EFFICIENTTDMPC: IMPROVED MPC OBJECTIVES FOR SAMPLE-EFFICIENT CONTINUOUS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce EfficientTDMPC, a sample-efficient model-based reinforcement learning method for continuous control built on the TD-MPC family of algorithms. Central to this family is a planner that aims to find an action sequence that maximizes the estimated return. The return is estimated using a learned model and value networks, each of which can introduce error. EfficientTDMPC introduces three contributions that improve performance by aiming to reduce this error. First, we introduce a multi-horizon planning objective that evaluates the value at different rollout depths and averages them. Second, to our knowledge we are the first to train a value-equivalent dynamics ensemble. Our improved objective then averages over rollouts from multiple dynamics heads. Third, we add pessimistic reanalyze for tasks that can terminate early. Applying our contributions to a recent baseline (BMPC) yields EfficientTDMPC, which to our knowledge is the new state of the art in sample efficiency on HumanoidBench and the DeepMind Control Suite, reaching BMPC's final aggregated performance using 57% fewer environment steps.

1 INTRODUCTION

Model-based reinforcement learning (MBRL) methods that learn a world model and use it for planning have achieved strong sample efficiency in continuous control (Yu et al., 2020; Janner et al., 2019; Chua et al., 2018; Hansen et al., 2022; 2024), with applications spanning robotics, autonomous vehicles, and spatial understanding and reasoning (Hafner et al., 2020; 2025; Meo et al., 2025). The TD-MPC family (Hansen et al., 2022) has been very successful in this domain by introducing a latent dynamics model with an MPC planner to find action sequences that maximize a bootstrapped return estimate.

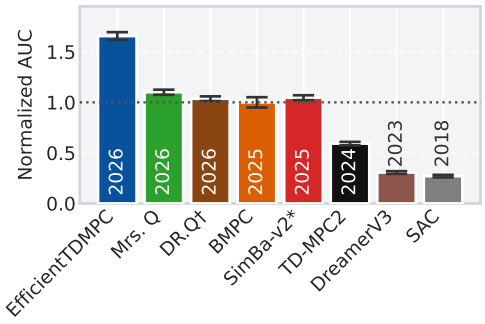


Figure 1: Normalized mean area under the learning curve (AUC) on HumanoidBench (14 tasks) and the DeepMind Control Suite (28 tasks) over the first half of their full step budget. Bars are relative to BMPC. EfficientTDMPC outperforms the state of the art by over 45%. *† Further details in Appendix A.2.

Several recent works in the family, like TD-M(PC)^2 (Lin et al., 2026), BOOM (Zhan et al., 2025) and BMPC (Wang et al., 2025), then distill this planner into the policy network, relying even more strongly on planning. The planner's purpose is to find action sequences that maximize the estimated return. This estimate is created by rolling out the learned dynamics and reward models and then bootstrapping with a value network. However, when the learned model or value network is inaccurate, the planner can exploit these inaccuracies to find action sequences that are estimated to yield a higher return than they actually do in the environment (Chua et al., 2018; Janner et al., 2019).

EfficientTDMPC introduces several contributions that improve performance by aiming to reduce the error in this return estimate. Firstly, we introduce a multi-horizon planning objective: we average the return estimates across different rollout depths, which has been successful in MCTS (Browne et al., 2012; Silver et al., 2018; Schrittwieser et al., 2020; Oren et al., 2026) but is novel in MPC to

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.

our knowledge. Secondly, to the best of our knowledge, we are the first to train a value-equivalent dynamics ensemble. We then use that ensemble during planning by averaging the return across all dynamics models which aims to prevent the exploitation of inaccurate models (Chua et al., 2018; Janner et al., 2019; Yu et al., 2020). Thirdly, we introduce pessimistic reanalyze, which penalizes action sequences for which the return estimate is uncertain when creating policy targets. We also add practical improvements such as cheaper planning during reanalyze, which we find reduces wall-clock training time without sacrificing performance. Overall, to the best of our knowledge, EfficientTDMPC is the new state of the art in sample efficiency on HumanoidBench and the DeepMind Control Suite. It also reaches the aggregate baseline (BMPC) performance in 57 % fewer environment steps without significantly increasing wall-clock training time.

- **Improved planner objectives.** We introduce three changes to the planner objective and show that they contribute significantly to performance: a multi-horizon planning objective, a value-equivalent dynamics ensemble, and pessimistic reanalyze.
- **Pipeline improvements.** Per-step replay insertion, horizon increase and a reduced reanalyze budget.
- **Performance.** To the best of our knowledge, EfficientTDMPC is the new state of the art in sample efficiency on HumanoidBench and the DeepMind Control Suite.

2 BACKGROUND

Problem setting and notation: We consider continuous-control problems modeled as discounted Markov decision processes $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. Here, $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^m$ denote the state and action spaces, where $s \in \mathcal{S}$ are states and $a \in \mathcal{A}$ are actions, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the state-transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps states to action distributions. We define the value of a policy at state s as the expected discounted return when starting from s and following π thereafter,

$$V^\pi(s) = \mathbb{E}_{\pi,P}\Big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\Big|\; s_0 = s,\ a_t \sim \pi(\cdot \mid s_t), \\ s_{t+1} \sim P(\cdot \mid s_t, a_t)\Big]. \tag{1}$$

TD-MPC family: Our method builds on the TD-MPC family of works (Hansen et al., 2022; 2024) which use a strong planner to find an action sequence that maximizes the estimated return. The estimated return of an action sequence is then obtained by rolling out a latent world model and bootstrapping with a value network. The planner is used both for acting in the environment and for distilling its behavior into a policy network. Several follow-up works like TD-M(PC)^2, BOOM, and BMPC then distill this planner into the policy, relying even more strongly on the planner. We build on BMPC (Wang et al., 2025) specifically, since it is a recent contribution that has shown strong empirical performance.

Latent world model: Table 1 lists the model components of BMPC.

Table 1: Model components of BMPC

Component	Notation	Role
Encoder	h_θ	Maps observation s_t to latent $z_t = h_\theta(s_t)$
Dynamics model	f_θ	Predicts next latent: $z_{t+1} = f_\theta(z_t, a_t)$
Reward model	R_θ	Predicts scalar reward from a latent-action pair
Policy (prior)	$\pi_\theta(\cdot \mid z)$	Stochastic policy used to guide planning
Value network	$V_{\theta,i}(z)$	Ensemble of state-value networks $i \in \{1, .., N_v\}$

Given an observed state s_t, the encoder produces the latent representation $z_t = h_\theta(s_t)$. The dynamics head is then trained to predict the next latent $\hat{z}_{t+1} = f_\theta(z_t, a_t)$ given an action a_t. The reward and value networks can be evaluated on encoded or rolled-out latents; as detailed below, they are

trained primarily on dynamics rollouts. We use $X_{\theta,i}$ to denote the i-th ensemble member, and use the notation X_θ for the average over the ensemble.

Planner objective In the TD-MPC family, a planner optimizes an action sequence of length H to maximize the expected return. The exact objective used is the state-action value of the action sequence, which is defined as the expected return of executing the action sequence $\mathbf{a}_{t:t+H-1}$ starting from s_t, and then following policy π thereafter.

$$Q_H^\pi(s_t, \mathbf{a}_{t:t+H-1}) = \mathbb{E}_P\Big[\sum_{u=0}^{H-1} \gamma^u r(s_{t+u}, a_{t+u}) + \gamma^H V^\pi(s_{t+H}) \\ \Big| \; s_t,\; \mathbf{a}_{t:t+H-1},\; s_{t+u+1} \sim P(\cdot \mid s_{t+u}, a_{t+u}) \Big]. \tag{2}$$

Since direct access to this quantity is not available, the TD-MPC family estimates the state-action value by rolling out its learned dynamics and reward models, and bootstrapping with the value network at the end of the rollout. Specifically, given a latent state z_t and an action sequence $\mathbf{a}_{t:t+H-1}$, the state-action value estimate is defined as

$$\begin{aligned} \hat{Q}_H^\pi(z_t, \mathbf{a}_{t:t+H-1}) &= \sum_{u=0}^{H-1} \gamma^u R_\theta(\hat{z}_{t+u}, a_{t+u}) + \gamma^H V_\theta^\pi(\hat{z}_{t+H}), \\ \hat{z}_{t+u+1} &= f_\theta(\hat{z}_{t+u}, a_{t+u}), \end{aligned} \tag{3}$$

where $\hat{z}_t = z_t$. This state-action value estimate, which we also call the return estimate, is then used as the objective for the planner.

Specifically, MPPI is used to solve for the optimal action sequence:

$$\mathbf{a}^*_{t:t+H-1} = \arg \max_{\mathbf{a}_{t:t+H-1}} \hat{Q}_H^\pi(z_t, \mathbf{a}_{t:t+H-1}) \tag{4}$$

Details left out about MPPI and this process can be found in the TD-MPC2 paper (Hansen et al., 2024). Here the first step of the optimal action sequence $\mathbf{a}^*_t$ can be considered as the action sampled from the planning policy π_{plan}. All members of the TD-MPC family sample from this expert policy π_{plan} to act in the environment.

Value-equivalent training The TD-MPC family is part of the value-equivalent (MuZero-style) family which trains the reward and value networks on the latents rolled out by the dynamics model, rather than only the directly encoded latents. For a given trajectory in the replay buffer:

$$z_t, a_t, r_t, ..., z_{t+H-1}, a_{t+H-1}, r_{t+H-1}, z_{t+H}$$

the dynamics head rolls out the same actions $[a_t, ..., a_{t+H-1}]$ from the start state z_t to create the estimated latents:

$$\hat{z}_{t+1}, \ldots, \hat{z}_{t+H} \quad \text{where} \quad \hat{z}_{t+u+1} = f_\theta(\hat{z}_{t+u}, a_{t+u}). \tag{5}$$

These estimated latents are not only used to train the dynamics head, but also to train the reward and value networks. Specifically, the reward head is trained to predict the *true* reward r_{t+u}, using the estimated latent $\hat{z}_{t+u}$. Similarly, the value head is trained to predict the value target corresponding to the *true* latent $y(z_{t+u})$, using only the estimated latent $\hat{z}_{t+u}$. The value target $y(z_{t+u})$ corresponding to the true latent z_{t+u} is calculated as $\hat{Q}_1^\pi(z_{t+u}, a_{t+u} \sim \pi_\theta(z_{t+u}))$. The result of training networks on the rolled out latents is that the value/reward networks become coupled to the dynamics model. If the dynamics model makes a large error, the value/reward networks will be trained to compensate for it.

