table 3: Total Average Return (TAR) on 7 DMC Suite tasks and 7 H-Bench tasks. Mean ± Std over 5 seeds. **Bold** = best, underlined = second-best. Higher is better.

Task	DreamerV3	TD-MPC2	BMPC	BOOM	LEAF
Humanoid-stand	4.5 ± 0.5	663.7 ± 26.5	700.0 ± 69.6	909.6 ± 16.8	**946.7** ± 4.8
Humanoid-walk	1.6 ± 0.1	628.2 ± 13.3	701.3 ± 102.6	901.4 ± 19.9	**941.2** ± 9.2
Humanoid-run	0.8 ± 0.2	184.6 ± 19.4	257.7 ± 125.6	443.0 ± 18.2	**509.7** ± 4.1
Dog-stand	20.6 ± 1.8	798.9 ± 23.3	896.3 ± 105.7	898.5 ± 109.2	**968.9** ± 2.8
Dog-walk	6.1 ± 1.2	493.9 ± 311.4	911.3 ± 34.6	**952.1** ± 4.4	945.9 ± 6.2
Dog-trot	6.5 ± 1.4	500.0 ± 13.6	888.7 ± 30.1	863.2 ± 46.6	**950.5** ± 4.7
Dog-run	3.1 ± 2.2	169.9 ± 101.0	494.5 ± 84.2	654.0 ± 36.1	**682.2** ± 16.8
AVG. DMC Suite.	6.2 ± 1.1	491.3 ± 72.6	692.8 ± 78.9	803.1 ± 35.9	**849.3** ± 6.9
Sit Simple	47.9 ± 4.4	607.1 ± 244.9	537.5 ± 46.1	887.7 ± 14.1	**914.6** ± 5.1
Sit Hard	15.4 ± 3.2	139.2 ± 38.5	546.4 ± 102.8	267.7 ± 32.0	**856.7** ± 29.3
Balance	12.6 ± 0.5	24.9 ± 1.2	**366.9** ± 67.6	72.2 ± 5.2	355.9 ± 95.2
Bookshelf	45.3 ± 3.1	96.6 ± 27.0	403.7 ± 147.4	730.0 ± 15.4	**752.9** ± 12.0
Door	10.1 ± 2.4	134.4 ± 80.7	210.0 ± 26.2	14.5 ± 1.3	**293.2** ± 2.6
Spoon	20.5 ± 4.6	54.5 ± 11.7	343.0 ± 3.4	307.4 ± 13.2	**385.1** ± 4.3
Pole	47.8 ± 8.9	98.7 ± 8.8	568.0 ± 38.3	**687.6** ± 56.9	665.8 ± 16.6
AVG. H-Bench.	28.5 ± 3.9	165.1 ± 59.0	425.1 ± 61.7	425.3 ± 19.7	**603.5** ± 23.6

