CAPACITY-AWARE POLICY IMPROVEMENT FOR MODEL-BASED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Model-based reinforcement learning (MBRL) can combine a learned policy with online planning for data-efficient continuous control. However, in planner-guided learning, the planner collects experience, whereas actor updates evaluate actions sampled from the nominal policy. This policy-planner mismatch can expose actor updates to inaccurate value estimates for actions poorly covered by replay data. Critic-based weighting can prioritize promising actions, but concentrating weight on a few samples can amplify the influence of errors in their value estimates. We propose Capacity-Aware Policy Improvement (CAPI), which formulates weighting within a finite actor batch as an entropy-regularized allocation problem with an explicit capacity constraint. The resulting allocation reweights the TD-MPC2 actor objective, prioritizing high-scoring actions while limiting the normalized weight share assigned to each sample. Experiments on high-dimensional continuous-control tasks from the DeepMind Control Suite and HumanoidBench demonstrate that CAPI improves data efficiency and final performance compared to strong baselines.

1 INTRODUCTION

In continuous control, model-free reinforcement learning can require extensive environment interaction (Haarnoja et al., 2018; Schulman et al., 2017). Model-based reinforcement learning (MBRL) can improve data efficiency by using learned dynamics for imagined rollouts and online planning (Sutton, 1991; Janner et al., 2019). Recent latent world-model and MPC methods such as TD-MPC2 and long-horizon visual MPC have shown that learned models can scale to diverse high-dimensional tasks (Hansen et al., 2024; Du et al., 2026).

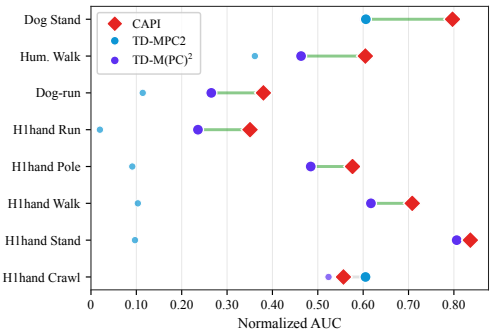


Figure 1: **Sample efficiency (normalized AUC).** Normalized AUC summarizes cumulative performance over the training horizon across 8 representative tasks. Green stems indicate CAPI improvement over the best baseline.

TD-MPC2 combines temporal-difference learning with model predictive control (MPC) in a learned latent space (Hansen et al., 2022; 2024). The planner performs sampling-based optimization of finite-horizon action sequences, with a learned value function estimating the return beyond the planning horizon. In this setting, the nominal policy is trained through critic-guided updates and also supplies candidate actions to the planner (Wang & Ba, 2020; Hansen et al., 2024). The planner selects actions for data collection, whereas actor updates evaluate actions sampled from the nominal policy (Sikchi et al., 2022; Lin et al., 2025). This policy-planner mismatch can leave some actor actions poorly covered by replay data. Their value estimates may therefore be less reliable, exposing actor learning to extrapolation error and value overestimation (Fujimoto et al., 2018; Lin et al., 2025).

Existing approaches address this mismatch by coupling policy learning and planning or regularizing the actor toward planner-generated behavior (Sikchi et al., 2022; Lin et al., 2025). A complementary

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.

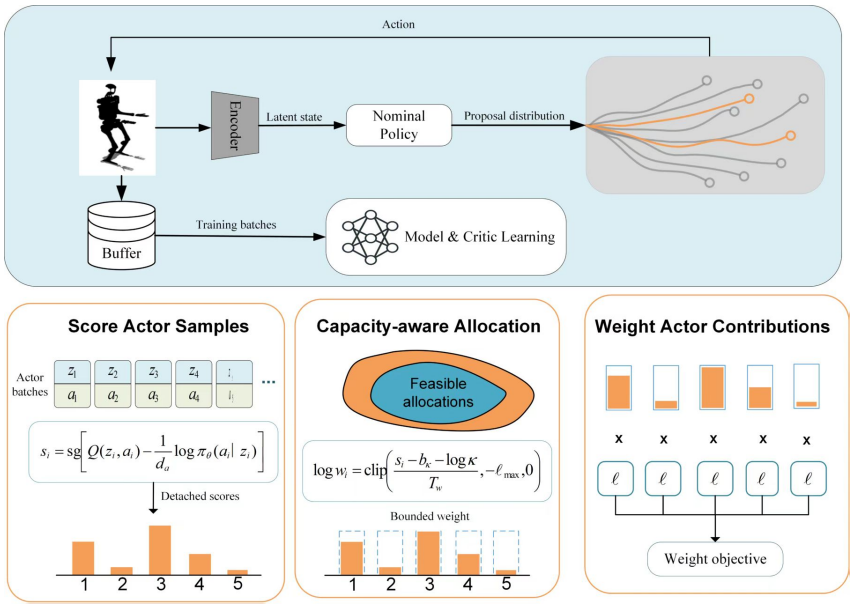


Figure 2: **CAPI within the latent MPC learning loop.** The nominal policy parameterizes the proposal distribution for online planning, and executed actions populate the replay buffer used to train the latent model and critic. During actor learning, CAPI scores actor samples, maps the detached scores to bounded capacity-aware weights, and scales each per-sample actor loss by its corresponding weight.

