LEARNING MULTI-STEP POLICY FOR PLANNING VIA WORLD-MODEL-GUIDED FLOW MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning with learned world models has demonstrated strong performance in online reinforcement learning for continuous control, but prominent TD-MPC2-style methods generate action sequences by repeatedly sampling from a one-step Gaussian policy conditioned on model-predicted states. This construction limits the policy's ability to capture diverse planner-generated actions, reduces temporal coherence across the full sequence, and makes sequence generation sensitive to model prediction errors. We propose FLOWP, which replaces the one-step Gaussian policy with a multi-step flow policy. We train the policy with world-model-guided flow matching, which leverages model-predicted returns to improve the policy while regularizing it toward the planner's action distribution. By jointly sampling entire action sequences, our method can represent diverse planner-induced distributions and temporal dependencies while reducing sensitivity to model prediction errors. We evaluate FLOWP on 7 high-dimensional DMControl tasks and 14 HumanoidBench tasks using the same hyperparameters across all tasks. FLOWP demonstrates strong sample efficiency, achieving the highest normalized task-average performance throughout most of training on both benchmarks.

1 INTRODUCTION

Online reinforcement learning (RL) enables agents to improve their policies through interaction with the environment, making it a promising framework for continuous control. When environment interaction is costly, learning effectively from limited experience becomes a central challenge in online RL. Model-based RL addresses this challenge by learning an internal representation of the environment, called a world model. World models enable agents to predict the consequences of actions without executing them in the real environment, thereby improving sample efficiency (Ha & Schmidhuber, 2018). Specifically, LOOP, TD-MPC, and subsequent methods incorporate online planning with learned world models and have demonstrated strong performance (Sikchi et al., 2022; Hansen et al., 2022; 2024; Wang et al., 2025; Zhan et al., 2025; Lin et al., 2026). At each environment step, these methods sample action sequences, evaluate them using a learned world model, and execute the first action of a high-return sequence. TD-MPC and TD-MPC2 train this policy by maximizing the learned value function (Hansen et al., 2022; 2024). Building on TD-MPC2, subsequent work showed that a mismatch between the planner's and policy's action distributions can degrade performance and introduced imitation of planner actions into policy learning (Wang et al., 2025).

During planning, TD-MPC2 and its successors generate each action sequence by repeatedly sampling actions from a learned one-step Gaussian policy conditioned on states predicted by the world model. However, this construction limits the policy's ability to capture diverse planner-generated actions, prevents direct modeling of temporal dependencies across the full sequence, and makes sequence generation sensitive to model prediction errors.

In this paper, we introduce **Flo**w Policy for **W**orld-model-based **P**lanning (FLOWP), which replaces the one-step Gaussian policy with a multi-step flow policy that jointly generates an entire action sequence. We train this policy using *world-model-guided flow matching*, which weights planner-generated action sequences according to bootstrapped returns predicted by the learned world model. The resulting objective improves the policy while regularizing it toward the planner's action distribution. During planning, the expressive flow policy jointly samples entire action sequences, allowing

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

it to represent diverse planner-induced distributions, capture temporal dependencies, and reduce sensitivity to compounding world-model errors during sequence generation. Figure 1 provides an overview of the proposed framework.

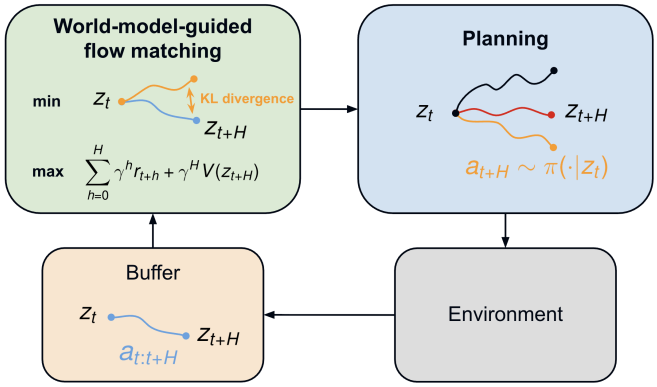

Figure 1: Overview of FLOWP. World-model-guided flow matching trains a multi-step policy using action sequences from the replay buffer and model-predicted returns. The policy proposes candidate action sequences for planning, and transitions collected through environment interaction are stored in the replay buffer for subsequent training.

We evaluate our method on 7 high-dimensional DMControl tasks and 14 HumanoidBench tasks using the same hyperparameters across all tasks. Our method demonstrates high sample efficiency and achieves the highest normalized task-average performance over most of training on both benchmarks. These results demonstrate the effectiveness of FLOWP for world-model-based planning.

Our main contributions are as follows:

- We introduce a flow policy into model-based online RL.
- We introduce world-model-guided flow matching, which improves the policy using bootstrapped returns predicted by the world model while regularizing it toward the planner's action distribution.
- We demonstrate improved sample efficiency and strong performance across 21 challenging continuous-control tasks with a single set of hyperparameters.

2 Preliminaries and Background

Problem setting. We consider online reinforcement learning (RL) for continuous control. At each step t, an agent observes a state s_t, selects a continuous action a_t, and receives a reward r_t from the environment. The objective is to maximize the expected discounted return $\mathbb{E}[R_t]$, where $R_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ and $0 \leq \gamma < 1$ is the discount factor. The state-value function $V(s_t) := \mathbb{E}[R_t \mid s_t]$ and action-value function $Q(s_t, a_t) := \mathbb{E}[R_t \mid s_t, a_t]$ denote the expected returns conditioned on the current state and state–action pair, respectively.

Planning in World Models. TD-MPC and its successors select actions through policy-guided planning with a learned world model and value function (Hansen et al., 2022; 2024; Wang et al., 2025; Zhan et al., 2025; Lin et al., 2026). The world models used in these methods consist of an encoder h_θ, which maps the raw state s_t to a latent state z_t, a dynamics model f_θ, which predicts the next state z_{t+1} from z_t and an action a_t, and a reward model g_θ, which predicts the immediate reward r_t:

$$z_t = h_\theta(s_t), \tag{1}$$

$$z_{t+1} = f_\theta(z_t, a_t), \tag{2}$$

$$r_t = g_\theta(z_t, a_t), \tag{3}$$

where θ is a learnable parameter. For an H-step action sequence $a_{t:t+H} = (a_t, \ldots, a_{t+H-1})$, the bootstrapped return estimated with the learned world model and value function V_θ is defined as

$$J_\theta(z_t, a_{t:t+H}) \coloneqq \sum_{h=0}^{H-1} \gamma^h r_{t+h} + \gamma^H V_\theta(z_{t+H}). \tag{4}$$

Ideally, the action is selected by solving the following planning objective:

$$a_t^* = \arg\max_{a_t} \max_{a_{t+1:t+H}} J_\theta(z_t, a_{t:t+H}), \tag{5}$$

Because directly solving this optimization problem is generally intractable, TD-MPC and its successors approximately optimize it through iterative sampling and refinement based on CEM (Rubinstein, 1999) and MPPI (Williams et al., 2017). Because only a finite set of action sequences can be sampled, this set must contain high-return candidates for planning to succeed, and finding such candidates becomes increasingly difficult as the action dimension grows. To alleviate this problem, these methods learn a policy π_ϕ that samples an action conditioned on the latent state z_t:

$$a_t \sim \pi_\phi(\cdot \mid z_t), \tag{6}$$

where ϕ denotes the learnable policy parameters. The policy serves as the action distribution used to sample action sequences during planning.