Planner distillation and reanalyze: During acting, BMPC stores π_{plan} as a policy target and trains π_θ with $D_{\text{KL}}(\pi_\theta(\cdot \mid z_t) \,\|\, \pi_{\text{plan}})$. However, these targets become stale over time, so they are reanalyzed (Schrittwieser et al., 2020) by replanning on replay states. Since these refreshed policy targets are never executed in the environment, there is no new data collected to ensure the model and value heads are accurate on them.

3 Related work

TD-MPC family and planner distillation As discussed in Section 2, reanalyzed planner targets distilled into the policy are never executed in the environment; therefore errors exploited in those targets may never be corrected through interaction with the environment. BOOM (Zhan et al., 2025) and TD-M(PC)^2 (Lin et al., 2026) address this issue by regularizing the policy towards the action taken in the environment, which leads to large improvements on tasks with high-dimensional action spaces. A downside, however, is that the policy is regularized towards stale actions, which is suboptimal. BMPC (Wang et al., 2025) then suggests to train the policy fully with distillation, using reanalyze (Schrittwieser et al., 2020) to refresh the stale policy targets.

Multi-horizon value estimation Combining value estimates across multiple rollout depths is a common strategy in (model-based) RL to reduce the variance of a return estimate. DreamerV3 (Hafner et al., 2025) rolls out policy actions in its learned model and uses TD(λ) (Sutton, 1988) to combine the return estimates from different rollout depths, yielding lower variance value estimates. MCTS-based methods like TSMCTS (Oren et al., 2026) aggregate the value of child nodes at different depths to reduce the variance of their value estimates. To our knowledge, combining return estimates across multiple rollout depths has not been done for MPC objectives.

Dynamics ensembles in model-based reinforcement learning Ensembles of learned dynamics models are widely used in MBRL. PETS (Chua et al., 2018), MBPO (Janner et al., 2019), MOPO (Yu et al., 2020) and STEVE (Buckman et al., 2018) use bootstrapped dynamics ensembles for uncertainty-aware model rollouts. The ensembles reduce variance in the rollouts, but also provide an estimate of the epistemic uncertainty of the model rollout. To our knowledge, none of these methods are value-equivalent/MuZero-style, meaning none train the value/reward networks on the latents rolled out by multiple dynamics models (See section 2).

Pessimism in reinforcement learning Pessimism in reinforcement learning is a strategy to avoid epistemically uncertain regions (Rashidinejad et al., 2021). Pessimism is often implemented as a penalty to the value estimates, either by taking a minimum over an ensemble or subset of the ensemble (Chen et al., 2021; Haarnoja et al., 2018; Oren et al., 2025b; Hansen et al., 2024; Fujimoto et al., 2018) or by subtracting a disagreement based penalty (Yu et al., 2020). This penalty can be applied to value targets (Haarnoja et al., 2018; Chen et al., 2021; Fujimoto et al., 2018), to the policy objective (Mani et al., 2026), or to the planner objective (Chua et al., 2018; Janner et al., 2019; Yu et al., 2020; Oren et al., 2025a). Mrs. Q (Chang et al., 2026) recently showed that pessimism is critical for achieving strong performance on HumanoidBench (Sferrazza et al., 2024), but also notes that it can hurt performance on DMC (Tassa et al., 2018).

4 EfficientTDMPC

We introduce EfficientTDMPC, a TD-MPC-style method that improves the reliability of model-based return estimates used for planning. The TD-MPC family relies heavily on its planner to find actions that maximize the estimated return. Reducing the error in the return estimate can be expected to yield better results from planning. The return estimate is obtained by rolling out the learned dynamics and reward models, and bootstrapping with the value network at the end of the rollout. As such, the return estimate is prone to error from all components, which compounds with rollout depth. EfficientTDMPC makes the planner objective less sensitive to errors in any single learned model rollout.

First, it uses a multi-horizon planning objective that averages the return estimates from rolling out and bootstrapping at different depths, which has been successful in other works that employ MCTS (Browne et al., 2012; Silver et al., 2018; Schrittwieser et al., 2020), and is known to reduce estimator variance under weak assumptions (Oren et al., 2026; Browne et al., 2012). Secondly, it introduces a value-equivalent dynamics ensemble, whose return predictions are averaged, which previous work has shown can reduce error (Chua et al., 2018; Janner et al., 2019; Yu et al., 2020) of model-based return estimates. Third, it introduces pessimistic reanalyze, which prevents distilling action distributions into the policy for which the value estimates are very uncertain. We then introduce two practical training improvements that improve replay buffer freshness and reduce wall-clock training time. We build our contributions on BMPC (Wang et al., 2025), a recent strong contribution to the TD-MPC family.

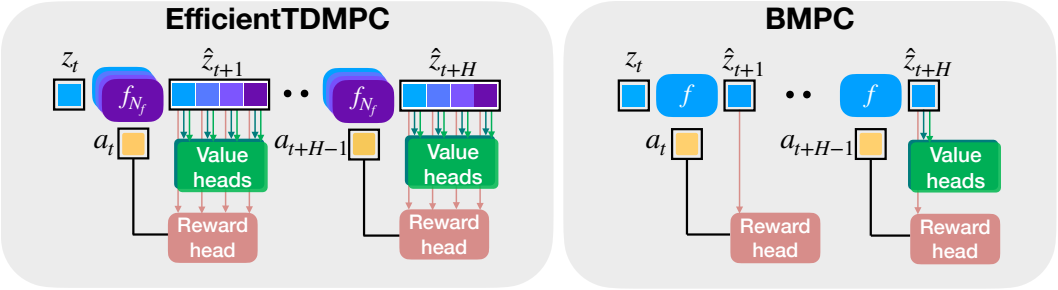


Figure 2: **Process of return estimation of EfficientTDMPC vs BMPC** (Right) BMPC rolls out a single dynamics head, predicts the reward earned at each depth and bootstraps with its value ensemble. (Left) EfficientTDMPC creates multiple rollouts using an ensemble of dynamics models. It then predicts the reward and bootstrapped value at each depth. The estimate is then averaged over the different rollouts and the return estimates from different rollout depths.

4.1 Multi-horizon planning objective

We introduce a multi-horizon planning objective, obtained by averaging the state-action value estimates across different rollout horizons. As discussed in Section 3, aggregating model-based return estimates across different rollout horizons has been a successful way to reduce estimator variance (Sutton, 1988; Feinberg et al., 2018; Buckman et al., 2018). However, these prior works roll out action sequences from the *policy network*, so each depth is an estimate of the same quantity, the value. As such, they are simply averaging estimates of the same quantity to get a reduced variance estimate. In our work, the action sequence is proposed by the MPC planner rather than by the policy network. Bootstrapping with the value function at depth h replaces every later planned action with the policy, so the expected return changes with h. Each term is the value of a different prefix of the planned sequence, not another estimate of the same value. We average these prefix values:

$$\hat{Q}_{\text{aggregate},H}(z_t, \mathbf{a}_{t:t+H-1}) = \frac{1}{H}\sum_{h=1}^{H} \hat{Q}_h(z_t, \mathbf{a}_{t:t+h-1}) \tag{6}$$

where each $\hat{Q}_h$ only evaluates the value of the first h actions in the action sequence, and bootstraps with the value network at depth h. The objective favors action sequences whose prefixes remain valuable under policy continuation, rather than trajectories whose value only becomes high at the terminal planning depth. We use the multi-horizon planning objective during planning at acting time, together with an increase in the rollout horizon.

4.2 Value-equivalent dynamics ensemble

To the best of our knowledge, we are the first to propose to train a value-equivalent dynamics ensemble. We then use this ensemble to create improved value estimates during planning and value target creation. Prior work has shown that dynamics ensembles can reduce the error in model-based return estimates (Chua et al., 2018), but, to the best of our knowledge, this has only been shown for methods that train the dynamics separately from the value and reward networks. We jointly train an ensemble of dynamics heads and choose to train the value and reward networks on the latents rolled out by *all* dynamics heads. We will denote the dynamics heads as $f_{\theta,i}$ for $i \in \{1, .., N_f\}$, where N_f is the number of dynamics heads.

Training We initialize heads differently and train each dynamics head on the same sampled trajectories from the replay buffer $(z_t, a_t, z_{t+1}, ..., z_{t+H-1}, a_{t+H-1}, z_{t+H})$, each then rolls out the action sequence $(a_t, ..., a_{t+H-1})$ from the initial state z_t to create its estimated latents $\hat{z}^i_{t+u+1} = f_{\theta,i}(\hat{z}^i_{t+u}, a_{t+u})$. We then train the reward and value heads to use the estimated latents $\hat{z}^i_{t+u}$ from each dynamics head to predict the true reward r_{t+u} and value target $y(z_{t+u})$ corresponding to the true latent. This implies that the value and reward heads are trained to compensate for the errors of multiple dynamics models. In Section 5.2 we show that introducing an ensemble of dynamics heads into this system can significantly improve sample efficiency. Details in Appendix D.