question concerns how a finite actor batch allocates weight among critic-evaluated actions. Critic-based weighting can prioritize promising actions, but large scores can concentrate the normalized weight on a few samples (Wang et al., 2020). If these scores reflect estimation errors, the affected samples can receive a disproportionate share of the weighted objective. We therefore seek to retain score-based selectivity while explicitly limiting each sample's normalized weight share.

Motivated by this allocation problem, we propose **C**apacity-**A**ware **P**olicy **I**mprovement (**CAPI**). CAPI models the weighting of a finite actor batch as an entropy-regularized allocation problem with a capacity constraint. The allocation prioritizes actions with higher critic-derived scores while bounding each sample's normalized weight share. For a batch of n samples, the constraint limits each share to κ/n, where $\kappa \geq 1$ controls the allowed concentration relative to uniform weighting. We derive a closed-form solution to the capacity-constrained allocation problem. The constraint acts on aggregation weights and can be combined with mechanisms for conservative value estimation and actor-planner alignment. Figure 2 illustrates the overall architecture.

We evaluate CAPI on challenging continuous-control benchmarks from the DeepMind Control Suite and HumanoidBench (Tassa et al., 2018; Sferrazza et al., 2024). Compared with TD-MPC2 and planner-aware regularization baselines, CAPI improves the performance of the MPC policy, outperforming TD-MPC2 by 59.0% on DMControl and 209.8% on HumanoidBench. These performance gains are accompanied by improved sample efficiency, as shown in Figure 1.

2 Related Work

Temporal-Difference Model Predictive Control. Model-based reinforcement learning uses learned dynamics for policy training, online planning, or both (Timor et al., 2026). Temporal-difference model predictive control (TD-MPC) combines short-horizon MPC with value-driven reinforcement learning (Hansen et al., 2022). TD-MPC2 further develops this framework with scalable, robust latent world models for continuous-control tasks and serves as a strong baseline across diverse benchmarks (Hansen et al., 2024). Despite their efficiency, TD-MPC methods struggle in high-dimensional domains due to cumulative model errors, value estimation bias, and policy mismatch (Chang et al., 2026).

Off-Policy Learning with Online Planning. Combining online planning with policy learning creates an off-policy learning problem (Lowrey et al., 2019; Sikchi et al., 2022; M. Moerland et al., 2023). Replay data are generated by planner-improved behavior, whereas value learning and actor improvement are defined with respect to the learned policy (M. Moerland et al., 2023; Hansen et al., 2022; 2024). This policy-planner mismatch can induce out-of-distribution value queries and amplify function approximation errors into persistent value overestimation (Fujimoto et al., 2018; Lin et al., 2025). Drawing on conservative policy improvement in offline RL (Kumar et al., 2020; Fujimoto & Gu, 2021), planning-augmented methods address this mismatch through actor regularization, policy constraints, or continuous planner-to-policy bootstrapping (Sikchi et al., 2022; Lin et al., 2025; Zhan et al., 2026).

Weighted Policy Improvement. Weighted policy improvement involves a trade-off between prioritizing high-scoring actions and controlling the concentration of sample weights. Within a finite actor batch, highly concentrated weights can make the weighted objective more sensitive to errors in the scores of a few samples. A common approach formulates policy improvement as advantage-weighted regression, assigning greater weight to dataset actions with higher estimated advantages (Peng et al., 2019; Nair et al., 2020; Hu et al., 2026). Such weighting can be combined with clipping to limit individual weight magnitudes, as in the exponential variant of critic regularized regression (Wang et al., 2020). Subsequent work examines the bias and variance of batch-normalized advantage-weighted estimators (Cetin et al., 2024; 2025) and explores adaptive weight scaling to balance advantage preservation against gradient variance (Zhang et al., 2026).

3 Preliminaries

3.1 Reinforcement Learning

We consider a discounted Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $p(s'|s,a)$ is the transition dynamics, $r(s,a)$ is the reward function, and $\gamma \in [0,1)$ is the discount factor (Bellman, 1957; Sutton & Barto, 2018). For a policy π, let ρ^π denote the trajectory distribution induced by executing π in the environment. The standard control objective is

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

and the corresponding action-value function is

$$Q^\pi(s,a) = \mathbb{E}_{\tau\sim\rho^\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\,\middle|\, s_0=s, a_0=a\right]. \tag{2}$$

3.2 Latent Model Predictive Control

Latent model predictive control performs prediction and planning in a learned compact representation rather than directly in the raw observation space (Hansen et al., 2024). Specifically, TD-MPC2 uses an observation encoder $z = h(s)$, a latent dynamics model $z' = d(z,a)$, a reward predictor $R(z,a)$, a nominal policy prior $\pi_\theta(\cdot \mid z)$, and an action-value function $Q(z,a) \approx Q^\pi(z,a)$. Here, z denotes a deterministic latent state.