Flow policies. Flow policies apply flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023) to policy learning by generating continuous action sequences through a learned transport process. This is typically implemented by a time-dependent velocity field v_ϕ, which maps a flow time $\tau \in [0, 1]$, an intermediate action sequence $a^\tau_{t:t+H}$ at τ, and a state z_t to the instantaneous velocity of the transport process:

$$\frac{da^\tau_{t:t+H}}{d\tau} = v_\phi(a^\tau_{t:t+H}, \tau, z_t), \tag{7}$$

where ϕ is a learnable parameter. The flow starts from a standard Gaussian action sequence $a^0_{t:t+H} \sim \mathcal{N}(0, I)$ at flow time 0 and transports it to a generated action sequence $a^1_{t:t+H}$ at flow time 1, conditioned on z_t. We denote this sampling process by

$$a_{t:t+H} \sim \pi_\phi(\cdot \mid z_t). \tag{8}$$

In the common linear-path formulation with uniform time sampling (Lipman et al., 2024), flow matching trains the vector field by minimizing the following objective, which matches its output to the target velocity along an interpolation between a standard Gaussian action sequence $a^0_{t:t+H}$ and a data action sequence $a_{t:t+H}$:

$$\mathbb{E}_{\substack{z_t \sim \rho,\, a^0_{t:t+H} \sim \mathcal{N}(0,I), \\ a_{t:t+H} \sim \mu(\cdot|z_t),\, \tau \sim \mathcal{U}(0,1)}} \left[\left\| v_\phi(a^\tau_{t:t+H}, \tau, z_t) - (a_{t:t+H} - a^0_{t:t+H}) \right\|_2^2 \right], \tag{9}$$

where ρ denotes the distribution of states, $\mathcal{N}(0, I)$ denotes the standard Gaussian distribution, $\mu(\cdot \mid z_t)$ denotes the conditional distribution of data action sequences, $\mathcal{U}(0, 1)$ denotes the uniform distribution, and $a^\tau_{t:t+H} = (1 - \tau)a^0_{t:t+H} + \tau a_{t:t+H}$. Flow models can capture multimodal distributions (Lipman et al., 2023; Park et al., 2025; Black et al., 2025) and generate samples through a deterministic ODE, whereas diffusion models are commonly formulated through stochastic denoising processes (Sohl-Dickstein et al., 2015; Ho et al., 2020) or score-based stochastic differential equations (Song et al., 2021).

3 Method

3.1 Method Overview

TD-MPC (Hansen et al., 2022) and its successors learn a one-step policy and use it to guide planning; TD-MPC2 and the subsequent baselines considered here parameterize this prior as a Gaussian policy. TD-MPC and TD-MPC2 (Hansen et al., 2022; 2024) train the policy to maximize the learned value function. Subsequent methods (Wang et al., 2025; Zhan et al., 2025; Lin et al., 2026) incorporate imitation of planner-generated actions into policy learning to address performance degradation

Algorithm 1 FLOWP training

Require: Initialize encoder h_θ, dynamics model f_θ, reward model g_θ, value ensemble $\{V_{i,\theta}\}_i$, flow policy π_ϕ and replay buffer $\mathcal{D}$.

for seed steps times **do**
  Sample a random action a_t
  Interact with environment using the action: $(s_{t+1}, r_t) \leftarrow \text{env.step}(a_t)$
  Store transition (s_t, a_t, r_t, s_{t+1}) into replay buffer $\mathcal{D}$
  Update h_θ, f_θ, g_θ and $\{V_{i,\theta}\}_i$ by minimizing Eq. 16.
for each iteration **do**
  // Data Collection
  $z_t \leftarrow h_\theta(s_t)$
  $a_t \leftarrow \text{Plan}(z_t, \pi_\phi, J_\theta)$
  $(s_{t+1}, r_t) \leftarrow \text{env.step}(a_t)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r_t, s_{t+1})$
  // Update networks
  Update h_θ, f_θ, g_θ and $\{V_{i,\theta}\}_i$ by minimizing Eq. 16.
  Update π_ϕ by minimizing Eq. 13.
  Update $\{V^-_{i,\theta}\}_i$ via EMA.

caused by a mismatch between the planner's and policy's action distributions. In these methods, multi-step action sequences are generated during planning by repeatedly using the learned world model and policy. Starting from the current state, the policy samples a one-step action, the dynamics model predicts the next state, and this process is repeated to obtain the full action sequence.

However, these approaches face three limitations. First, a Gaussian policy lacks the expressiveness needed to capture the diverse actions generated by the planner (Huang et al., 2023; Zhan et al., 2025). Second, sampling the action at each time step separately prevents modeling temporal dependencies across the full action sequence, which can reduce temporal coherence (Chi et al., 2025; Zhao et al., 2023; Li et al., 2025). Third, later actions are conditioned on states predicted by the world model, so prediction errors can propagate through the sequence and reduce the quality of the sampled sequences (Cheng et al., 2026; Park et al., 2026b).

To address these limitations, we replace the one-step Gaussian policy with a multi-step flow policy, and train this policy using our proposed *world-model-guided flow matching*, which improves the policy using the world model and value function while regularizing it toward imitating action sequences discovered through planning (Section 3.2). This policy provides greater expressiveness for capturing planning-induced action distributions (Lipman et al., 2023; Park et al., 2025), and guides planning by jointly sampling temporally coherent action sequences from the current state (Zhao et al., 2023; Chi et al., 2025; Li et al., 2025) (Section 3.3).

We follow TD-MPC2 (Hansen et al., 2024) for world-model learning and BMPC (Wang et al., 2025) for value-function learning (Section 3.4).

The training pseudocode is presented in Algorithm 1.

3.2 World-Model-Guided Flow Matching

We train the policy to maximize the bootstrapped return $J_\theta(z_t, a_{t:t+H})$. At the same time, we regularize the policy toward the planner's action distribution. Let π_ϕ denote the H-step flow policy, where ϕ is a learnable parameter, and μ the H-step action distribution generated by the planner. Their mismatch can be measured by the reverse KL divergence from the policy to the planner distribution:

$$D(\pi_\phi(\cdot \mid z_t) \,\|\, \mu(\cdot \mid z_t)) \coloneqq \int \pi_\phi(a_{t:t+H} \mid z_t) \log \frac{\pi_\phi(a_{t:t+H} \mid z_t)}{\mu(a_{t:t+H} \mid z_t)} \, da_{t:t+H}. \tag{10}$$

Therefore, similar to BRAC (Wu et al., 2019), we define the policy loss as follows:

$$-\mathbb{E}_{z_t \sim \rho} \left[\mathbb{E}_{a_{t:t+H} \sim \pi_\phi(\cdot \mid z_t)} \left[J_\theta(z_t, a_{t:t+H}) \right] - \alpha \, D(\pi_\phi(\cdot \mid z_t) \,\|\, \mu(\cdot \mid z_t)) \right], \tag{11}$$

where $\alpha > 0$ controls the strength of regularization toward the planner. As proven in Appendix A, optimizing the objective in Eq. 11 is equivalent to minimizing the following weighted flow matching

objective:

$$\mathbb{E}_{\substack{z_t\sim\rho,\\ a^0_{t:t+H}\sim\mathcal{N}(0,I),\\ a_{t:t+H}\sim\mu(\cdot|z_t),\\ \tau\sim\mathcal{U}(0,1)}}\left[\frac{1}{Z_\theta(z_t)}\exp\left(\frac{J_\theta(z_t,a_{t:t+H})}{\alpha}\right)\left\|v_\phi(z_t,a^\tau_{t:t+H},\tau)-(a_{t:t+H}-a^0_{t:t+H})\right\|_2^2\right], \tag{12}$$

where $a^\tau_{t:t+H}=(1-\tau)a^0_{t:t+H}+\tau a_{t:t+H}$, $Z_\theta(z_t)=\mathbb{E}_{a_{t:t+H}\sim\mu(\cdot|z_t)}\left[\exp(J_\theta(z_t,a_{t:t+H})/\alpha)\right]$ is the normalization factor. We call this approach *world-model-guided flow matching*, as the flow matching loss is weighted by a function of the bootstrapped returns predicted by the learned world model and value function. In practice, we minimize the following empirical objective:

$$\mathbb{E}_{\substack{(s_t,a_{t:t+H})\sim\mathcal{D},\\ a^0_{t:t+H}\sim\mathcal{N}(0,I),\\ \tau\sim\mathcal{U}(0,1)}}\left[\exp\left(\frac{A_\theta(z_t,a_{t:t+H})}{\alpha}\right)\left\|v_\phi(a^\tau_{t:t+H},\tau,z_t)-\left(a_{t:t+H}-a^0_{t:t+H}\right)\right\|_2^2\right], \tag{13}$$

where $z_t=h_\theta(s_t)$ and $a^\tau_{t:t+H}=(1-\tau)a^0_{t:t+H}+\tau a_{t:t+H}$. Here, $A_\theta(z_t,a_{t:t+H})=J_\theta(z_t,a_{t:t+H})-V_\theta(z_t)$ denotes the advantage. By sampling $(s_t,a_{t:t+H})$ from the replay buffer $\mathcal{D}$ and encoding the raw state s_t with h_θ, we obtain the latent state z_t together with the action sequence generated by the planner from that state. If we allow the expectation and exponential to be interchanged, the normalization factor can be approximated as $Z_\theta(z_t)=\exp(V_\theta(z_t)/\alpha)$. This converts the weight into an advantage based form similar to that used in AWAC (Nair et al., 2021).