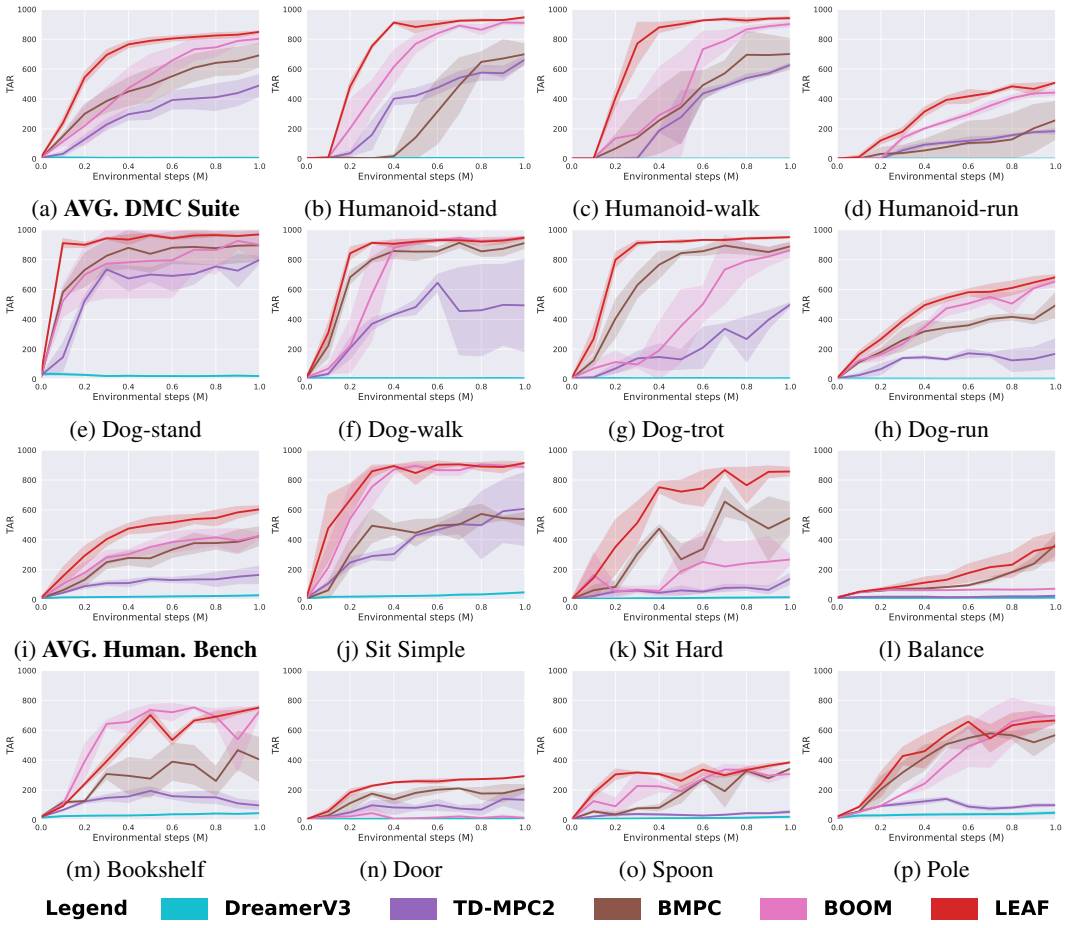


Figure 6: **Training curves Compared with MBRL baselines.** The solid lines represent the mean, while the shaded regions indicate the confidence interval over five runs. The average performance curves for the two benchmarks appear in the left corner of the 1st and 3rd rows, respectively, highlighted in **bold**.

D Environmental Details

D.1 Benchmark Introduction

DeepMind Control Suite. The evaluation suite consists of the 7 most challenging control tasks for the *dog* and *humanoid* domains, divided into: (1) *Standing tasks*, requiring the preservation of upright equilibrium, and (2) *Moving tasks*, which entail the additional objective of reaching a target velocity. For tasks involving movement, we compute the total reward by scaling the posture reward with the speed tracking reward: ***Reward*** $=$ (***Standing reward***) $\times$ (***Forward velocity reward***).



Figure 7: Dog

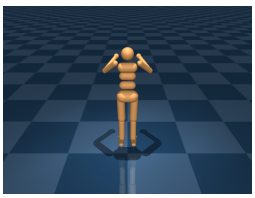


Figure 8: Humanoid

Standing reward: Encourages the agent to maintain an upright posture.

Forward velocity reward: Ensures the agent moves at the target speed ($1\,\mathrm{m/s}$ for dog-walk, $3\,\mathrm{m/s}$ for dog-trot, $9\,\mathrm{m/s}$ for dog-run, $1\,\mathrm{m/s}$ for humanoid-walk and $10\,\mathrm{m/s}$ for humanoid-run).

Humanoid Bench. We evaluate our method on 7 representative Humanoid Bench tasks with the Unitree H1 humanoid equipped with dexterous hands, covering locomotion, stabilization, and whole-body manipulation. The robot is generally initialized from a standing configuration with small joint perturbations, while task-specific randomization is applied when required. The task objectives, reward structures, and termination conditions are summarized below.



Figure 9: Sit Simple

Objective. Sit onto a chair placed closely behind the robot.
Reward: $R(s,a) = \texttt{control} \times \texttt{sit} \times \texttt{still}$, where $\texttt{sit} = (0.5 \times \texttt{sitting}_z + 0.5 \times \texttt{on_chair}) \times \texttt{upright} \times \texttt{posture}$. The reward encourages the robot to lower its pelvis onto the chair while maintaining an upright, stable, and stationary seated posture.
Termination. 1000 steps, or when $z_{\text{pelvis}} < 0.5$.