The encoder, dynamics model, reward predictor, and critic are jointly optimized by minimizing a multi-step prediction loss over trajectory segments sampled from the replay buffer $\mathcal{B}$:

$$\mathcal{L}_{\text{model}}(\theta) = \mathbb{E}_{\mathcal{B}}\left[\sum_{t=0}^{H}\rho^t\left(\|d_\theta(z_t,a_t) - \text{sg}(h_\theta(s_{t+1}))\|_2^2 + \text{CE}(\hat{r}_t, r_t) + \text{CE}(\hat{Q}_t, q_t)\right)\right], \tag{3}$$

where sg denotes stop-gradient, CE is the discrete-regression cross-entropy loss, and q_t is a temporal-difference target computed with an exponential-moving-average target critic. The coefficient $\rho \in (0,1]$ controls the decay of loss weights across rollout steps. The nominal policy is optimized separately with an actor objective.

3.3 MPPI PLANNING

During data collection and evaluation, TD-MPC2 acts with a planner policy π_H induced by finite-horizon MPPI rather than directly with the nominal policy. Starting from latent state z_0, the planner forms a population of N candidate action sequences using both Gaussian search samples and trajectories proposed by π_θ. Each candidate is rolled out through the learned dynamics and assigned the estimated return $G^i = \sum_{h=0}^{H-1} \gamma^h R(z_h^i, a_h^i) + \gamma^H Q(z_H^i, a_H^i)$. MPPI converts these returns into normalized exponential weights α^i and updates the Gaussian search distribution at each horizon step as (Williams et al., 2015)

$$\mu_h^* = \sum_{i=1}^{N} \alpha^i a_h^i, \qquad \sigma_h^* = \sqrt{\sum_{i=1}^{N} \alpha^i (a_h^i - \mu_h^*)^2}. \tag{4}$$

After a fixed number of refinement iterations, the planner executes the first action drawn from the final distribution and repeats the procedure in a receding-horizon manner.

4 METHOD

4.1 WEIGHT ALLOCATION WITHIN A FINITE ACTOR BATCH

CAPI formulates weighting within a finite actor batch as an allocation of normalized contributions across samples. At each imagined rollout depth, the actor update uses a batch $\mathcal{D} = \{(z_i, a_i)\}_{i=1}^n$, where z_i is obtained from a model rollout initialized with replay data, and $a_i \sim \pi_\theta(\cdot \mid z_i)$ is sampled from the actor. Let ω_i denote the contribution assigned to sample i, with $\omega \in \Delta_n$ and $\Delta_n = \{\omega \in \mathbb{R}_+^n : \sum_i \omega_i = 1\}$. For the per-sample actor loss $\ell_\pi(z_i, a_i)$, the weighted objective is $\widehat{\mathcal{L}}_\pi(\theta; \omega) = \sum_i \omega_i \ell_\pi(z_i, a_i)$. The allocation is held fixed during differentiation, yielding $\nabla_\theta \widehat{\mathcal{L}}_\pi(\theta; \omega) = \sum_i \omega_i \nabla_\theta \ell_\pi(z_i, a_i)$. Thus, ω_i controls the relative contribution of each sample to the actor update, without bounding the magnitude of its gradient.

Preference weighting can concentrate normalized influence on a few critic-preferred samples. For an unnormalized weight u_i, its normalized share is $\omega_i = u_i / \sum_{j=1}^n u_j$; clipping u_i alone therefore does not bound ω_i, which also depends on the remaining weights. CAPI therefore imposes $\omega_i \le \kappa/n$, where $\kappa \ge 1$ specifies the maximum amplification over the uniform share $1/n$. The choice $\kappa = 1$ recovers uniform weighting, while the constraint becomes inactive for $\kappa \ge n$.

The capacity constraint also bounds cumulative concentration. Let $\omega_{(1)} \ge \cdots \ge \omega_{(n)}$ denote the allocation coefficients in descending order. Then, for every $m \in \{1, \ldots, n\}$,

$$\sum_{j=1}^{m} \omega_{(j)} \le \min\left\{1, \frac{\kappa m}{n}\right\}. \tag{5}$$

Hence, even the m most influential samples cannot jointly receive more than $\min\{1, \kappa m/n\}$ of the total weight, and at least $\lceil n/\kappa \rceil$ samples are required to carry the allocation. This control is particularly relevant in planning-based MBRL, where the updated actor subsequently serves as a proposal distribution for finite-sample planning. The next subsection defines the preference scores and derives the corresponding allocation.