3.3 Flow policy-guided planning

During planning, we sample several action sequences from the flow policy. To generate these action sequences, we independently sample multiple standard Gaussian action sequences $a^0_{t:t+H}\sim\mathcal{N}(0,I)$ and integrate the flow ODE in Eq. 7 from $\tau=0$ to $\tau=1$ for each sequence, conditioned on the current state z_t. All other aspects of the planning procedure are identical to those of TD-MPC2 (Hansen et al., 2024).

3.4 World Model and Value Function Learning

Given a trajectory segment sampled from the replay buffer, the encoder and latent dynamics model produce a multi-step latent rollout, while the reward model predicts the reward at each rollout step:

$$\begin{aligned}\hat{z}_t&=h_\theta(s_t),\\ \hat{z}_{t+h+1}&=f_\theta(\hat{z}_{t+h},a_{t+h}),\\ z_{t+h}&=\mathrm{sg}(h_\theta(s_{t+h}))\,,\\ \hat{r}_{t+h}&=g_\theta(\hat{z}_{t+h},a_{t+h}).\end{aligned} \tag{14}$$

Here, $\hat{z}_{t+h}$ denotes the model-predicted latent state, whereas z_{t+h} is the target representation of the corresponding observed state. The stop-gradient operator sg prevents the target representation from being updated through the latent consistency objective.

Each member of the value ensemble evaluates the predicted latent states using an N-step TD target:

$$\begin{aligned}\hat{V}_{t+h}&=V_{i,\theta}(\hat{z}_{t+h}),\\ V_{t+h}&=\sum_{k=0}^{N-1}\gamma^k r_{t+h+k}+\gamma^N\bar{V}^-_\theta(\mathrm{sg}(h_\theta(s_{t+h+N}))).\end{aligned} \tag{15}$$

Here, N is the TD horizon used to construct the bootstrapped return (Sutton, 1988). The target value ensemble $\bar{V}^-_\theta$ is updated by an exponential moving average of the online parameters (Lillicrap et al., 2016).

Using these predictions and targets, we jointly optimize the world model components h_θ, f_θ, and g_θ and each value function $V_{i,\theta}$ with

$$\mathbb{E}_{\substack{(s_{t:t+H+1},a_{t:t+H},\\ r_{t:t+H})\sim\mathcal{D}}}\left[\sum_{h=0}^{H-1}\lambda^h\left(\|\hat{z}_{t+h+1}-z_{t+h+1}\|_2^2+\mathrm{CE}(\hat{r}_{t+h},r_{t+h})+\mathrm{CE}(\hat{V}_{t+h},V_{t+h})\right)\right], \tag{16}$$

where $\lambda \in (0, 1]$ discounts losses at more distant rollout steps. The three terms train the latent dynamics, reward model, and value function, respectively. The latent consistency term is a joint-embedding objective that aligns the predicted next latent state with the encoded target representation (Grill et al., 2020). Following TD-MPC2, the reward and value targets are represented as soft categorical distributions in a log-transformed space, and CE denotes the corresponding cross-entropy loss (Bellemare et al., 2017; Hafner et al., 2025; Hansen et al., 2024).

4 EXPERIMENTS

We evaluate FLOWP against six online RL baselines on two continuous-control benchmarks. Further experimental details are provided in Appendix B.

Baselines. The first two baselines serve as reference methods, whereas the remaining four perform online planning with learned world models: (1) SAC (Haarnoja et al., 2018), a representative model-free RL algorithm based on maximum entropy RL, (2) DreamerV3 (Hafner et al., 2025), a representative model-based RL algorithm that learns a policy from trajectories generated by a world model, (3) TD-MPC2 (Hansen et al., 2024), which plans with a learned world model and value function and trains its policy to maximize the learned value function, (4) BMPC (Wang et al., 2025), which replaces TD-MPC2's policy objective with imitation of actions generated by the planner, (5) BOOM (Zhan et al., 2025), which augments TD-MPC2's value-maximizing policy objective with a value-weighted forward KL divergence term that regularizes the policy toward the improved planner's action distribution, (6) TD-M(PC)^2 (Lin et al., 2026), which augments TD-MPC2's value-maximizing policy objective with a reverse KL divergence term that regularizes the policy toward the planner's action distribution. Appendix B.1 describes how the baseline results were obtained.

Benchmarks. Our evaluation suite comprises two challenging benchmarks: (1) seven high-dimensional tasks from DMControl (Tassa et al., 2018), involving humanoid and dog agents with state and action dimensions of 67/21 and 223/38, respectively; and (2) fourteen locomotion tasks from HumanoidBench (Sferrazza et al., 2024), featuring the Unitree H1hand robot with state and action dimensions of 151/61. Task visualizations are provided in Appendix C.

We conduct experiments with FLOWP using 3 seeds for each task.

4.1 RESULTS

We evaluate learning performance over one million environment steps and report episode return as a function of environment interactions. Figures 2 and 3 present the results on DMControl and HumanoidBench, respectively. Each figure shows the normalized task-average learning curve followed by the per-task learning curves. Because return scales differ across tasks, we normalize each task's returns by the final return of TD-MPC2 before averaging across tasks. We summarize performance using two metrics: the area under the learning curve (AUC) over the full environment-step budget, which measures sample efficiency, and the final return at one million environment steps, which measures final performance. Numerical results for both metrics are reported in Appendix B.3, and details of the aggregation procedure are provided in Appendix B.2.

DMControl. Figure 2 shows that FLOWP attains the highest normalized task-average return over most of training. Tables 2 and 3 show that it achieves the highest normalized task-average AUC, while its final normalized average closely matches that of BOOM and exceeds those of the other baselines. The distinction between the two metrics is evident on `dog-run`, where FLOWP achieves the highest AUC but BOOM attains a higher final return.

HumanoidBench. Figure 3 and Tables 4 and 5 show that FLOWP achieves the highest normalized task-average AUC and final return. At the task level, it achieves the highest AUC on eight of the fourteen tasks and the highest final return on seven tasks. Nevertheless, the strongest baselines remain superior on several tasks, indicating that the benefit of FLOWP is task dependent.

Discussion. The performance margin over the baselines is larger on HumanoidBench, whose action space has 61 dimensions, than on the evaluated DMControl domains, whose action spaces have 21 or 38 dimensions. This pattern is consistent with the quality of the policy-generated candidates becoming more important as the action-sequence search space grows. Jointly modeling the sequence can increase the probability that a finite candidate set contains high-return action sequences.

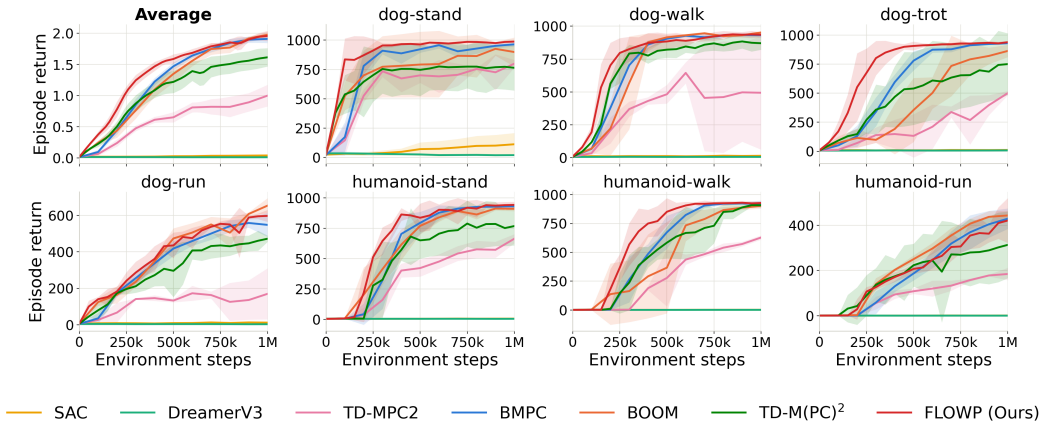

Figure 2: DMControl learning curves. The first panel shows the normalized average over all seven tasks, and the remaining panels show raw episode returns for each task. Shaded regions indicate 95% confidence intervals.