Figure 10: Sit Hard

Objective. Sit onto a chair under randomized initial body pose and chair configuration.
Reward: $R(s,a) = \texttt{control} \times \texttt{sit} \times \texttt{still}$, where $\texttt{sit} = (0.5 \times \texttt{sitting}_z + 0.5 \times \texttt{on_chair}) \times \texttt{upright} \times \texttt{posture}$. Compared with *Sit Simple*, this variant randomizes the robot's initial orientation and position, making the same seated-pose objective require more robust whole-body reorientation and stabilization.
Termination. 1000 steps, or when $z_{\text{pelvis}} < 0.5$.

D.2 Reproducibility Statement & Detailed Hyperparameters

We provide the implementation of LEAF in the supplementary material. The detailed hyperparameter configuration of our algorithm is reported in Table 4.

In this paper, we evaluate each algorithm on each task over five random seeds. All experiments are conducted on a workstation equipped with an Intel(R) Xeon(R) Gold 6530 32-core CPU and an NVIDIA GeForce RTX 5090 GPU. Taking the *Humanoid-run* task as an example, training LEAF for 1M environment steps takes approximately 17 hours.

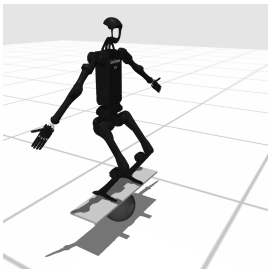


Figure 11: Balance

Objective. Maintain balance on an unstable board with a fixed spherical pivot.
Reward: $R(s,a) = \texttt{control} \times \texttt{stable} \times \texttt{still}$, where $\texttt{stable} = \texttt{height} \times \texttt{upright}$. The reward encourages the robot to remain upright and stationary while avoiding excessive control effort on the unstable platform.
Termination. 1000 steps, or when $z_{\text{pelvis}} < 0.8$, or when the board loses valid contact with the support.



Figure 12: Bookshelf

Objective. Pick and place multiple objects across the bookshelf in a prescribed order.
Reward: $R(s,a) = 0.2 \times \texttt{stable} + 0.4 \times \texttt{object_goal} + 0.4 \times \texttt{hand_object}$. Here, $\texttt{object_goal}$ encourages placing the active object near its target shelf location, while $\texttt{hand_object}$ guides the hands toward the object.
Termination. 1000 steps, when all objects are placed, when $z_{\text{pelvis}} < 0.58$, or when the active object falls.

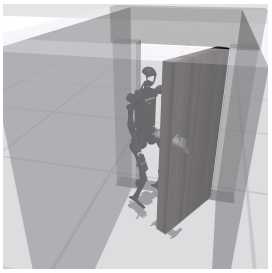


Figure 13: Door

Objective. Pull the door open using the doorknob and move through the doorway while keeping the door open.
Reward: $R(s,a) = 0.1 \times \texttt{stable} + 0.45 \times \texttt{open_door} + 0.05 \times \texttt{open_hatch} + 0.05 \times \texttt{hand_hatch} + 0.35 \times \texttt{passage}$. The reward encourages reaching the handle, rotating the latch, opening the door, and moving through the doorway while staying balanced.
Termination. 1000 steps, or when $z_{\text{pelvis}} < 0.58$.

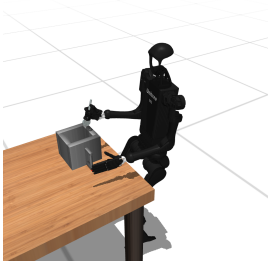


Figure 14: Spoon

Objective. Grab a spoon and use it to follow a circular trajectory inside a pot.
Reward: $R(s,a) = 0.15 \times \texttt{stable} + 0.25 \times \texttt{hand_tool} + 0.25 \times \texttt{in_pot} + 0.35 \times \texttt{trajectory}$. The reward encourages grasping the spoon, keeping it inside the pot, and tracking the desired stirring trajectory while maintaining balance.
Termination. 1000 steps, or when $z_{\text{pelvis}} < 0.58$.