4.2 CAPACITY-AWARE POLICY IMPROVEMENT

We define the detached preference score for each sample as

$$s_i = \mathrm{sg}\left[Q(z_i, a_i) - \frac{1}{d_a} \log \pi_\theta(a_i \mid z_i) \right], \tag{6}$$

where d_a is the action dimension. In the absence of the capacity constraint, entropy regularization yields $\omega_i \propto \exp(s_i)$. Consequently, equation 6 combines an exponential preference for actions with high estimated values and a proposal-density correction normalized by the action dimension.

To translate these scores into update weights, the allocation should prioritize high-scoring samples without concentrating the finite update on only a few samples. We formalize this trade-off as entropy-regularized score maximization under the capacity constraint:

$$\omega^\star = \underset{\omega\in\Delta_n,\ \omega_i\le\kappa/n}{\arg\max} \left\{ \sum_{i=1}^{n} \omega_i s_i + \mathcal{H}(\omega) \right\}, \tag{7}$$

where $\mathcal{H}(\omega) = -\sum_i \omega_i \log \omega_i$ is the Shannon entropy. The choice $\kappa = 1$ recovers uniform allocation, whereas a larger κ permits stronger but explicitly bounded selectivity. The allocation is applied to the actor through the rescaled gate $w_i = n\omega_i/\kappa$, so that $w_i \le 1$. This common rescaling leaves the relative allocation unchanged.

Theorem 1 (Closed-form capacity-aware gate)**.** *Let $\kappa > 1$ and $s_i \in \mathbb{R}$ for $i = 1, \dots, n$. The problem in equation 7 has a unique optimizer $\omega^\star$, and the corresponding ideal gate has the closed form*

$$w_i^\star = \min\left\{1, \frac{\exp(s_i - b_\kappa)}{\kappa}\right\}, \tag{8}$$

where b_κ is the unique solution of

$$\frac{1}{n}\sum_{i=1}^{n} \min\{\kappa, \exp(s_i - b_\kappa)\} = 1. \tag{9}$$

Theorem 1 characterizes allocation within a finite actor batch. To obtain a policy-level counterpart, we isolate action reweighting at a fixed latent state z. For samples $a_i \sim \pi(\cdot \mid z)$, the allocation changes their uniform empirical mass from $1/n$ to ω_i, giving the empirical reweighting ratio $n\omega_i$. Thus, the finite-sample constraint $\omega_i \le \kappa/n$ motivates the population constraint $q(a \mid z) \le \kappa\pi(a \mid z)$. Let $\chi = 1/d_a$ and let Q_χ^π denote the entropy-regularized action value of the current policy (Haarnoja et al., 2018), with $V_\chi^\pi(z) = \mathbb{E}_{a\sim\pi(\cdot|z)}[Q_\chi^\pi(z,a) - \chi \log \pi(a \mid z)]$. At each latent state z, define the ideal update over normalized action densities. The same κ now limits amplification relative to the current policy density:

$$\pi^+(\cdot \mid z) = \underset{q(\cdot|z)\le\kappa\pi(\cdot|z)}{\arg\max} \left\{ \mathbb{E}_{a\sim q}\left[Q_\chi^\pi(z,a) + (1-\chi)\log\pi(a \mid z)\right] + \mathcal{H}(q(\cdot \mid z)) \right\}. \tag{10}$$

The optimal density ratio follows an exponential rule in $Q_\chi^\pi - \chi \log \pi$ until the density constraint becomes active, mirroring the form of Theorem 1. Thus, the statewise operator and the finite gate share a capacity-controlled reweighting rule. The following theorem concerns the exact operator rather than a single sampled actor-gradient step.

Theorem 2 (Capacity-constrained soft policy improvement)**.** *Consider a discounted MDP with bounded rewards and $\gamma \in [0, 1)$. Assume that the policy densities and their entropy-regularized values are finite, and that equation 10 is solved exactly at every state. For any $\kappa \ge 1$, the resulting policy satisfies the following inequality at every state-action pair (z, a):*

$$Q_\chi^{\pi^+}(z,a) \ge Q_\chi^\pi(z,a). \tag{11}$$

Theorem 2, proved in Appendix A.2, establishes improvement for the ideal regularized operator. The implemented actor update applies the same reweighting structure to learned scores within a finite actor batch. Because these scores are estimated from the batch, the exponential mapping at unit temperature can turn modest score differences into large differences in the weights applied to actor gradients. We therefore retain the threshold and saturation structure of Theorem 1, while tempering the score-to-weight mapping used during actor optimization. The resulting log-weight is

$$\log w_i = \mathrm{clip}\left(\frac{s_i - b_\kappa - \log\kappa}{T_w}, -\ell_{\max}, 0\right), \tag{12}$$

where $\mathrm{clip}(x, l, u) = \max\{l, \min\{x, u\}\}$ and $\ell_{\max} > 0$. Here b_κ is the threshold defined in equation 9, so no second normalization is introduced, and $T_w \ge 1$ is the weight temperature. For two samples with $e^{-\ell_{\max}} < w_i, w_j < 1$, $\log(w_i/w_j) = (s_i - s_j)/T_w$; hence, increasing T_w reduces weight contrast without changing the score ordering. The mapping remains monotone in s_i and saturates at one at the same threshold as the ideal solution. With the exact threshold from equation 9,