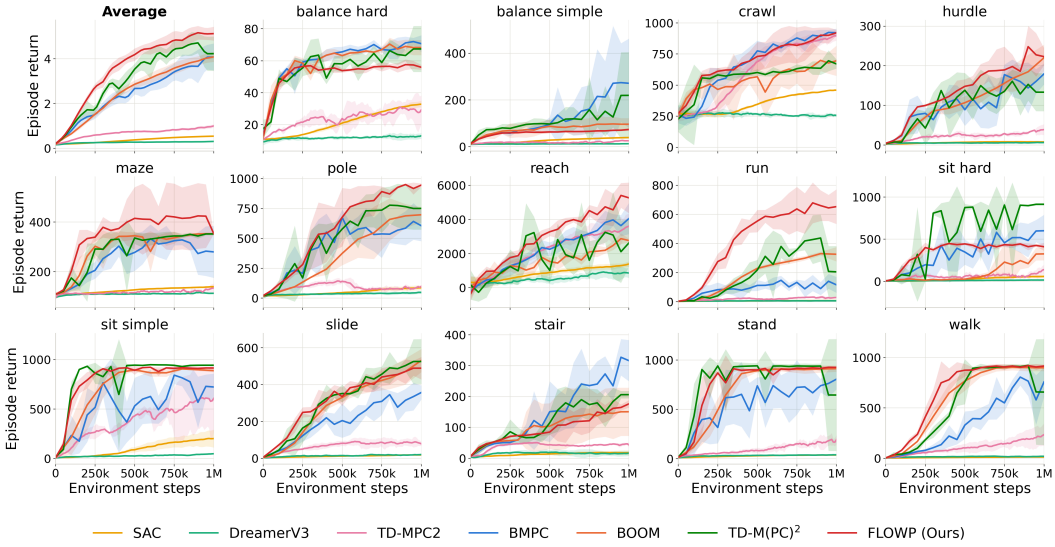

Figure 3: HumanoidBench learning curves. The first panel shows the normalized task-average curve, and the remaining panels show raw episode returns for each task. Shaded regions indicate 95% confidence intervals.

On DMControl, FLOWP achieves the highest AUC on all tasks except `humanoid-run`, indicating consistently improved sample efficiency. One possible explanation is that, during the early stages of training when the world model is still immature, the planner generates diverse and temporally complex action sequences through exploration. The expressive flow policy may imitate this planner-induced action distribution more effectively than a one-step Gaussian policy.

On HumanoidBench, the task-level results reveal a more nuanced pattern. When a one-step Gaussian policy already generates effective action sequences for planning, jointly modeling the full sequence may provide only limited improvement. In particular, the autoregressive factorization of a one-step policy may provide a favorable inductive bias when effective actions depend primarily on local state feedback because it decomposes action-sequence generation into local state-conditioned decisions along the model-predicted trajectory. The relative strength of the one-step baselines on `balance_hard`, `balance_simple`, `stair`, `slide`, and `sit_hard` is consistent with this possibility. The balance tasks require continual postural corrections, `stair` and `slide` require adaptation to uneven terrain, and `sit_hard` requires precise posture and contact adjustments when sitting. On `stand`, `sit_simple`, and `crawl`, the strongest methods achieve returns close to the maximum possible score, leaving limited room for further improvement. The gains of FLOWP on `walk`, `run`, and `hurdle` may reflect the benefit of temporally coordinated action proposals for locomotion and obstacle traversal. The gains on `maze`, `pole`, and `reach` may additionally reflect the ability of the flow policy to represent multimodal distributions over high-return action sequences.

4.2 Ablation Study

Contribution of flow-matching weights. We examine whether the flow-matching weights provide benefits beyond imitation of the planner's action-sequence distribution. Specifically, we compare the full method with an unweighted variant corresponding to $\alpha \to \infty$ in Eqs. 12 and 13. This is equivalent to directly training the flow policy to imitate the planner's action-sequence distribution. All other model components, training settings, and planning procedures are kept identical to those of the full method. We evaluate both variants on `dog-run` from DMControl and `run` from HumanoidBench using three seeds for each setting.

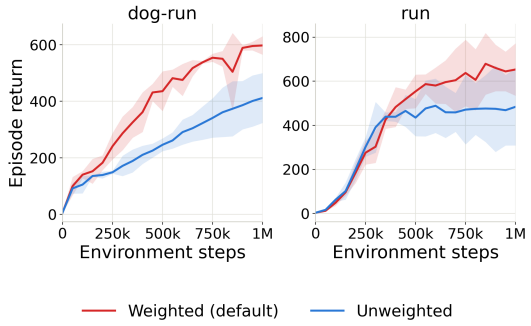

Figure 4: Ablation of the flow-matching weights on DMControl `dog-run` and HumanoidBench `run`. The weighted variant is the default method, whereas the unweighted variant assigns equal weight to all planner-generated action sequences. Curves show the mean episode return over three seeds, and shaded regions indicate 95% confidence intervals.

Figure 4 shows that the benefit of weighting differs across the two tasks. On `dog-run`, the weighted variant begins to separate from the unweighted variant early in training, and the gap generally widens thereafter. The unweighted variant continues to improve but remains substantially below the weighted variant at the end of training, indicating that imitation of the planner's action-sequence distribution alone is insufficient on this task. On HumanoidBench `run`, the two variants perform similarly during the early stage of training. The weighted variant then continues to improve, whereas the unweighted variant largely plateaus, resulting in a higher mean return for the weighted variant during the latter half of training. However, the resulting gap is smaller than that on `dog-run`, and the confidence intervals overlap substantially in the latter half of training. The smaller gain from weighting on HumanoidBench `run` than on `dog-run` may reflect the greater benefit of planner imitation in higher-dimensional action spaces (Wang et al., 2025; Zhan et al., 2025; Lin et al.,

2026). Overall, the results indicate that return-based weighting can improve performance beyond unweighted planner imitation, although the magnitude of the gain and the strength of the evidence vary across tasks.

5 Related Work

Model-based online RL. AlphaZero (Silver et al., 2017) and A0C (Moerland et al., 2018) assume access to an exact environment model and learn policies that guide Monte Carlo tree search (MCTS) (Coulom, 2006) for action selection. MuZero (Schrittwieser et al., 2020), EfficientZero (Ye et al., 2021), and Sampled MuZero (Hubert et al., 2021) learn a world model and a policy, and select actions through MCTS guided by the learned policy. PETS (Chua et al., 2018) and PlaNet (Hafner et al., 2019) learn world models and select actions through sampling-based planning based on CEM (Rubinstein, 1999), without learning policies. The Dreamer family (Hafner et al., 2020; 2021; 2025) and MBDPO (Cheng et al., 2026) learn both a world model and a policy, and select actions directly according to the learned policy without online planning. TD-MPC (Hansen et al., 2022), TD-MPC2 (Hansen et al., 2024), BMPC (Wang et al., 2025), BOOM (Zhan et al., 2025), TD-M(PC)^2 (Lin et al., 2026), EfficientTDMPC (Evers et al., 2026) and FLOWP learn a world model and a policy, and select actions using sampling-based planning that builds on CEM and MPPI (Rubinstein, 1999; Williams et al., 2017). Among these methods, only FLOWP uses a flow policy. EfficientTDMPC is concurrent work that introduces dynamics ensembles and uncertainty penalties, which are complementary to our approach and can be naturally incorporated into it.