For DACER, DIME, FlowRL, and FLAC, we use their official implementations from `https://github.com/happy-yan/DACER-Diffusion-with-Online-RL`, `https://github.com/ALRhub/DIME`, `https://github.com/bytedance/FlowRL`, and `https://github.com/bytedance/FLAC`, respectively, and align the experimental settings, including the number of environment interaction steps, evaluation frequency, and reporting protocol, for a fair comparison. For SAC and DreamerV3, we report the baseline results released by TD-MPC2, which are obtained from the official implementations of `https://github.com/denisyarats/pytorch_sac` and `https://github.com/danijar/dreamerv3`, respectively. For BMPC and BOOM, we use their official implementations from `https://github.com/wertyuilife2/bmpc` and `https://github.com/molumitu/BOOM_MBRL`, respectively, and also align the environment interaction budget and evaluation protocol with our setting.

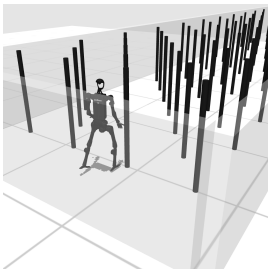


Figure 15: Pole

Objective. Move forward through a dense field of thin poles without collision.
Reward: $R(s,a) = (0.5 \times \texttt{stable} + 0.5 \times \texttt{move}) \times \texttt{collision_discount}$. Here, `move` encourages forward locomotion, while `collision_discount` penalizes contacts with poles.
Termination. 1000 steps, or when $z_{\text{pelvis}} < 0.5$.

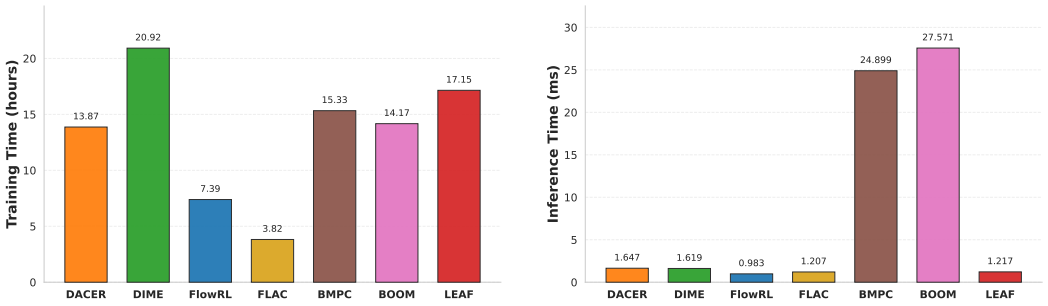
Table 4: Hyperparameter settings.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Training			
Learning rate	3×10^{-4}	Target network update rate	0.5
Encoder learning rate	1×10^{-4}	Discount factor (γ)	0.99
Sample batch size	1	Gradient clipping norm	20
Replay batch size	256	Optimizer	RAD [30]
Buffer size	1×10^6	Loss norm	Moving $(5\%, 95\%)$
Environment steps	1×10^6	Sampling	Uniform
Look-ahead Target			
Reward loss coefficient (c_r)	0.1	Dynamics loss coefficient (c_f)	20
Value loss coefficient (c_v)	0.1	Value functions ensemble	5
Number of value bins	101	Value dropout rate	1%
Warmup steps	1000	Look-ahead horizon (H)	3
MPPI iterations (J)	6 (8 if $\dim(\mathcal{A}) > 20$)	Candidate sequences (N)	512
Prior temperature (κ)	1.0	Minimum prior std	0.05
Two-phase Variational Flow			
Switching time (t_c)	0.8	Initial distribution	$\mathcal{U}(-1, 1)$
Phase I steps (K_1)	2	Phase II steps (K_2)	1
Flow loss coefficient (α_1)	1.0	Gradient-flow loss coefficient (α_2)	1.0
Model loss coefficient (α_3)	1.0	Stop-gradient operator	Enabled
Architecture (around 5M parameters in total)			
Encoder layers	2	Latent space dimension	512
Encoder dimension	256	Task embedding dimension	96
MLP hidden layer dimension	512	MLP activation	Mish
MLP normalization	LayerNorm		