Value estimation We use this ensemble for improved model based return estimates, which are used during planning and for creating value targets. Similar to TS_∞ sampling from PETS (Chua et al., 2018), we roll out action (sequences) with a single dynamics head, estimate the value and then use the mean and variance over the ensemble. We define a state-action value estimate that uses a single dynamics head with index $i \in \{1, .., N_f\}$ and a single value head with index $j \in \{1, .., N_v\}$ as,

$$\hat{Q}^\pi_H(z_t, \mathbf{a}_{t:t+H-1}; i, j) \;=\; \sum_{u=0}^{H-1} \gamma^u R_\theta(\hat{z}_{t+u}, a_{t+u}) \;+\; \gamma^H V_{\theta,j}(\hat{z}_{t+H}),\; \hat{z}_{t+u+1} = f_{\theta,i}(\hat{z}_{t+u}, a_{t+u}) \tag{7}$$

where $\hat{z}_t = z_t$. This is the state-action value estimate obtained by rolling out the dynamics head with index i and bootstrapping with the value head with index j. We can then obtain $N_f \times N_v$ different estimates of the state-action value by varying i and j. We then estimate the mean of the state-action value and its variance,

$$\hat{Q}^\pi_H(z_t, \mathbf{a}_{t:t+H-1}) = \mathbb{E}_{i,j}[\hat{Q}^\pi_H(z_t, \mathbf{a}_{t:t+H-1}; i, j)] \tag{8}$$

$$\hat{\sigma}^2_{\hat{Q}^\pi_H}(z_t, \mathbf{a}_{t:t+H-1}) = \frac{Var_{i,j}[\hat{Q}^\pi_H(z_t, \mathbf{a}_{t:t+H-1}; i, j)]}{N_f N_v} \tag{9}$$

This variance of the mean estimate, written as $\hat{\sigma}^2_{\hat{Q}^\pi_H}$, is then used in Section 4.3 as a penalty to the planner objective during reanalyze. We acknowledge that this estimate underestimates the true variance because it assumes that the individual estimates are independent, which is not the case since the same value or dynamics head is used for several of them. In practice we find this is not an issue.

4.3 Pessimistic reanalyze

We propose pessimistic reanalyze, which adds an optional penalty to the planner objective during reanalyze, to avoid distilling actions into the policy on which the model and value networks are not well trained. As discussed in Section 3, a known issue in the TD-MPC family is that of a mismatch between the actions taken in the environment and the actions output by the policy network (Zhan et al., 2025). Since the model and value networks are not trained on the policy actions, they are more likely to produce inaccurate predictions for them. Prior work solved this by regularizing the policy towards the action that was taken in the environment, but in later parts of the training that action becomes stale, which can be expected to slow down learning. BMPC solves this by reanalyzing the stale targets, but by doing so it reintroduces the issue where the model may not be trained on the actions output by the policy. EfficientTDMPC introduces pessimistic reanalyze to make policy targets that are fresh, but are also regularized towards actions that the model has actually been trained on. Specifically, we define the pessimistic reanalyze objective as follows:

$$J_{\text{reanalyze},H}(\mathbf{a}_{t:t+H-1}) = \hat{Q}^\pi_H(z_t, \mathbf{a}_{t:t+H-1}) - \beta \hat{\sigma}_{\hat{Q}^\pi_H}(z_t, \mathbf{a}_{t:t+H-1}), \tag{10}$$

where $\hat{\sigma}_{\hat{Q}^\pi_H}(z_t, \mathbf{a}_{t:t+H-1})$ is the square root of the estimated variance defined in Section 4.2. Here β trades off maximizing expected return and minimizing uncertainty. A higher β leads to a planner that finds actions with more certain return estimates, which are more likely to be accurate. Since the reanalyze targets are distilled into the policy, the policy also outputs actions that lead to more certain return estimates, which in turn leads to more reliable value targets. We note that the option to add pessimism is fundamentally a tradeoff that different tasks respond differently to. We find in Section 5.2 that a useful heuristic is to apply pessimism to tasks that have early termination.

4.4 Practical modifications

Data freshness EfficientTDMPC inserts each transition immediately, as opposed to previous work in the TD-MPC family which adds newly collected transitions after a full episode. This shortens the delay before new experience can affect model, value, and policy learning.

Cheap reanalyze planning Reanalyze is responsible for the majority of the wall-clock training time, so we reduce its planning budget by reducing the particles, elites and policy seeds by a factor of eight. We find in Section 5.2 that, for reanalyze, fewer planning particles are sufficient in practice.

5 EXPERIMENTS

We implement our EfficientTDMPC agent using a value-equivalent dynamics ensemble with four dynamics heads. During planning at acting time we increase the rollout horizon from three to six and optimize the multi-horizon planning objective. To save compute we do not apply the multi-horizon planning objective to planning during reanalyze. To decrease compute further we reduce the number of reanalyze planning particles, elites and policy seeds by a factor of 8. We find that pessimistic reanalyze is beneficial on some tasks and harmful on others. The pattern is consistent with early episode termination (tasks that can terminate early tend to benefit; others do not), in line with prior work (Chang et al., 2026; Oren et al., 2025b; Mani et al., 2026). We therefore apply pessimistic reanalyze with a fixed $\beta = 10$ only on tasks that can terminate early. We evaluate on the 7 hard and 21 easy DMC tasks as well as the 14 commonly used HumanoidBench with hand tasks. We describe the baselines used for comparison in Appendix A.3.

5.1 RESULTS

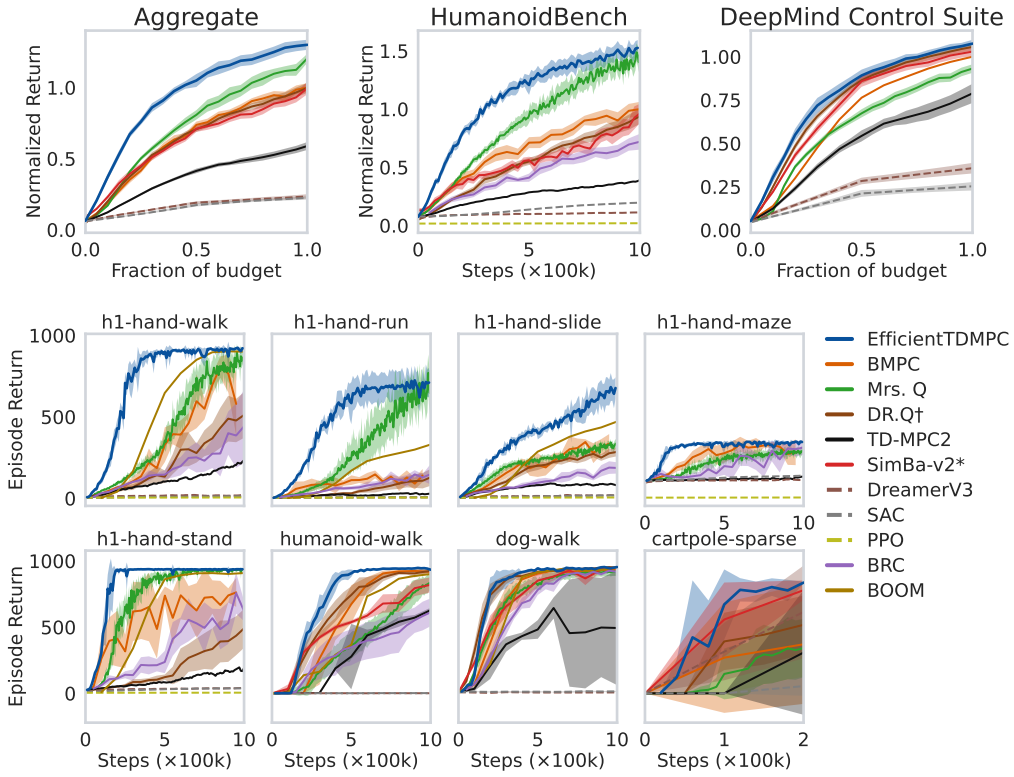


Figure 3: **Learning curves.** Top, left to right: equal-weight aggregate (0.5 HumanoidBench + 0.5 DeepMind Control Suite), HumanoidBench (14 tasks), and the DeepMind Control Suite (suite-equal Easy and Hard). Bottom: learning curves on selected tasks. Shaded regions are 95% CIs. *SimBa-v2 is evaluated on HumanoidBench *without* hands. †DR.Q is missing several HumanoidBench tasks. Full per-task grids are in Appendices B.1, B.2, and B.3.

Figure 3 shows an equal-weight aggregate of HumanoidBench and the DeepMind Control Suite, the two suites separately, and selected tasks where EfficientTDMPC substantially outperforms prior methods, reaching BMPC's final aggregate performance using 57% fewer environment steps. EfficientTDMPC is state of the art in terms of sample efficiency on HumanoidBench and the DeepMind Control Suite.

5.2 Component ablations

Contribution ladder and ensemble scaling. We add our changes to BMPC cumulatively and measure normalized AUC on five representative tasks (quadruped-walk, reacher-hard, dog-stand, humanoid-walk, h1hand-walk; 5–8 seeds). Figure 4a shows the result. Our practical improvements (horizon increase, per-step buffer insertion, cheaper reanalyze) already mildly improve performance; adding the multi-horizon planning objective and the value-equivalent dynamics ensemble then yields further gains, for a final $+56\%$ AUC relative to BMPC. Figure 4b isolates the effect of adding a value-equivalent dynamics ensemble of different sizes to BMPC. We find the AUC on the ablation tasks increases by 10%, 22%, and 33% for 2, 4, and 8 dynamics heads, respectively. Note that EfficientTDMPC uses only 4 dynamics heads to limit wall-clock training time.

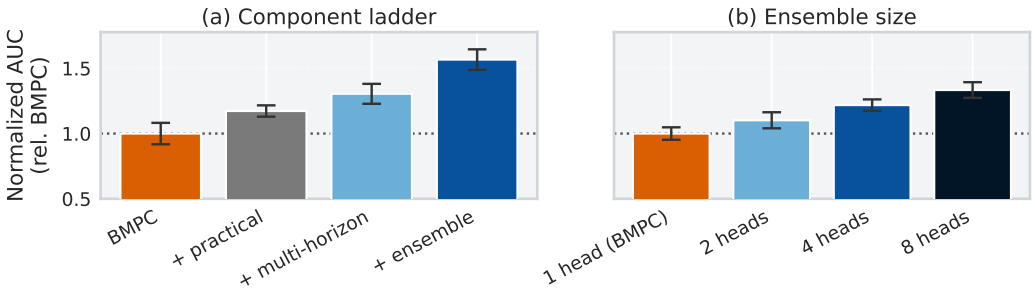


Figure 4: **Contribution ablation.** (a) Cumulative component ladder adding practical changes, then the multi-horizon planning objective, then the value-equivalent dynamics ensemble (4 dynamics heads). We show normalized AUC over 5 tasks (quadruped-walk, reacher-hard, dog-stand, humanoid-walk, h1hand-walk) using 5+ seeds each. (b) Adding a value-equivalent dynamics ensemble of different sizes to BMPC.