Flow policies in RL. Existing methods that use flow policies in RL are predominantly model-free. Some methods use weighted flow matching (Zhang et al., 2025a; Nguimatsia Tiofack et al., 2026; Khan et al., 2026; Lv et al., 2025; Gao et al., 2026; Mei et al., 2026; Pfrommer et al., 2025). Other methods use unweighted flow matching (Zheng et al., 2023; Alles et al., 2025; Park et al., 2025; Li & Levine, 2026; Doo et al., 2026; Zhou et al., 2026; Koirala & Campbell, 2026; Rouxel et al., 2025; McAllister et al., 2026; Yi et al., 2026; Zhang et al., 2025b; Yang et al., 2026; Lei et al., 2026; Park et al., 2026a; Hao et al., 2026; Lyu et al., 2025; Li et al., 2025). MAC (Park et al., 2026b) is an offline model-based RL method that uses action chunks not only in its flow policy but also in its learned dynamics and reward models. Although FlowMPC (Hamel, 2026) does not optimize its flow policy through RL, it uses a learned world model and planning to refine action sequences sampled from the policy. None of these methods has been applied to online model-based RL.

6 Conclusion and future works

We introduced FLOWP, a world-model-based online planning method that uses a multi-step flow policy trained with world-model-guided flow matching. By jointly generating an entire action sequence from the current state, the policy can model diverse planning-induced action distributions and temporal dependencies, making it less sensitive to model prediction errors. Across 21 challenging continuous-control tasks from DMControl and HumanoidBench, our method achieved the highest normalized task-average AUC on both benchmarks and competitive final performance using a single set of hyperparameters. These results demonstrate the effectiveness of FLOWP for sampling-based planning.

The policy improvement achieved by our method is limited to the support of the planner's action distribution. A promising direction for future work is to complement the current return-weighted objective with gradient-based policy improvement, enabling the flow policy to improve beyond the support of the planner's action distribution.

AI Use Statement

In this work, we used generative AI tools to aid and polish writing, including translation and revisions to grammar, wording, and readability; for retrieval and discovery, including suggesting search terms and potentially relevant related work; and for research ideation or execution, including brainstorming research directions and discussing experimental and implementation choices. We did not use generative AI tools to draft sections of the paper, generate synthetic datasets, or prove mathematical claims. We reviewed all AI-assisted work, verified suggested literature against the original

sources, and independently evaluated AI-assisted research ideas and technical suggestions. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility Statement

We provide a complete description of the proposed method in Section 3, including its training procedure in Algorithm 1, and specify the benchmark tasks, baselines, and evaluation protocol in Section 4. Appendix B reports the network architectures, hyperparameters, and implementation details used across all tasks; Appendix B.1 documents how the baseline results were obtained; and Appendix B.2 describes the processing and aggregation of evaluation results. We also provide the source code associated with this work as part of the supplementary material. In addition, Appendix A provides a complete proof of the theoretical result underlying world-model-guided flow matching.

References

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations*, 2023.

Marvin Alles, Nutan Chen, Patrick van der Smagt, and Botond Cseke. FlowQ: Energy-guided flow policies for offline reinforcement learning. *arXiv preprint arXiv:2505.14139*, 2025.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 2017.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025.

Xiaoyuan Cheng, Wenxuan Yuan, Zhancun Mu, Yuanzhao Zhang, Yiming Yang, Hai Wang, Zhuo Sun, and Che Liu. Scaling world-model reinforcement learning through diffusion policy optimization. *arXiv preprint arXiv:2605.26282*, 2026.

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.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 2018.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, pp. 72–83, 2006.

JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-Flow: Stable and expressive reinforcement learning with flow-based policy. *arXiv preprint arXiv:2605.13435*, 2026.

Thomas Evers, Cristian Meo, Wendelin Bohmer, Justin Dauwels, and Yaniv Oren. EfficientTDMPC: Improved MPC objectives for sample-efficient continuous control. *arXiv preprint arXiv:2605.16692*, 2026.

Ting Gao, Stavros Orfanoudakis, Nan Lin, Elvin Isufi, Winnie Daamen, and Serge Hoogendoorn. Flow matching policy with entropy regularization. *arXiv preprint arXiv:2603.17685*, 2026.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019.

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

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Chandon Hamel. FlowMPC: Improving flow matching policies with world models. *arXiv preprint arXiv:2606.16286*, 2026.

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

Nicklas A. Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 2022.

Ruijie Hao, Longfei Zhang, Yang Dai, Yang Ma, Xingxing Liang, and Guangquan Cheng. Flow-based policy with distributional reinforcement learning in trajectory optimization. *arXiv preprint arXiv:2604.00977*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Zhiao Huang, Litian Liang, Zhan Ling, Xuanlin Li, Chuang Gan, and Hao Su. Reparameterized policy learning for multimodal trajectory optimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 13957–13975. PMLR, 2023.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4476–4486, 2021.

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

Prajwal Koirala and Mark Campbell. VGFM: Expressive robot policies via dense value guidance in flow matching. *arXiv preprint arXiv:2609.14261*, 2026.

Boshu Lei, Kostas Daniilidis, and Antonio Loquercio. Reinforcement learning for flow-matching policies with density transport. *arXiv preprint arXiv:2606.08602*, 2026.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

Qiyang Li, Zhiyuan Paul Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, volume 38, pp. 55518–55553, 2025.

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. In *International Conference on Learning Representations*, 2016.

Haotian Lin, Pengcheng Wang, Jeff Schneider, and Guanya Shi. TD-M(PC)^2: Improving temporal difference MPC through policy constraint. In *Proceedings of the 8th Annual Learning for Dynamics and Control Conference*, volume 331 of *Proceedings of Machine Learning Research*, pp. 705–736. PMLR, 2026.

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

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 93967–93990, 2025.

Mingyang Lyu, Yinqian Sun, Erliang Lin, Huangrui Li, Ruolin Chen, Feifei Zhao, and Yi Zeng. Reinforcement fine-tuning of flow-matching policies for vision-language-action models. *arXiv preprint arXiv:2510.09976*, 2025.

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

Yunpeng Mei, Jiakai He, Hongjie Cao, Chenyu Wang, Xiaowen Zhu, Yihan Zhou, Jiamin Wang, Chenbo Xin, Peng Cheng, Yuxuan Yang, Yijie Wang, Xinhu Zheng, Gao Huang, Jie Chen, and Gang Wang. Potential-guided flow matching for vision-language-action policy improvement. *arXiv preprint arXiv:2606.04968*, 2026.

Thomas M. Moerland, Joost Broekens, Madalina M. Drugan, and Marco A. Wiering. A0C: Alpha zero in continuous action space. *arXiv preprint arXiv:1805.09613*, 2018.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. In *International Conference on Learning Representations*, 2021.

Franki Nguimatsia Tiofack, Théotime Le Hellard, Fabian Schramm, Nicolas Perrin-Gilbert, and Justin Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. In *International Conference on Learning Representations*, 2026.

Bumgeun Park, Hyukjun Yang, and Donghwan Lee. Refinement-based flow policy optimization. *arXiv preprint arXiv:2609.15123*, 2026a.

Kwanyoung Park, Seohong Park, Youngwoon Lee, and Sergey Levine. Scalable offline model-based RL with action chunks. In *International Conference on Learning Representations*, 2026b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 2025.

Samuel Pfrommer, Yixiao Huang, and Somayeh Sojoudi. Reinforcement learning for flow-matching policies. *arXiv preprint arXiv:2507.15073*, 2025.

Quentin Rouxel, Clemente Donoso, Fei Chen, Serena Ivaldi, and Jean-Baptiste Mouret. Extremum flow matching for offline goal conditioned reinforcement learning. In *2025 IEEE-RAS 24th International Conference on Humanoid Robots*, 2025. doi: 10.1109/Humanoids65713.2025.11203207.

Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588: 604–609, 2020.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.061.

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1622–1633. PMLR, 2022.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

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

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988. doi: 10.1007/BF00115009.

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.

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

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40 (2):344–357, 2017.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Shunpeng Yang, Ben Liu, and Hua Chen. PolicyFlow: Policy optimization with continuous normalizing flow in reinforcement learning. In *International Conference on Learning Representations*, 2026.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 25476–25488, 2021.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E. Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, Karen Liu, and Angjoo Kanazawa. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026.

Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. In *International Conference on Learning Representations*, 2025a.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky T. Q. Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

Zhiyuan Zhou, Andy Peng, Charles Xu, Qiyang Li, Jost Tobias Springenberg, Kevin Frans, and Sergey Levine. Test-time gradient guidance of flow policies in reinforcement learning. *arXiv preprint arXiv:2606.11087*, 2026.

A THEORETICAL PROOF

We show that the policy minimizing Eq. 11 is identical to the flow policy induced by a vector field minimizing Eq. 12.

Fix an environment time t and a state z_t. Define the improved planner distribution

$$\pi_\theta^*(a_{t:t+H} \mid z_t) \coloneqq \frac{1}{Z_\theta(z_t)} \exp\left(\frac{J_\theta(z_t, a_{t:t+H})}{\alpha}\right) \mu(a_{t:t+H} \mid z_t). \tag{17}$$

For any policy π_ϕ, the term inside the expectation over states in Eq. 11 satisfies

$$\begin{aligned} &- \mathbb{E}_{a_{t:t+H}\sim\pi_\phi(\cdot|z_t)}\left[J_\theta(z_t, a_{t:t+H})\right] + \alpha\, D(\pi_\phi(\cdot \mid z_t) \,\|\, \mu(\cdot \mid z_t)) \\ =&\alpha\, D(\pi_\phi(\cdot \mid z_t) \,\|\, \pi_\theta^*(\cdot \mid z_t)) - \alpha \log Z_\theta(z_t). \end{aligned} \tag{18}$$

Since the KL divergence is nonnegative, the objective above is minimized by $\pi_\theta^*(\cdot \mid z_t)$. Consequently, the objective in Eq. 11 is minimized by π_θ^* for every state.

It remains to show that the policy induced by a vector field minimizing Eq. 12 is π_θ^*. Eq. 12 can be rewritten as

$$\mathbb{E}_{\substack{z_t\sim\rho,\\ a^0_{t:t+H}\sim\mathcal{N}(0,I),\\ a_{t:t+H}\sim\pi_\theta^*(\cdot|z_t),\\ \tau\sim\mathcal{U}(0,1)}} \left[\left\| v_\phi(z_t, a^\tau_{t:t+H}, \tau) - (a_{t:t+H} - a^0_{t:t+H})\right\|_2^2\right]. \tag{19}$$

This is the standard conditional flow matching objective. By the standard flow matching result (Lipman et al., 2023), assuming sufficient expressive capacity, the policy induced by a vector field minimizing this objective is identical to π_θ^*. Since π_θ^* minimizes Eq. 11, this completes the proof.

B EXPERIMENTAL DETAILS

Hardware. All experiments were run on a single NVIDIA A100.

Architectural details. All components of FLOWP are implemented as MLPs. We summarize the architecture using PyTorch-like notation:

```
Total parameters: 5,219,349
Encoder: ModuleDict(
  (state): Sequential(
    (0): NormedLinear(in_features=S, out_features=256, bias=True, act=Mish)
    (1): NormedLinear(in_features=256, out_features=512, bias=True, act=SimNorm)
  )
)
Dynamics: Sequential(
  (0): NormedLinear(in_features=512+A, out_features=512, bias=True, act=Mish)
  (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
  (2): NormedLinear(in_features=512, out_features=512, bias=True, act=SimNorm)
)
Reward: Sequential(
  (0): NormedLinear(in_features=512+A, out_features=512, bias=True, act=Mish)
  (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
  (2): Linear(in_features=512, out_features=101, bias=True)
)
```

```
Policy prior: Sequential(
  (0): NormedLinear(in_features=512+A*H+1, out_features=512, bias=True, act=Mish)
  (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
  (2): Linear(in_features=512, out_features=183, bias=True)
)
V-functions: Vectorized 5x Sequential(
  (0): NormedLinear(in_features=512, out_features=512, bias=True, dropout=0.01, act=Mish)
  (1): NormedLinear(in_features=512, out_features=512, bias=True, act=Mish)
  (2): Linear(in_features=512, out_features=101, bias=True)
)
```

where `S` is the raw-state dimensionality, `A` is the action space, and `H` is the horizon. The exact parameter counts listed above are for `S=151`, `A=61` and `H=3`.

Hyperparameters. We use a single hyperparameter configuration across all tasks. For hyperparameters shared with TD-MPC2, we adopt its default values (Hansen et al., 2024). Table 1 lists all hyperparameters used in our experiments.

Table 1: Hyperparameters used across all tasks. Settings marked as heuristic are determined automatically during training, and T denotes the episode length.

Hyperparameter	Value	Hyperparameter	Value
Planning		**Policy prior**	
Horizon (H)	3	Flow steps	10
Iterations	6 (+2 if $\|\mathcal{A}\| \geq 20$)	**Replay buffer**	
Population size	512	Capacity	1,000,000
Policy prior samples	24	Sampling	Uniform
Number of elites	64	**Optimization**	
Minimum std.	0.05	Update-to-data ratio	1
Maximum std.	2	Batch size	256
Temperature	0.5	Consistency loss coef.	20
Architecture (5M)		Reward loss coef.	0.1
Encoder dim	256	Value loss coef.	0.1
MLP dim	512	V-fn. momentum coef.	0.99
Latent state dim	512	Optimizer	Adam
Activation	LayerNorm + Mish	Learning rate	3×10^{-4}
V-function dropout rate	1%	Encoder learning rate	1×10^{-4}
Number of V-functions	5	Gradient clip norm	20
Number of reward/value bins	101	Discount factor	$\mathrm{clip}\big(1 - \frac{5}{T}, [0.95, 0.995]\big)$
SimNorm dim	8	Seed steps	$\max(5T, 1000)$
SimNorm temperature	1	Advantage coef ($1/\alpha$)	Heuristic
		Weight clip	20

Following TD-M(PC)^2, we make the regularization coefficient α proportional to the action-space dimension d_{action} (Lin et al., 2026). To improve training stability, we also adapt α to the scale of the sampled advantages using an exponential moving average (EMA) of their percentile range, following TD-MPC2 (Hansen et al., 2024). Combining these adjustments, we set α as

$$\alpha = \frac{d_{\text{action}}}{61} \times \mathrm{EMA}(\max\{\mathrm{Per}(A_\theta, 95) - \mathrm{Per}(A_\theta, 5), 1\}, 0.01).$$

B.1 Baseline details

In this section, we explain how we obtained the results used for the baselines.

SAC. For DMControl, we use the SAC results from the official TD-MPC2 repository: `https://github.com/nicklashansen/tdmpc2`. For HumanoidBench, we use the SAC results from the official HumanoidBench repository:

`https://github.com/carlosferrazza/humanoid-bench`.

DreamerV3. For DMControl, we use the DreamerV3 results from the official TD-MPC2 repository. For HumanoidBench, we use the DreamerV3 results from the official HumanoidBench repository.

TD-MPC2. For DMControl, we use the TD-MPC2 results from the official TD-MPC2 repository. For HumanoidBench, we use the TD-MPC2 results from the official HumanoidBench repository.

BMPC. We use the BMPC results from the official BMPC repository: `https://github.com/wertyuilife2/bmpc`.

BOOM. We use the authors' reported results and conduct experiments with three seeds for each additional HumanoidBench task using the official BOOM repository: `https://github.com/molumitu/BOOM_MBRL`.

TD-M(PC)^2. Because machine-readable numerical results are not released by the authors, we conduct experiments with three seeds for each task using an implementation based on the official TD-M(PC)^2 repository (`https://github.com/DarthUtopian/tdmpc_square_public`).

B.2 Results aggregation

Interpolation. Evaluation checkpoints may differ across seeds because evaluations can occur at episode boundaries rather than at fixed environment steps. We therefore linearly interpolate each seed's raw evaluation curve onto a common grid $t \in \{0, \Delta, 2\Delta, \dots, B\}$, where $B = 10^6$ is the environment-step budget shown in the figures and Δ is the grid spacing. To accommodate small offsets between the final checkpoint and B, a curve is held at its last logged value for grid points shortly after its final evaluation.