Algorithm 1 CAPI: Capacity-Aware Policy Improvement

Require: Nominal policy π_θ, latent model $\{h, d, R\}$, critic Q, MPPI planner P, replay buffer $\mathcal{B}$, relative-influence bound κ, weight temperature T_w, log-clip range $\ell_{\max}$

Initialize π_θ, $\{h, d, R\}$, Q, and warm up $\mathcal{B}$ with random actions.
for each training iteration **do**
  // Planner-guided collection
  Planning: $a_t \leftarrow P(\pi_\theta, Q, d, R, h(s_t))$
  Environment step: $(r_t, s_{t+1}) \leftarrow \text{env. step}(a_t)$
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  // Model, critic, and policy learning
  Sample trajectory batch $\{s_t, a_t, r_t, s_{t+1}\}_{t:t+H} \sim \mathcal{B}$
  Update $\{h, d, R, Q\}$ using the standard TD-MPC objective.
  At each rollout depth, form $\{(z_i, a_i)\}_{i=1}^n$ with $a_i \sim \pi_\theta(\cdot|z_i)$ and scores s_i by equation 6.
  Compute b_κ using Algorithm 2 and obtain weights w_i from equation 12.
  Update π_θ using equation 13, optionally including the planner-likelihood term in Section 4.3.
end for

setting $T_w = 1$ recovers the weights in Theorem 1 when the lower clipping bound is inactive. For $T_w \geq 1$, tempering and lower clipping do not decrease the ideal weights, while preserving $w_i \leq 1$. As $\sum_j w_j^\star = n/\kappa$, the normalized practical share still satisfies $w_i / \sum_j w_j \leq \kappa/n$. Thus κ continues to bound each sample's relative influence, even though the practical weights need not sum to n/κ. The numerical solver and its score-clipping step are described in Appendix B.2. For each actor sample, we use the loss

$$\ell_\pi(z_i, a_i) = \alpha_{\text{ent}} \log \pi_\theta(a_i \mid z_i) - \text{scale}\big(\bar{Q}(z_i, a_i)\big)\,,$$

where $\bar{Q}$ is the critic ensemble mean and scale denotes the running Q-value scaling in TD-MPC2. The actor is updated with

$$\mathcal{L}_\pi^{\text{CAPI}} = \frac{1}{n} \sum_{i=1}^{n} \text{sg}[w_i] \ell_\pi(z_i, a_i), \tag{13}$$

where gradients do not pass through the weights.

4.3 Practical Stabilizers

We further incorporate conservative critic scores and optional planner-likelihood regularization into actor learning.

Conservative critic score. The capacity bound limits how much of the actor update one sample can receive, but does not ensure that its critic score is accurate. Overestimated action values can therefore still bias the actor update (Fujimoto et al., 2018; Nauman et al., 2024; Nauman & Cygan, 2025). In equation 6, we obtain $Q(z_i, a_i)$ from the conservative ensemble estimate $\bar{Q}_i - \beta\sigma_i$, where $\bar{Q}_i$ and σ_i are the mean and standard deviation of the critic ensemble. The disagreement penalty lowers the score when critics differ in their evaluations of an action, and β determines the strength of this correction.

Planner-Likelihood Regularization. Capacity-aware allocation can prioritize actions close to planner choices when their critic scores are high. However, the score in equation 6 contains no explicit measure of planner agreement. An action far from the planner choice may therefore receive a large weight when its estimated value is high. We optionally incorporate a local Gaussian reference p_H, formed from the planner-selected action and search standard deviation stored with each collected transition. Specifically, we replace the per-sample loss in equation 13 by $\ell_\pi^{\text{ref}}(z_i, a_i) = \ell_\pi(z_i, a_i) - \lambda \log p_H(a_i \mid z_i)$, retaining the same detached weight w_i. Both the critic-driven loss and the planner-likelihood term are multiplied by the same detached weight. Samples with small capacity weights therefore contribute less to both terms. The reference modifies the weighted actor objective without imposing a separate constraint on the actor policy.