Pessimistic reanalyze. Figure 5 analyzes the effect of applying pessimistic reanalyze.

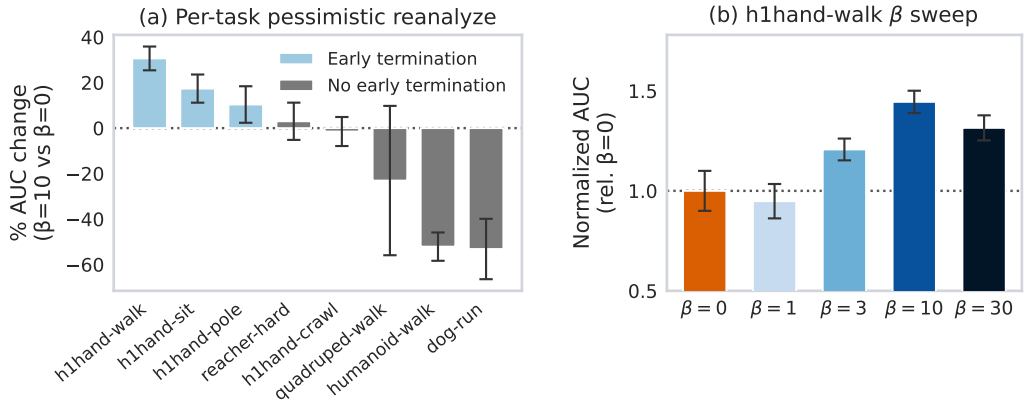


Figure 5: **When to use pessimistic reanalyze, and which β.** (a) Percent AUC change from adding pessimistic reanalyze (β=10 vs. β=0) on h1hand-walk, h1hand-sit, h1hand-pole, h1hand-crawl, reacher-hard, quadruped-walk, dog-run, and humanoid-walk (8 seeds; 95% CIs). Light blue: early termination; grey: no early termination. (b) h1hand-walk normalized AUC for $\beta \in \{0, 1, 3, 10, 30\}$ (6 seeds; 95% CI); β=10 is our default.

Panel (a) reports the percentage change in AUC by adding pessimistic reanalyze (β=10 vs. β=0) on 8 tasks (four HumanoidBench, four DMC). In line with prior work (Chang et al., 2026; Oren et al., 2025b; Mani et al., 2026), we find that tasks that terminate early benefit from pessimistic reanalyze, while tasks that do not, such as DMC tasks and h1hand-crawl, do not benefit. For this reason, we apply pessimistic reanalyze only on tasks that can terminate early. Panel (b) reports normalized AUC

on h1hand-walk for $\beta \in \{0, 1, 3, 10, 30\}$, where pessimistic reanalyze helps. Moderate β improves over β=0, β=10 is best, and β=30 is worse than β=10 but still above β=0.

Cheap reanalyze and wall-clock. Reanalyze dominates training time, so we reduce its planning budget by a factor of eight (particles, elites, and policy seeds). Figure 6 shows that this slightly *improves* HumanoidBench AUC and only slightly reduces DMC, while keeping wall-clock training time similar to BMPC.

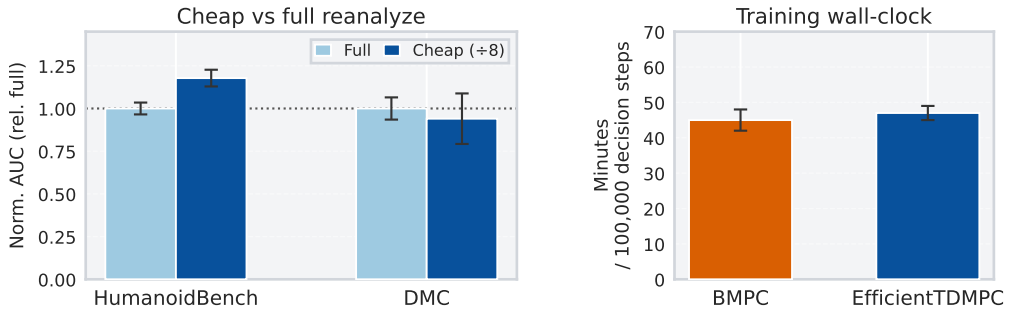


Figure 6: **Cheap reanalyze and wall-clock.** Left: normalized AUC of cheap reanalyze (64 particles, 8 elites, and 3 policy seeds; $\div 8$ budget) relative to full reanalyze (512). HumanoidBench: four h1hand tasks (5 seeds). DMC: quadruped-walk, reacher-hard, dog-stand, humanoid-walk ($\geq$5 seeds). 95% CIs. Right: training wall-clock in minutes per 100,000 decision steps on one NVIDIA A100 (47 ± 2 for EfficientTDMPC vs. 45 ± 3 for BMPC).

6 Conclusion and Limitations

EfficientTDMPC introduces three improvements to MPC objectives in model-based RL. We apply these improvements to a baseline in the TD-MPC family and show that each significantly increases sample efficiency. Firstly, we replace the standard value bootstrapped objective with our multi-horizon planning objective, which requires the estimated value to be high along the full rollout, not just at the end. Secondly, we average the objective over rollouts from multiple dynamics heads. To the best of our knowledge, we are the first to train a value-equivalent dynamics ensemble. Thirdly, we introduce the option of pessimistic reanalyze to the TD-MPC framework, which aims to discourage the policy network from exploiting estimation errors. We find that this is beneficial in environments that can terminate early, so we apply it to those only. We do not yet show that this heuristic is effective on other benchmarks. The dynamics ensemble adds additional compute, but this can be compensated for by making reanalyze planning cheaper, which we find does not significantly hurt performance. To the best of our knowledge, the resulting EfficientTDMPC is state of the art in sample efficiency on HumanoidBench and the DeepMind Control Suite. It also outperforms the baseline's aggregate performance using 57% fewer environment steps.

AI use statement

We used generative AI tools for writing assistance, editing, and LaTeX formatting. We further used AI assistance for the development and analysis of experiments. All method descriptions, claims, and reported numbers were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims, and figures produced with AI assistance.

Ethics statement

This work studies model-based reinforcement learning on simulated continuous-control benchmarks. It does not involve human subjects, personal data, or dual-use models with a direct path to harm. We do not foresee significant ethical risks beyond those of standard RL research.

Reproducibility statement

The method is specified in Section 4 as a modification of BMPC. Hyperparameters, the evaluation protocol, and baseline sources are in Appendix A. Appendix D contains the losses related to the value equivalent dynamics ensemble. We intend to release code upon acceptance.

References

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810. URL https://doi.org/10.1109/TCIAIG.2012.2186810.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/f02208a057804ee16ac72ff4d3cec53b-Abstract.html.

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning, 2026. URL https://arxiv.org/abs/2601.21306.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double Q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=AY8zfZm0tDd.

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. URL https://proceedings.neurips.cc/paper/2018/hash/3de568f8597b94bda53149c7d7f5958c-Abstract.html.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

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. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

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. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/cf73d57b6dcda32b293df7c2d5341f49-Abstract-Conference.html.

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. URL https://proceedings.mlr.press/v162/hansen22a.html.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/5faf461eff3099671ad63c6f3f094f7f-Abstract.html.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 33352–33403. PMLR, 2025. URL https://proceedings.mlr.press/v267/lee25u.html.

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. URL https://proceedings.mlr.press/v331/lin26a.html.

Jiafei Lyu, Zichuan Lin, Scott Fujimoto, Kai Yang, Yangkun Chen, Saiyong Yang, Zongqing Lu, and Deheng Ye. Debiased model-based representations for sample-efficient continuous control, 2026. URL https://arxiv.org/abs/2605.11711.

Kaustubh Mani, Yann Pequignot, Vincent Mai, and Liam Paull. SHAPO: Sharpness-aware policy optimization for safe exploration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7cUxi8LbKD.

Cristian Meo, Mircea Lică, Zarif Ikram, Akihiro Nakano, Vedant Shah, Aniket Rajiv Didolkar, Dianbo Liu, Anirudh Goyal, and Justin Dauwels. Masked generative priors improve world models sequence modelling capabilities. In *ICLR 2025 Workshop on World Models*, 2025. URL https://openreview.net/forum?id=vRdn74Jkqp.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners, 2025. URL https://arxiv.org/abs/2505.23150.

Yaniv Oren, Viliam Vadocz, Matthijs T. J. Spaan, and Wendelin Böhmer. Epistemic monte carlo tree search. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/bcbd670951a6dede2123961de19e5ea3-Abstract-Conference.html.

Yaniv Oren, Moritz A. Zanger, Pascal R. van der Vaart, Mustafa Mert Çelikok, Wendelin Böhmer, and Matthijs T. J. Spaan. Value improved actor critic algorithms. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4a2228d3db1135f13705b0b5844c6708-Abstract-Conference.html.

Yaniv Oren, Joery A. de Vries, Pascal R. van der Vaart, Matthijs T. J. Spaan, and Wendelin Böhmer. Twice sequential monte carlo for tree search. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=rs06uAQpuH.

Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. In *Advances in Neural Information Processing Systems*, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/60ce36723c17bbac504f2ef4c8a46995-Abstract.html.

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. doi: 10.1038/s41586-020-03051-4. URL https://www.nature.com/articles/s41586-020-03051-4.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoid-Bench: Simulated humanoid benchmark for whole-body locomotion and manipulation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024. doi: 10.15607/RSS.2024.XX.061. URL https://www.roboticsproceedings.org/rss20/p061.html.