Per-task learning curves. For task i and grid step t, let $R_{i,j,t}$ denote the interpolated evaluation return of seed $j \in \{1, \dots, N_i\}$. We compute the sample mean, sample standard deviation, and standard error as

$$\begin{aligned} \hat{\mu}_{i,t} &= \frac{1}{N_i} \sum_{j=1}^{N_i} R_{i,j,t}, \\ \hat{\sigma}_{i,t} &= \sqrt{\frac{1}{N_i - 1} \sum_{j=1}^{N_i} (R_{i,j,t} - \hat{\mu}_{i,t})^2}, \\ \widehat{\mathrm{SEM}}_{i,t} &= \frac{\hat{\sigma}_{i,t}}{\sqrt{N_i}}. \end{aligned} \tag{20}$$

Each per-task curve reports $\hat{\mu}_{i,t}$ with a 95% confidence band $\hat{\mu}_{i,t} \pm 1.96\, \widehat{\mathrm{SEM}}_{i,t}$ based on a normal approximation. We denote its half-width by $b_{i,t} = 1.96\, \widehat{\mathrm{SEM}}_{i,t}$.

Normalized task-average curves. Because return scales differ substantially across tasks, we normalize each task by TD-MPC2's final return $c_i = \hat{\mu}_{i,\text{TD-MPC2},B}$. For an aggregate over n tasks, the normalized task-average curve is

$$\bar{\mu}_t = \frac{1}{n} \sum_{i=1}^{n} \frac{\hat{\mu}_{i,t}}{c_i}. \tag{21}$$

Thus, a value of 1.0 corresponds to TD-MPC2's final normalized performance rather than a task-success threshold. Assuming independence across tasks, we propagate the per-task confidence intervals as

$$\bar{b}_t = \frac{1}{n} \sqrt{\sum_{i=1}^{n} \left(\frac{b_{i,t}}{c_i} \right)^2}, \tag{22}$$

and report the aggregate confidence band as $\bar{\mu}_t \pm \bar{b}_t$. This interval reflects propagated seed-to-seed uncertainty but not variation in performance across tasks.

Area under the learning curve. We summarize sample efficiency using the area under the learning curve (AUC), divided by the environment-step budget $B = 10^6$ so that it represents the mean return over training. For seed j of task i, we compute the AUC on the common interpolation grid

Table 2: DM Control: AUC. Mean ± 95% CI over seeds; best mean per row in bold. The Average row is normalized by TD-MPC2's final return on each task (dimensionless); the other rows are raw episode return. AUC is the mean return over the 0–1M step learning curve. †: per-seed data unavailable, CI is a conservative approximation.

Task	SAC	DreamerV3	TD-MPC2	BMPC	BOOM	TD-M(PC)^2	FLOWP (Ours)
dog-stand	65.2 ± 47.6	26.7 ± 6.8	607 ± 71	$791 \pm 54^\dagger$	$748 \pm 189^\dagger$	695 ± 188	$\mathbf{899 \pm 38}$
dog-walk	12.8 ± 2.8	6.8 ± 1.9	384 ± 126	$709 \pm 41^\dagger$	$687 \pm 72^\dagger$	691 ± 46	$\mathbf{767 \pm 57}$
dog-trot	9.1 ± 1.9	6.4 ± 1.6	197 ± 35	$595 \pm 46^\dagger$	$411 \pm 129^\dagger$	460 ± 211	$\mathbf{746 \pm 57}$
dog-run	9.6 ± 2.1	3.7 ± 3.6	120 ± 30	$356 \pm 31^\dagger$	$385 \pm 38^\dagger$	299 ± 47	$\mathbf{388 \pm 26}$
humanoid-stand	5.4 ± 0.3	4.8 ± 0.3	353 ± 40	$598 \pm 68^\dagger$	$597 \pm 66^\dagger$	504 ± 115	$\mathbf{668 \pm 52}$
humanoid-walk	1.5 ± 0.3	0.9 ± 0.3	282 ± 24	$546 \pm 54^\dagger$	$469 \pm 157^\dagger$	486 ± 97	$\mathbf{645 \pm 52}$
humanoid-run	1.0 ± 0.1	0.7 ± 0.0	94.3 ± 4.2	$193 \pm 35^\dagger$	$\mathbf{232 \pm 17}^\dagger$	179 ± 102	201 ± 27
Average	0.03 ± 0.01	0.01 ± 0.00	0.59 ± 0.05	$1.22 \pm 0.05^\dagger$	$1.19 \pm 0.07^\dagger$	1.06 ± 0.12	$\mathbf{1.37 \pm 0.04}$

Table 3: DM Control: return at 1M environment steps. Mean ± 95% CI over seeds; best mean per row in bold. The Average row is normalized by TD-MPC2's final return on each task (dimensionless); the other rows are raw episode return.

Task	SAC	DreamerV3	TD-MPC2	BMPC	BOOM	TD-M(PC)^2	FLOWP (Ours)
dog-stand	113 ± 96	20.5 ± 2.9	799 ± 32	964 ± 19	899 ± 124	764 ± 193	$\mathbf{987 \pm 4}$
dog-walk	15.3 ± 3.6	6.7 ± 0.7	494 ± 432	932 ± 13	$\mathbf{952 \pm 5}$	870 ± 49	935 ± 20
dog-trot	11.1 ± 3.7	7.4 ± 1.8	500 ± 19	929 ± 6	863 ± 53	750 ± 265	$\mathbf{938 \pm 12}$
dog-run	11.9 ± 6.0	3.0 ± 3.0	170 ± 140	548 ± 66	$\mathbf{654 \pm 41}$	472 ± 17	597 ± 33
humanoid-stand	6.4 ± 0.1	4.7 ± 0.6	664 ± 37	931 ± 15	910 ± 19	768 ± 159	$\mathbf{942 \pm 26}$
humanoid-walk	2.0 ± 0.6	1.2 ± 0.5	628 ± 18	917 ± 31	901 ± 22	908 ± 16	$\mathbf{927 \pm 15}$
humanoid-run	1.2 ± 0.2	0.6 ± 0.2	185 ± 27	429 ± 45	$\mathbf{443 \pm 21}$	313 ± 147	421 ± 99
Average	0.04 ± 0.02	0.01 ± 0.00	1.00 ± 0.17	1.91 ± 0.07	$\mathbf{1.98 \pm 0.05}$	1.61 ± 0.15	1.96 ± 0.08

using the trapezoidal rule

$$\mathrm{AUC}_{i,j} = \frac{1}{B} \sum_{k=0}^{K-1} \frac{R_{i,j,t_k} + R_{i,j,t_{k+1}}}{2} \left(t_{k+1} - t_k\right), \tag{23}$$

where $t_k = k\Delta$ and $K = B/\Delta$. We compute the mean and 95% confidence interval of the per-seed AUC values in the same manner as for the per-task learning curves. For the normalized task-average AUC reported in the Average rows of Tables 2 and 4, we divide each task's AUC by c_i and then average across tasks.

Baselines reported with aggregate statistics. For the published BMPC and BOOM results, the authors provide aggregate statistics rather than seed-level evaluation logs. For BMPC, we use the released upper and lower confidence bounds directly and set $b_{i,t} = (\mathrm{ucb}_{i,t} - \mathrm{lcb}_{i,t})/2$, assuming that they represent 95% confidence intervals, although the exact statistic is not documented in the release. For BOOM's published results, we use the released mean and standard deviation and assume three seeds, yielding $b_{i,t} = 1.96\,\hat{\sigma}_{i,t}/\sqrt{3}$.

B.3 Numerical results

Table 4: HumanoidBench: AUC. Mean ± 95% CI over seeds; best mean per row in bold. The Average row is normalized by TD-MPC2's final return on each task (dimensionless); the other rows are raw episode return. AUC is the mean return over the 0–1M step learning curve. †: per-seed data unavailable, CI is a conservative approximation.