Here λ controls the strength of this optional guidance. The density p_H is a local training reference, not the full action distribution of the MPPI policy π_H. We disable this term on DMControl ($\lambda = 0$)

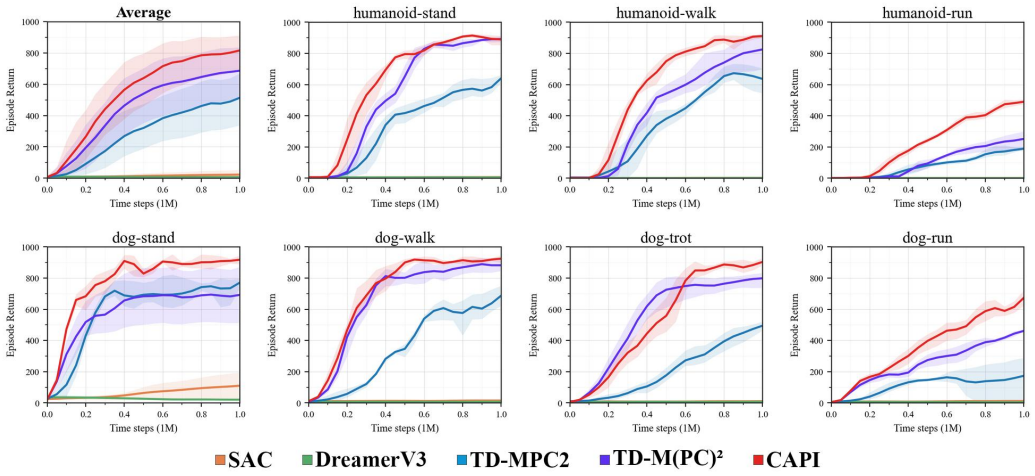


Figure 3: Learning curves on 7 DMControl tasks. Curves show the mean episode return over 5 seeds, with shaded regions indicating 95% bootstrap confidence intervals.

and enable it on HumanoidBench ($\lambda = 1$). This term is applied after the capacity-aware weights are determined and therefore does not affect the threshold or the allocation itself. We examine its empirical effect in Section 5.3 and Appendix B.1.

5 Experiments

5.1 Experimental Setup

We evaluate CAPI on two challenging continuous-control benchmarks. From the DeepMind Control Suite (DMControl) (Tassa et al., 2018), we select the high-dimensional *Humanoid* ($\mathcal{S} \in \mathbb{R}^{67}, \mathcal{A} \in \mathbb{R}^{21}$) and *Dog* ($\mathcal{S} \in \mathbb{R}^{223}, \mathcal{A} \in \mathbb{R}^{38}$) domains. We also evaluate all 11 locomotion tasks from HumanoidBench (Sferrazza et al., 2024) featuring the Unitree H1 robot ($\mathcal{S} \in \mathbb{R}^{151}, \mathcal{A} \in \mathbb{R}^{61}$). All tasks use state observations. The hyperparameters are reported in Table 3. We train for 1×10^6 environment steps on every task in both DMControl and HumanoidBench. To account for run-to-run variability, we report mean performance with 95% bootstrap confidence intervals over five independent seeds.

We choose four representative online RL algorithms as our baselines: (1) **SAC** (Haarnoja et al., 2018), a maximum-entropy off-policy actor-critic; (2) **DreamerV3** (Hafner et al., 2025), a model-based agent that trains its policy entirely through latent imagination; (3) **TD-MPC2** (Hansen et al., 2024), which combines a learned latent dynamics model with online MPPI planning guided by a policy prior; (4) **TD-M(PC)**^{2} (Lin et al., 2025), which adds a reverse-KL constraint anchoring the policy toward planner behavior to mitigate policy-planner mismatch. Further baseline details are in Appendix B.4.

5.2 Benchmark Results

Figure 3 summarizes the performance of CAPI and baselines across 7 DMControl tasks (full results in Appendix Table 1). On the DMControl suite, CAPI achieves an average return of 820.8, improving over TD-MPC2 by 59.0% (516.3) and over TD-M(PC)^2 by 18.6% (692.0). The gap is largest on high-dimensional Dog tasks. On Dog-run, CAPI reaches 686.3 compared with 179.2 for TD-MPC2 and 472.9 for TD-M(PC)^2. On Dog-stand, CAPI attains 939.6, whereas individual TD-M(PC)^2 runs range from 464 to 945. TD-M(PC)^2 therefore matches CAPI in its strongest run but exhibits substantially greater variation across seeds. CAPI achieves a higher mean return with a narrower confidence interval on this task. Section 5.3 further examines how planner guidance interacts with capacity-aware allocation.