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. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/10.1126/science.aar6404.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3 (1):9–44, 1988.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. EfficientZero V2: Mastering discrete and continuous control with limited data. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51041–51062. PMLR, 2024. URL https://proceedings.mlr.press/v235/wang24at.html.

Yuhang Wang, Hanwei Guo, Sizhe Wang, Long Qian, and Xuguang Lan. Bootstrapped model predictive control. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/8636419dea1aa9fbd25fc4248e702da4-Abstract-Conference.html.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/a322852ce0df73e204b7e67cbbef0d0a-Abstract.html.

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. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/c290dcfbb071656287ee373cdbe8a259-Abstract-Conference.html.

A IMPLEMENTATION DETAILS

Throughout this paper, HumanoidBench refers to the 14-task HumanoidBench suite with hands (the H1 hand embodiment), not the version without hands.

A.1 HYPERPARAMETERS

Table 2 compares the default EfficientTDMPC configuration used in the main paper against base BMPC. Entries in the EfficientTDMPC column are boldfaced when they differ from BMPC.

Table 2: **EfficientTDMPC versus BMPC hyperparameters.** Bold entries in the EfficientTDMPC column denote values that differ from BMPC. The main-paper agent uses one model/value/policy update per environment step (very similar wall-clock to BMPC).

Category	Parameter	EfficientTDMPC	BMPC
Architecture	Dynamics heads	**4**	1
	Latent dim	512	512
	MLP hidden dim	512	512
	MLP layers	2	2
	Encoder dim / layers	256 / 2	256 / 2
	SimNorm dim	8	8
	Value heads	2	2
	Reward heads	1	1
Training	Batch size	256	256
	Learning rate	3×10^{-4}	3×10^{-4}
	Encoder LR scale	0.3	0.3
	Grad clip norm	20	20
	EMA τ	0.01	0.01
	Model / Value / Policy updates per env step	1 / 1 / 1	1 / 1 / 1
	Reanalyze interval	10	10
Acting planner (MPPI)	Horizon	**6**	3
	Multi-horizon planning objective	**mean**	final
	Iterations	6 (+2 if $\|A\| > 20$)	6 (+2 if $\|A\| > 20$)
	Samples	512	512
	Elites	64	64
	Policy trajectories	24	24
	Temperature	0.5	0.5
	Min / Max std	0.05 / 2.0	0.05 / 2.0
Reanalyze planner (MPPI)	Horizon	3	3
	Multi-horizon objective	final	final
	Iterations	6 (+2 if $\|A\| > 20$)	6 (+2 if $\|A\| > 20$)
	Samples	**64**	512
	Elites	**8**	64
	Policy trajectories	**3**	24
	Temperature	0.5	0.5
	Min / Max std	0.05 / 2.0	0.05 / 2.0
	Batch size	20	20
	Pessimistic reanalyze β	$\boldsymbol{\beta = 10}$ **(early-term. tasks) /** 0 **else**	β=0
Policy	Log std bounds	$[-3, 1]$	$[-3, 1]$
	Entropy coeff	10^{-4}	10^{-4}
	Optimization	distillation	distillation
Discount	$\gamma_{\min}$ / $\gamma_{\max}$	0.95 / 0.995	0.95 / 0.995
	ρ	0.5	0.5
	Value bins / range	101 / $[-10, 10]$	101 / $[-10, 10]$
Replay	Buffer update interval	**every step**	every episode

The acting planner uses horizon 6 with the multi-horizon planning objective (mean over depths), while the reanalyze planner is configured separately with horizon 3, a terminal-only objective, and the smaller 64 / 8 / 3 sample, elite, and policy budget shown above. In code, both planners receive an additional two iterations automatically when the action dimension is greater than 20. Reanalyze uses batch size 20 for both methods and applies pessimistic reanalyze with β=10 only on tasks that can terminate early (otherwise β=0).

A.2 Generating performance figures

We report evaluation return versus environment steps. HumanoidBench is the 14-task suite with hands (h1hand), trained for 1M steps. On the DeepMind Control Suite we follow the standard split of 21 Easy tasks (200k steps, as in EfficientZero V2 (Wang et al., 2024)) and 7 Hard tasks (1M steps). EfficientTDMPC main-paper runs use 4 seeds per task unless a caption says otherwise. Per-task curves for every task are in Appendices B.1–B.3.

Per-task curves. Each point is the mean return over seeds. Shaded regions are 95% confidence intervals ($\text{mean} \pm 1.96\,\text{SE}$).

Normalization. To compare across tasks we divide each task's return by BMPC's mean return on that task at the same budget, so BMPC ends at 1.

Aggregates. Within HumanoidBench, and within Easy or Hard DMC, we average the normalized curves with equal weight per task. The DeepMind Control Suite panel in Figure 3 then averages the Easy and Hard suite means with equal weight, after aligning each task to the fraction of its own budget.

AUC and Figure 1. We interpolate every curve onto a shared step grid, anchoring a missing step 0 at BMPC's untrained score, and take the area under the normalized curve. We compute this separately on HumanoidBench, Easy DMC, and Hard DMC. Suite-equal DMC is the mean of the Easy and Hard AUCs. To highlight EfficientTDMPC's dominance specifically in early environment steps, the introduction bar restricts each of these areas to the first half of that suite's full evaluation budget (100k of 200k on Easy DMC; 500k of 1M on Hard DMC and HumanoidBench). It averages that HumanoidBench AUC with the suite-equal DMC AUC, then divides by BMPC's AUC over the same half so that BMPC equals 1. *SimBa-v2 HumanoidBench numbers use the embodiment without hands, scored against the harder BMPC with hands, and are therefore an upper bound. †DR.Q is missing hurdle, maze, balance_simple, and balance_hard on HumanoidBench.

A.3 Baseline details

We compare EfficientTDMPC against recent model-based and model-free baselines for continuous control. Whenever possible, we use official result files released by the authors. When official results are not available for a specific benchmark setting, we run the official implementation with the benchmark-specific configuration described below.

SAC. Soft Actor-Critic (SAC) (Haarnoja et al., 2018) is an off-policy model-free actor-critic baseline. We use the SAC results provided in the official BMPC repository: `https://github.com/wertyuilife2/bmpc`.

DreamerV3. DreamerV3 (Hafner et al., 2025) is a latent world-model agent trained with imagination-based policy optimization. We use the DreamerV3 results provided in the official BMPC repository: `https://github.com/wertyuilife2/bmpc`.

TD-MPC2. TD-MPC2 (Hansen et al., 2024) is the main TD-MPC-family baseline preceding BMPC, using latent dynamics, value bootstrapping, and MPPI planning. We use the TD-MPC2 results provided in the official BMPC repository: `https://github.com/wertyuilife2/bmpc`.

BMPC. Bootstrapped Model Predictive Control (BMPC) (Wang et al., 2025) is the primary baseline for our method. EfficientTDMPC is built on top of BMPC. We use the official BMPC implementation and released result files from: `https://github.com/wertyuilife2/bmpc`.

BOOM. BOOM (Zhan et al., 2025) is a model-based RL method that bootstraps an off-policy policy using world-model planning. We use the released BOOM result files from the official BOOM repository: `https://github.com/molumitu/BOOM_MBRL`.

BRC. Bigger, Regularized, Categorical (BRC) (Nauman et al., 2025) is a strong model-free baseline based on high-capacity categorical value functions. We use the released result files from the official BRC repository: `https://github.com/naumix/BiggerRegularizedCategorical/tree/main/results`.

SimBa-v2. SimBa-v2 (Lee et al., 2025) is a strong model-free baseline based on hyperspherical normalization. For DMC, we use the released result files from the official SimBa-v2 repository: `https://github.com/DAVIAN-Robotics/SimbaV2/tree/master/results`. For HumanoidBench, official SimBa-v2 results use the H1 embodiment *without* hands. That is a simpler special case of HumanoidBench (with hands); we therefore treat those SimBa-v2 numbers as an upper bound. We run the official implementation with action repeat 1, matching the HumanoidBench protocol.

Mrs. Q. Mrs. Q (Chang et al., 2026) is a recent baseline that studies pessimism and search in model-based reinforcement learning. We use the released result files from the official Mrs. Q repository: `https://github.com/facebookresearch/MRSQ/tree/main/results`.

DR.Q. DR.Q (Lyu et al., 2026) is a recent model-free baseline that learns debiased model-based representations for Q-learning. We use the authors' released result files: `https://github.com/dmksjfl/DR.Q`. On HumanoidBench, DR.Q is missing several tasks (`hurdle`, `maze`, `balance_simple`, `balance_hard`).

PPO. Proximal Policy Optimization (PPO) (Schulman et al., 2017) is included as an additional model-free baseline on HumanoidBench. We use the PPO results from the official HumanoidBench codebase: `https://github.com/carlosferrazza/humanoid-bench`.

EfficientZero V2. EfficientZero V2 (Wang et al., 2024) is an extension of MuZero for sample efficiency. `https://github.com/shengjiewang-jason/efficientzerov2`

A.4 Wall-clock comparison

We compare wall-clock training time of EfficientTDMPC against the official BMPC implementation on a single NVIDIA A100 GPU (excluding compilation and evaluation). Runs were timed over 200k environment steps, which with action repeat 2 is 100k decision steps; we report the original measured times as minutes per 100,000 decision steps (47 ± 2 for EfficientTDMPC vs. 45 ± 3 for BMPC). Both methods use one model, value, and policy update per environment step.