Task	SAC	DreamerV3	TD-MPC2	BMPC	BOOM	TD-M(PC)^2	FLOWP (Ours)
balance hard	21.2 ± 0.4	11.8 ± 1.0	23.6 ± 3.4	$\mathbf{61.0 \pm 3.5}^{\dagger}$	60.0 ± 0.9	56.0 ± 3.0	52.7 ± 1.4
balance simple	25.6 ± 1.5	11.5 ± 0.8	18.3 ± 2.9	$\mathbf{122 \pm 71}^{\dagger}$	76.2 ± 25.8	108 ± 40	59.7 ± 2.1
crawl	350 ± 9	262 ± 17	607 ± 25	$672 \pm 66^{\dagger}$	545 ± 112	568 ± 40	$\mathbf{699 \pm 76}$
hurdle	6.3 ± 0.5	5.2 ± 0.7	22.5 ± 1.0	$102 \pm 35^{\dagger}$	$108 \pm 17^{\dagger}$	102 ± 25	$\mathbf{140 \pm 16}$
maze	125 ± 3	110 ± 1	117 ± 1	$257 \pm 48^{\dagger}$	295 ± 11	289 ± 26	$\mathbf{360 \pm 73}$
pole	49.5 ± 7.1	36.0 ± 9.6	90.6 ± 6.7	$436 \pm 123^{\dagger}$	$368 \pm 83^{\dagger}$	502 ± 33	$\mathbf{591 \pm 87}$
reach	849 ± 187	531 ± 118	2123 ± 194	$2205 \pm 452^{\dagger}$	1537 ± 105	1823 ± 424	$\mathbf{3001 \pm 242}$
run	5.7 ± 0.6	5.6 ± 0.1	19.4 ± 1.6	$90.9 \pm 36.4^{\dagger}$	$187 \pm 14^{\dagger}$	201 ± 28	$\mathbf{441 \pm 59}$
sit hard	26.6 ± 4.7	10.1 ± 1.9	59.0 ± 11.0	$337 \pm 146^{\dagger}$	94.9 ± 45.0	$\mathbf{585 \pm 56}$	362 ± 1
sit simple	95.0 ± 23.1	27.1 ± 2.4	369 ± 94	$533 \pm 193^{\dagger}$	$725 \pm 72^{\dagger}$	$\mathbf{836 \pm 29}$	815 ± 15
slide	11.6 ± 2.0	14.2 ± 0.1	66.4 ± 14.0	$202 \pm 46^{\dagger}$	$288 \pm 34^{\dagger}$	313 ± 41	$\mathbf{316 \pm 27}$
stair	13.1 ± 3.2	15.5 ± 6.8	39.8 ± 1.9	$\mathbf{147 \pm 49}^{\dagger}$	101 ± 54	118 ± 17	97.1 ± 16.9
stand	28.9 ± 3.1	32.8 ± 3.2	95.9 ± 20.2	$562 \pm 198^{\dagger}$	$695 \pm 50^{\dagger}$	$\mathbf{805 \pm 65}$	769 ± 35
walk	8.7 ± 1.6	15.7 ± 3.5	104 ± 37	$342 \pm 81^{\dagger}$	$604 \pm 44^{\dagger}$	555 ± 28	$\mathbf{669 \pm 39}$
Average	0.37 ± 0.01	0.27 ± 0.01	0.69 ± 0.03	$2.44 \pm 0.29^{\dagger}$	$2.53 \pm 0.15^{\dagger}$	3.08 ± 0.16	$\mathbf{3.62 \pm 0.17}$

Table 5: HumanoidBench: return at 1M environment steps. Mean ± 95% CI over seeds; best mean per row in bold. The Average row is normalized by TD-MPC2's final return on each task (dimensionless); the other rows are raw episode return.

Task	SAC	DreamerV3	TD-MPC2	BMPC	BOOM	TD-M(PC)^2	FLOWP (Ours)
balance hard	32.9 ± 2.1	13.1 ± 1.8	29.3 ± 8.9	$\mathbf{70.7 \pm 5.2}$	68.3 ± 4.7	67.4 ± 14.4	55.9 ± 1.5
balance simple	39.0 ± 2.0	12.6 ± 0.7	24.9 ± 1.6	$\mathbf{271 \pm 191}$	95.9 ± 26.0	219 ± 185	73.5 ± 5.6
crawl	459 ± 6	256 ± 12	897 ± 38	$\mathbf{921 \pm 33}$	696 ± 121	670 ± 37	921 ± 26
hurdle	7.8 ± 1.6	6.2 ± 0.7	38.9 ± 13.4	179 ± 27	220 ± 33	133 ± 48	$\mathbf{223 \pm 22}$
maze	140 ± 4	113 ± 2	133 ± 19	280 ± 104	353 ± 3	352 ± 5	$\mathbf{353 \pm 1}$
pole	86.9 ± 19.4	47.8 ± 12.3	98.7 ± 12.2	606 ± 129	698 ± 64	750 ± 177	$\mathbf{946 \pm 14}$
reach	1392 ± 141	864 ± 286	3610 ± 699	4052 ± 908	2766 ± 107	2623 ± 1398	$\mathbf{5266 \pm 895}$
run	7.4 ± 0.7	6.4 ± 2.0	28.5 ± 1.9	116 ± 54	326 ± 49	205 ± 177	$\mathbf{653 \pm 119}$
sit hard	58.2 ± 16.8	15.4 ± 4.4	139 ± 53	600 ± 189	326 ± 76	$\mathbf{914 \pm 12}$	410 ± 47
sit simple	201 ± 90	47.9 ± 6.1	607 ± 339	722 ± 135	888 ± 16	$\mathbf{942 \pm 5}$	914 ± 11
slide	18.2 ± 3.9	20.6 ± 2.1	78.8 ± 10.8	357 ± 38	$\mathbf{531 \pm 63}$	526 ± 122	489 ± 61
stair	17.4 ± 7.4	16.2 ± 8.3	43.1 ± 7.8	$\mathbf{316 \pm 67}$	151 ± 78	206 ± 12	176 ± 12
stand	38.1 ± 1.1	40.9 ± 3.0	193 ± 45	806 ± 104	912 ± 32	645 ± 566	$\mathbf{925 \pm 9}$
walk	12.5 ± 1.8	19.5 ± 7.7	234 ± 109	757 ± 147	898 ± 33	657 ± 526	$\mathbf{915 \pm 13}$
Average	0.54 ± 0.03	0.31 ± 0.02	1.00 ± 0.07	4.08 ± 0.60	4.08 ± 0.22	4.23 ± 0.77	$\mathbf{5.12 \pm 0.31}$

C TASK VISUALIZATIONS

Figures 5 and 6 visualize the agents and environments used in our evaluation.

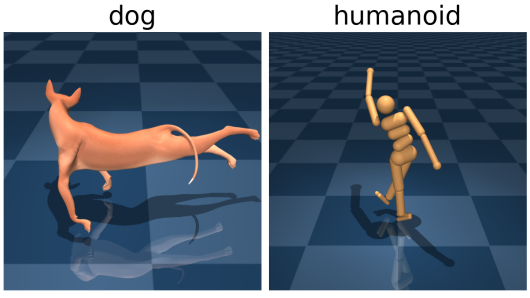

Figure 5: DMControl task visualizations. The evaluated tasks control dog and humanoid agents across 7 tasks.

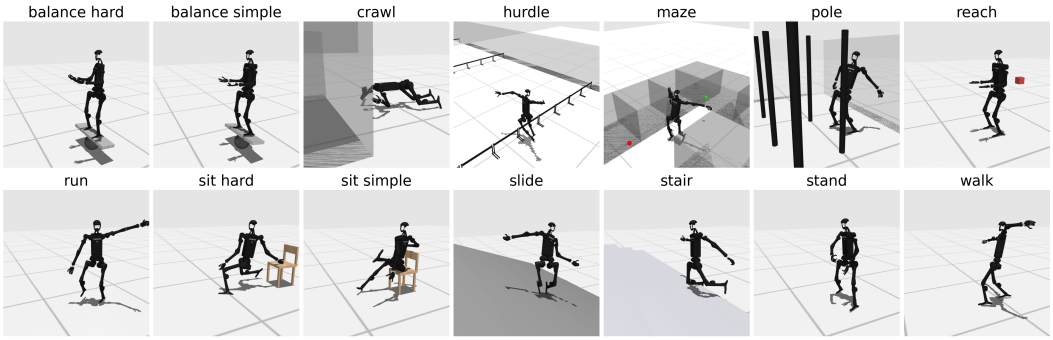

Figure 6: HumanoidBench locomotion task visualizations. The Unitree H1hand robot is controlled in 14 diverse tasks.