On the HumanoidBench suite, CAPI attains an average return of 650.9 across 11 tasks, improving over TD-MPC2 by 209.8% (210.1) and over TD-M(PC)^2 by 21.0% (537.8). The largest gains occur on tasks requiring precise whole-body coordination and balance. On H1hand-balance and H1hand-

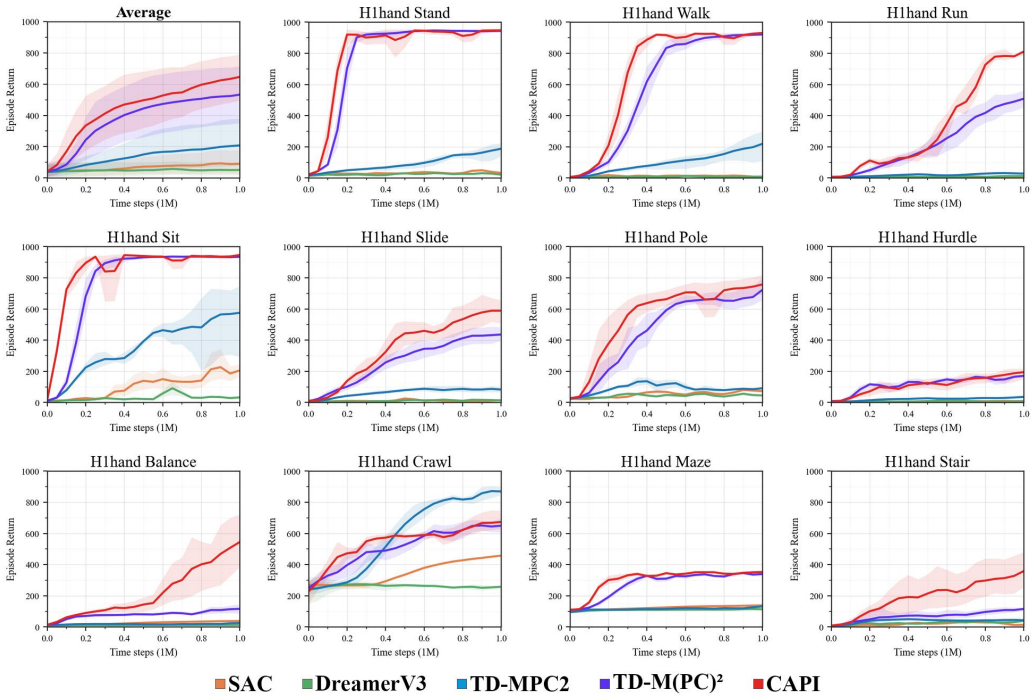


Figure 4: Learning curves on 11 HumanoidBench H1hand locomotion tasks plus the average across all tasks. Curves show the mean episode return over 5 seeds, with shaded regions indicating 95% bootstrap confidence intervals.

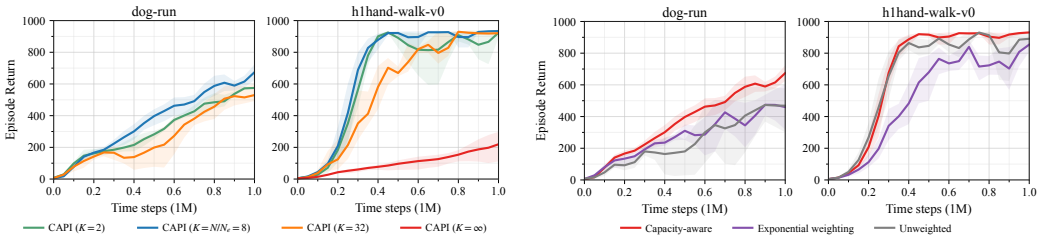
stair, CAPI reaches 556.9 and 367.2, whereas TD-M(PC)^2 achieves 112.7 and 124.0. TD-M(PC)^2 also exhibits wider confidence intervals on several HumanoidBench tasks.

Both benchmarks show that CAPI reaches competitive performance with substantially fewer environment steps than prior methods, with per-task normalized AUC scores reported in Appendix B.5.

5.3 Ablation Study

Ablation of the capacity ratio. We vary κ to examine how weight concentration within a finite actor batch affects learning (Figure 5(a)). In equation 7, κ limits each sample's share to κ/n, which is κ times the uniform share $1/n$. The default $\kappa = 8$ coincides numerically with the planner's sample-to-elite ratio in these experiments, but the ratio provides a convenient scale rather than a theoretically optimal capacity. Among the tested values, $\kappa = 8$ achieves the highest returns on *Dog-run* and reaches high returns earlier on *H1hand-walk*. The tighter $\kappa = 2$ remains competitive on the latter task, whereas $\kappa = 32$ learns more slowly there. Together, these results show sensitivity to the allocation constraint without identifying a universally optimal value.

Ablation of the allocation rule. Figure 5(b) compares capacity-aware allocation with normalized exponential weighting and an unweighted actor update on *Dog-run* and *H1hand-walk*. The exponential baseline assigns $w_i = n\,\mathrm{softmax}(s_i/T_w)$ and uses the same scores, weight temperature, and auxiliary terms as CAPI. It preserves selectivity based on the scores but exhibits substantially greater variation across seeds on *Dog-run*. The exponential growth of relative weights with score differences, together with the absence of an explicit bound on the normalized share assigned to each sample, allows a spuriously high score to cause an incorrectly ranked sample to dominate the actor update, thereby reducing update consistency. On *H1hand-walk*, exponential weighting also learns more slowly and reaches a final return of 861.1, compared with 932.3 for CAPI. CAPI retains the same preference for high scores while limiting every normalized share to κ/n, which restricts the influence of such samples. The unweighted update assigns equal coefficients to all samples and therefore avoids concentration driven by score weighting, but it also discards the selectivity needed to prioritize actor samples with higher scores. It learns more slowly on *Dog-run* and achieves lower mean final returns on both tasks, although it remains competitive on *H1hand-walk*. Together, these



(a) Capacity-ratio ablation.