B PER TASK RESULTS

B.1 PER-TASK LEARNING CURVES HUMANOIDBENCH

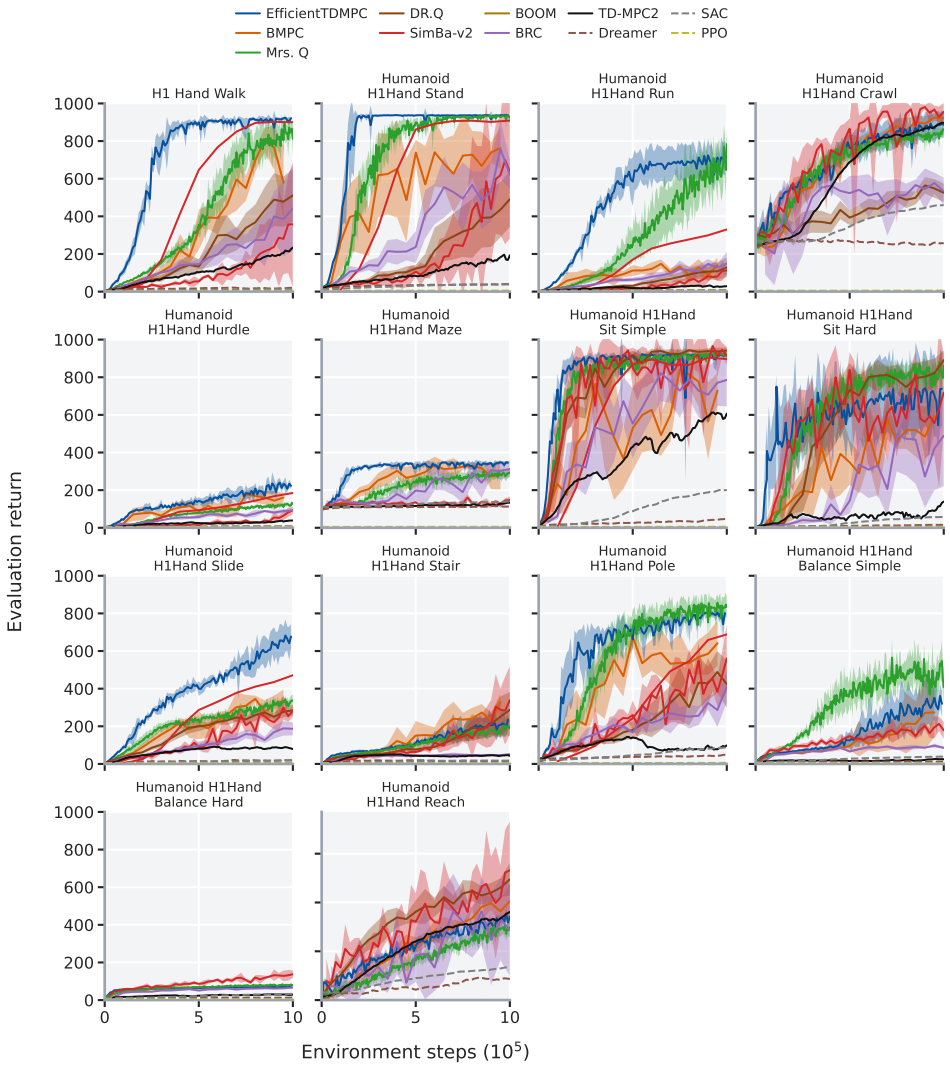


Figure 7: **Per-task results: HumanoidBench (1M environment steps).** Evaluation return on all 14 h1hand tasks with all available baselines. EfficientTDMPC is the default main-paper agent.

B.2 PER-TASK LEARNING CURVES HARD DMC

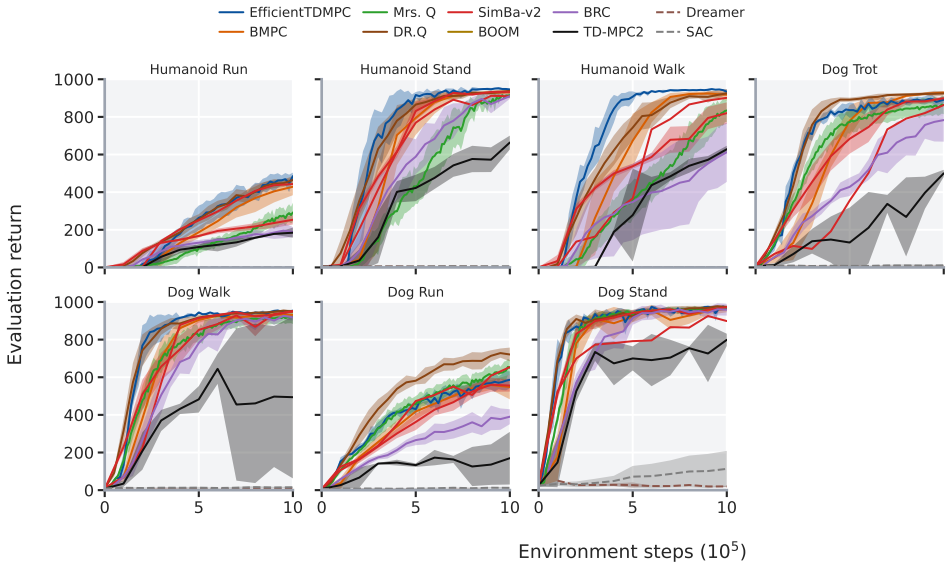


Figure 8: **Per-task results: Hard DMC (1M environment steps).** Evaluation return on all 7 hard DMC tasks with all available baselines. EfficientTDMPC is the default main-paper agent.

B.3 PER-TASK LEARNING CURVES EASY DMC

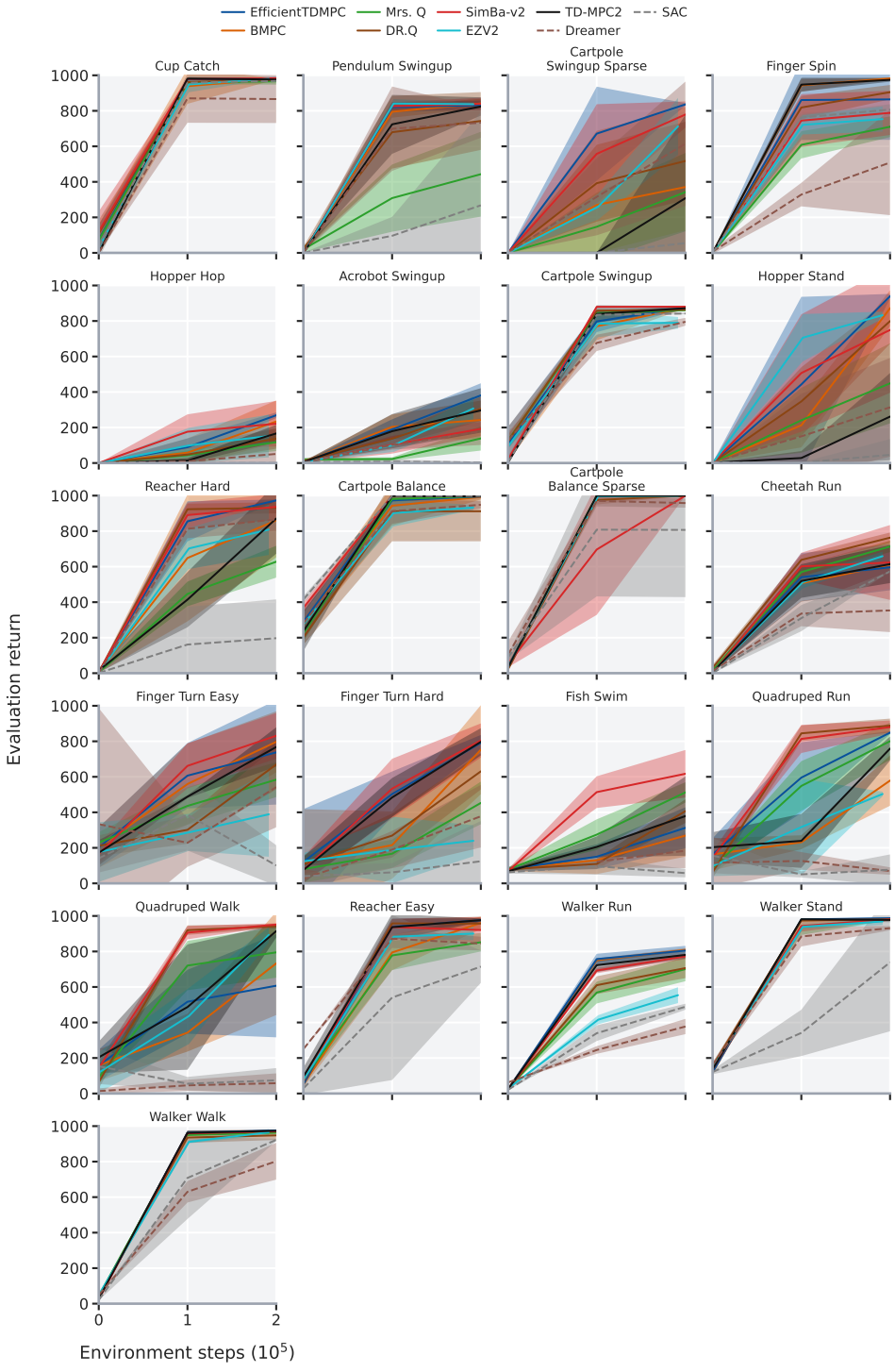


Figure 9: **Per-task results: Easy DMC (200k environment steps).** Evaluation return on all 21 easy DMC tasks with all available baselines. EfficientTDMPC is the default main-paper agent.

B.4 HUMANOIDBENCH PESSIMISTIC-REANALYZE SCOPE

EfficientTDMPC applies pessimistic reanalyze only. We ablate the effect of applying the same uncertainty penalty when acting in the environment as well. On 4 tasks from HumanoidBench, we compare pessimistic reanalyze against applying the penalty during all planning. We find that additionally applying the penalty during acting does not improve performance. We use the same coefficient β=10 for both conditions.