E Efficiency Comparison

Fig. 16 compares the training and inference efficiency of LEAF against strong generative RL and model-based RL baselines. LEAF requires 17.15 hours to train for 1M environment steps, which is higher than lightweight flow-based baselines such as FlowRL and FLAC. This additional training cost mainly comes from learning the transition dynamics and reward predictor, as well as using the learned dynamics for MPPI-based prior construction during policy optimization. Nevertheless, LEAF remains comparable to model-based baselines such as BMPC and BOOM, and is still faster than DIME.

At inference time, LEAF no longer performs online MPPI planning. Instead, action generation only requires forward integration through the learned two-phase flow. As a result, LEAF achieves an average per-action inference time of 1.217 ms, comparable to model-free generative RL baselines such as DACER, DIME, FlowRL, and FLAC. By contrast, planning-based methods BMPC and BOOM require 24.899 ms and 27.571 ms, respectively. These results show that LEAF achieves the best overall performance while maintaining low inference latency.



(a) Training Time

(b) Inference Time

Figure 16: **Efficiency Comparison.** Comparison of wall-clock training time for 1M environment steps and average per-action inference time across major baselines.

F Limitations and Future Works

Despite the strong empirical performance and theoretical foundations of LEAF, there are a few limitations that provide avenues for future research. First, the introduction of a predictive world model and the MPPI-based prior construction inevitably increases the training-time computational overhead and memory footprint compared to purely model-free generative baselines. Although the inference latency is effectively amortized and remains highly competitive, optimizing the training efficiency of the look-ahead mechanism is an important next step.

Second, the current two-phase generative process relies on several predefined hyperparameters, such as the phase-switching timestamp and the integration step size. While our ablation studies demonstrate robustness across a reasonable range of these values, achieving optimal performance on vastly different tasks may still require manual tuning. Developing an adaptive scheduling mechanism that dynamically adjusts the transition boundary based on the learning progress could further enhance the algorithmic flexibility.

Finally, our current empirical evaluations primarily focus on complex continuous control tasks using state-based observations. Extending the LEAF framework to handle high-dimensional visual inputs, such as pixel-based RL scenarios, presents an exciting direction. Integrating latent dynamics models and visual encoders into our proximal variational flow framework will be explored in future work to broaden the applicability of LEAF to more diverse real-world robotic environments.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Yes, the main claims in the abstract and introduction accurately reflect the paper's contributions and scope. Please refer to the Abstract and Section 1.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our approach in Section F, specifically focusing on its training-time computational overhead, the reliance on predefined hyperparameters for the two-phase generative process, and the current restriction to state-based observations.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The detailed assumptions and complete proofs for all theoretical results are explicitly stated and provided in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provide the implementation code in the supplementary material, along with comprehensive environmental details, baseline references, and hyperparameters in Appendix D.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: We provide the implementation code of the core algorithm in the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We summarize the experimental setup and benchmarks in Section 4.1, and provide comprehensive environmental and training details in Appendix D.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We report the mean and standard deviation of the Total Average Return (TAR) across five random seeds in all performance tables, and display confidence intervals as shaded regions in our training curves.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We specify the exact hardware resources used, including the CPU and GPU models, along with the execution time in Appendix D.2.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that our algorithmic research, evaluated entirely within standard simulated environments, strictly adheres to all guidelines.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

Justification: This paper focuses on foundational reinforcement learning methodology evaluated exclusively in simulated environments, and thus does not have a direct path to immediate societal applications or negative impacts.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: Our research focuses on continuous control policies for simulated environments, posing no high risk for misuse compared to large-scale text or image generative models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All existing assets, including benchmark environments and baseline implementations, are properly cited, with their respective source URLs explicitly detailed in Appendix D.

Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The newly introduced algorithmic framework is provided as source code within the supplementary material.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, human annotation, or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

Answer: [N/A]

Justification: Because all experiments are evaluated purely in simulated control benchmarks, this research does not involve human subjects, making IRB approval inapplicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

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

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.