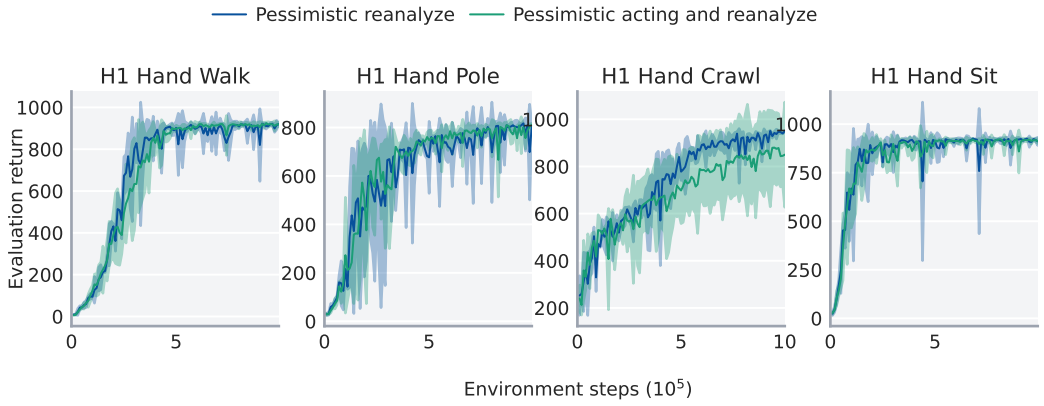


Figure 10: **HumanoidBench pessimistic-reanalyze scope.** Per-task evaluation return versus environment interactions on four HumanoidBench core tasks when the uncertainty penalty is applied during reanalyze only versus during training-time planning, evaluation-time planning, and reanalyze.

C QUALITATIVE ANALYSIS

C.1 PLANNER CROSS-SCORING QUANTITIES

We study the effect of averaging the planner objective over the ensemble. On four tasks we reload agent and replay-buffer checkpoints from training and plan from replay states. Each candidate action sequence is scored with three return estimates. The single-head estimate uses one value head and one dynamics head. The ensemble estimate averages the two value heads and four dynamics heads, which is the EfficientTDMPC acting objective. The environment estimate rolls the same three actions out in the simulator and bootstraps with the learned value heads, which we treat as a proxy for the true return.

Planning is run twice from each state, once under the single-head estimate and once under the ensemble estimate. Both runs use the reduced reanalyze budget: six MPPI iterations, 64 samples, eight elites, three policy trajectories, and temperature 0.5. Every planned sequence, and the policy action from the same state, is then scored under all three estimates. ΔR is the planner score minus the policy-action score, scaled from the three-step window to a 500-step episode. It estimates how much that planner is predicted to outperform the policy action. Figure 11 shows the mean of ΔR over 512 replay states. Error bars are the standard error across those states.

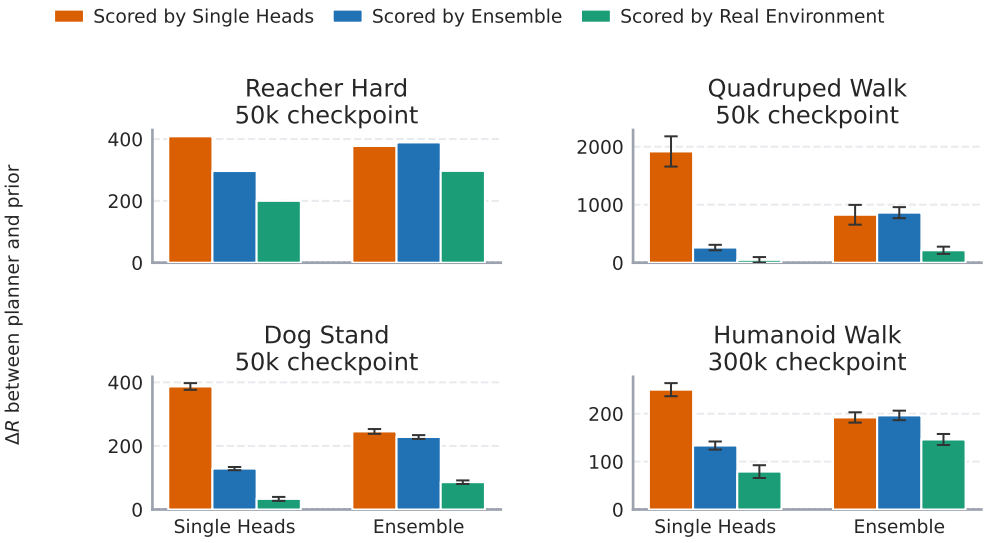


Figure 11: **Ensemble averaging of return estimate reduces single-head planner exploitation.** The plotted ΔR quantity estimates how much each planner is predicted to outperform the policy action from the same state. Bars show means over 512 replay states and error bars show standard errors over states.

Sequences maximized under the single-head estimate receive a very high score from that same estimate. Scored in the simulator, their return is much lower and is often worse than the policy action. Sequences maximized under the ensemble estimate are given a smaller advantage over the policy by their own objective, and that advantage agrees more closely with the simulator. Averaging the return over the ensemble therefore makes the planner less able to exploit an error in one value head and one dynamics head.

C.2 LATENT TRAJECTORY GALLERY

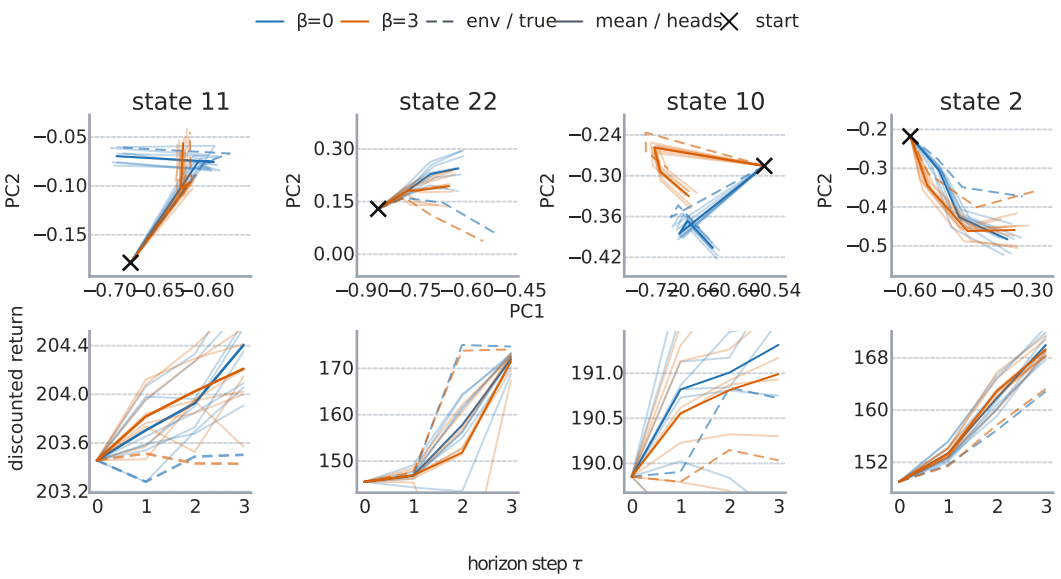


Figure 12: **Latent trajectory gallery on reacher-hard at 50k (four states).** Blue denotes actions from the optimistic planner ($\beta = 0$) and orange from the pessimistic planner ($\beta = 3$). The top row shows the latent trajectory in PCA space, both for each head's predicted trajectory, the mean trajectory across heads, and the true environment. The bottom row shows the estimated return of each trajectory at each depth.

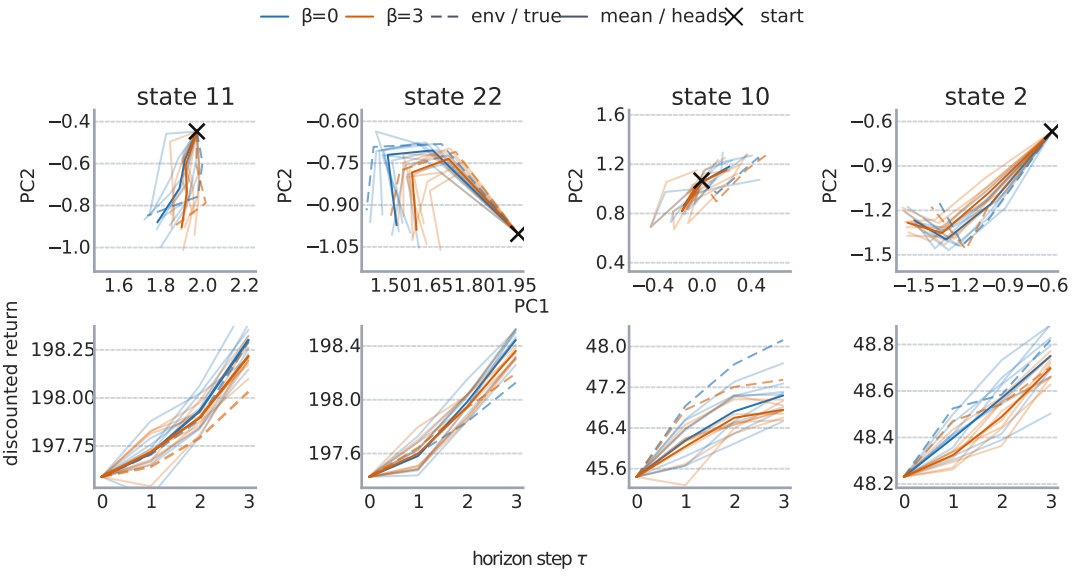


Figure 13: **Latent trajectory gallery on quadruped-walk at 50k (four states).** Same as Figure 12 but for quadruped-walk. Note that for some states the estimated value increases significantly.

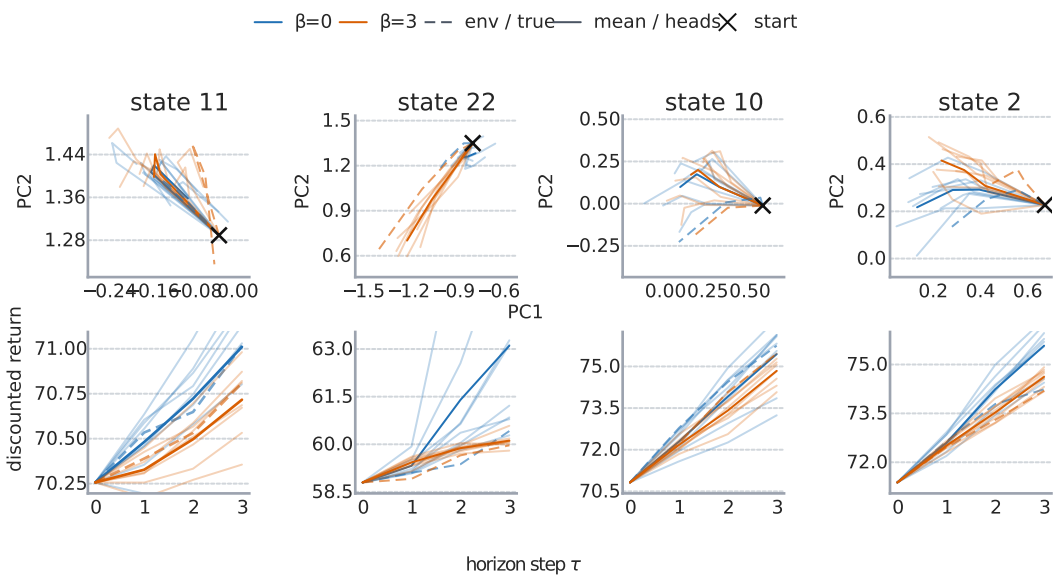


Figure 14: **Latent trajectory gallery on dog-stand at 50k (four states).** Same as Figure 12 but for dog-stand.

Across all three tasks, the state-level gallery makes the uncertainty effect concrete. Even when the predicted return stays competitive, the pessimistic planner tends to keep the per-head rollout bundle tighter than the optimistic planner.

C.3 Task visualization

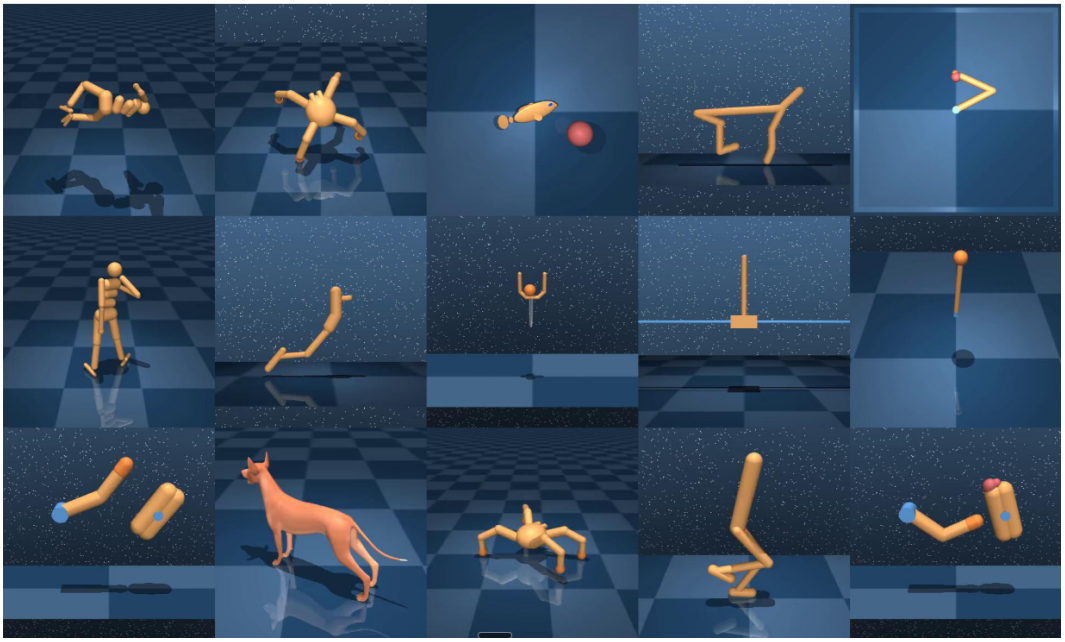


Figure 15: **DMControl tasks visualization.** Images of all the embodiments we control in the DMControl tasks. The tasks include controlling them to run, walk, jump, balance, reach, and perform actions like swing-up and spin, covering a diverse range of continuous control scenarios.

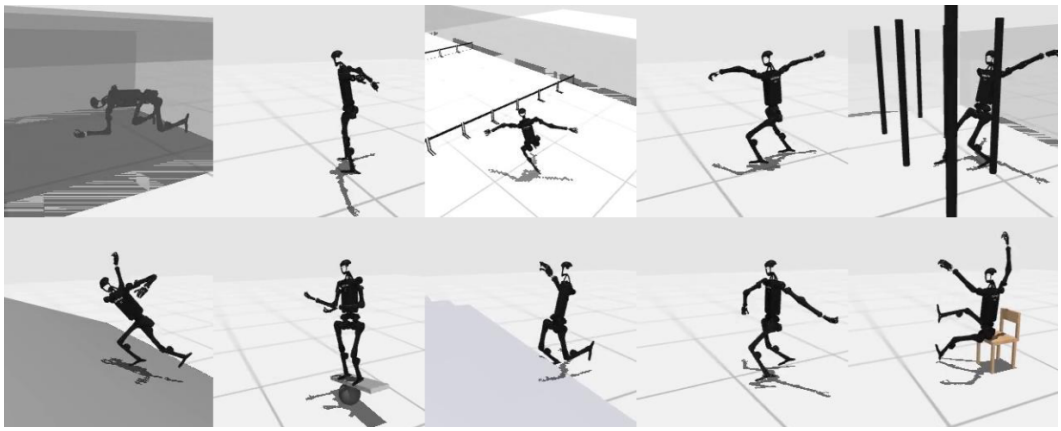


Figure 16: **HumanoidBench visualization.** Images of the Unitree robot we control on HumanoidBench. The tasks include running, walking, crawling, balancing, sitting, reaching, and performing actions like walking on stairs or walking while avoiding collisions with poles.

D TRAINING LOSSES

One update has two backward passes. The first trains the encoder, the dynamics heads, the reward head, and the value heads from one weighted loss. The second trains only the policy, on latents detached from that graph. The target value network $\bar{V}$ is an exponential moving average of the online value heads and receives no gradient.

A replay window is a short trajectory $(s_{t+u}, a_{t+u}, r_{t+u})$ together with the following observation and the stored planner distribution at each state. The encoder maps each observation to $z_{t+u} = h_\theta(s_{t+u})$. Dynamics heads have separate parameters. The reward head and the value heads are shared. Each dynamics head i rolls out the replay actions from the encoded start state,

$$\hat{z}^i_t = z_t, \qquad \hat{z}^i_{t+u+1} = f_{\theta,i}(\hat{z}^i_{t+u}, a_{t+u}). \tag{11}$$

Reward and value losses are applied to every latent of this rollout, including $\hat{z}^i_t = z_t$. Later true latents are stop-gradient targets. Each loss below is averaged over the minibatch and reduced over the window by $\sum_u \rho^u L_u$.

Consistency. Each head is trained by mean squared error onto the encoded next latent,

$$\mathcal{L}_{\text{cons}} = \sum_u \rho^u \frac{1}{N_f} \sum_{i=1}^{N_f} \left\| \hat{z}^i_{t+u+1} - \text{sg}(z_{t+u+1}) \right\|_2^2. \tag{12}$$

This updates $f_{\theta,i}$ along its own rollout and updates h_θ through z_t.

Reward. The shared reward head predicts the environment reward from each head's latent by cross-entropy,

$$\mathcal{L}_{\text{rew}} = \sum_u \rho^u \frac{1}{N_f} \sum_{i=1}^{N_f} \mathcal{L}_{\text{CE}}\big(R_\theta(\hat{z}^i_{t+u}, a_{t+u}),\, r_{t+u}\big). \tag{13}$$

This updates R_θ, and the gradient flows back through $\hat{z}^i$ into the dynamics heads and the encoder. The reward target is the environment reward.

Value. The value target of a true latent z is a one-step policy backup, and it is constant during the model backward. Sample $a \sim \pi_\theta(\cdot \mid z)$ and detach the action. Average the target value heads, then average the backup over dynamics heads:

$$y(z) = \text{sg}\left(R_\theta(z, a) + \gamma \frac{1}{N_f} \sum_{i=1}^{N_f} \bar{V}_\theta\big(f_{\theta,i}(z, a)\big) \right). \tag{14}$$

The same target $y(z_{t+u})$ is used for every dynamics head. Every value head is trained on every head's rollout latent to match it,

$$\mathcal{L}_{\text{val}} = \sum_{u} \rho^u \frac{1}{N_v N_f} \sum_{j=1}^{N_v} \sum_{i=1}^{N_f} \mathcal{L}_{\text{CE}}\big(V_{\theta,j}(\hat{z}^i_{t+u}),\, y(z_{t+u})\big). \tag{15}$$

The cross-entropy is computed per dynamics head and then averaged, so each head is trained to be value-accurate on its own latent. This updates $V_{\theta,j}$, and the gradient flows back through $\hat{z}^i$ into the dynamics heads and the encoder. Because $y(z)$ is stopped, the target network and the reward and dynamics evaluations inside the backup receive no gradient from $\mathcal{L}_{\text{val}}$.

The model loss is a weighted sum of $\mathcal{L}_{\text{cons}}$, $\mathcal{L}_{\text{rew}}$, and $\mathcal{L}_{\text{val}}$. One backward applies the gradient flow above.

Policy. The rollout latents are then detached, and the policy is distilled toward the stored planner distribution by

$$\mathcal{L}_{\pi} = \sum_{u} \rho^u \Big(D_{\text{KL}}\big(\pi_\theta(\cdot \mid \text{sg}(\hat{z}_{t+u})) \,\big\|\, \pi_{\text{plan}}\big) - \alpha\, \mathcal{H}(\pi_\theta) \Big), \tag{16}$$

averaged over dynamics heads. A second backward updates only π_θ. The target value parameters are then moved toward the online value parameters by an exponential moving average.