(b) Allocation-rule ablation.

Figure 5: **Capacity and allocation ablations.** In (a), the default $\kappa = 8$ performs best on *Dog-run* and reaches high returns earlier on *H1hand-walk*. In (b), capacity-aware allocation outperforms normalized exponential weighting and the unweighted actor update on both tasks. Curves show mean returns, with shaded regions indicating 95% bootstrap confidence intervals obtained by resampling independent runs within each setting.

results indicate that CAPI benefits from combining selectivity guided by critic scores with explicit control over allocation concentration.

Ablation of planner-likelihood regularization. To examine how CAPI interacts with planner-likelihood regularization across different action dimensions, we ablate λ on Dog-run ($d_a = 38$) and H1hand-walk ($d_a = 61$). Since the CAPI score does not explicitly measure planner agreement, this comparison probes the role of likelihood guidance under a fixed capacity rule. Adding the likelihood term moves actor samples closer to planner-selected actions on both tasks and increases the fraction of samples assigned the minimum weight (Appendix B.1). Yet return falls on Dog-run and rises substantially on H1hand-walk. This contrast reveals that planner alignment changes the actor proposals and the realized weight distribution, but does not consistently track return in these tasks.

6 Conclusion

We introduced CAPI, which formulates weighting within a finite actor batch as an entropy-regularized allocation problem subject to a capacity constraint. Motivated by policy-planner mismatch, CAPI prioritizes high-scoring actions while bounding each sample's share of the total weight. Our analysis provides a closed-form solution to the ideal allocation problem and establishes explicit bounds on weight concentration. Experiments on DMControl and HumanoidBench demonstrate improved sample efficiency and final performance on high-dimensional locomotion tasks.

More broadly, CAPI offers an alternative approach to policy improvement under policy-planner mismatch, shifting the focus from constraining the actor distribution to capacity-constrained allocation of update weights. Future work could formulate policy improvement over a shared pool of actor and planner proposals, allowing a single capacity-constrained allocation to determine how both sources shape the update.

AI use statement

In this work, we used generative AI tools for writing assistance, including text polishing, grammatical correction, and proofreading. We also used these tools to provide guidance on the design of ablation studies and to assist with a limited portion of the empirical result analysis. We have not used generative AI tools for core scientific conceptualization, formulating research hypotheses, theoretical derivations, or synthesizing primary empirical findings, and no other disclosure-mandated tasks are applicable to this work.

We have reviewed all AI-assisted work: all suggested textual edits and code snippets were manually vetted, tested, and integrated by the authors to ensure factual accuracy, technical correctness, and the absence of hallucinated citations. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility statement

The main paper presents the CAPI formulation and experimental protocol. The appendix provides implementation details, hyperparameters, and additional analyses. An anonymized implementation and configuration files are available at `https://anonymous.4open.science/r/capi-source-code-3A1B`.

References

Richard Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684, 1957.

Edoardo Cetin, Andrea Tirinzoni, Matteo Pirotta, Alessandro Lazaric, Yann Ollivier, and Ahmed Touati. Simple ingredients for offline reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2024.

Edoardo Cetin, Ahmed Touati, and Yann Ollivier. Finer behavioral foundation models via autoregressive features and advantage weighting. *Reinforcement Learning Journal*, 6:1647–1680, 2025.

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2026.

Yurui Du, Pinhao Song, Yutong Hu, and Renaud Detry. Elvis: Ensemble-calibrated latent imagination for long-horizon visual mpc. *arXiv preprint arXiv:2605.04709*, 2026.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

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

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

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Nick Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*. PMLR, 2022.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Edward Hu, Jie Wang, Xingfang Yuan, Fiona Luo, Muyao Li, Gaspard Lambrechts, Oleh Rybkin, and Dinesh Jayaraman. Real-world reinforcement learning of active perception behaviors. *Advances in Neural Information Processing Systems*, 38:117272–117308, 2026.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Haotian Lin, Pengcheng Wang, Jeff Schneider, and Guanya Shi. Td-m(pc)^2: Improving temporal difference mpc through policy constraint. *arXiv preprint arXiv:2502.03550*, 2025.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control. In *International Conference on Learning Representations (ICLR)*, 2019.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 16(1):1–118, 2023.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Michal Nauman and Marek Cygan. Decoupled policy actor-critic: Bridging pessimism and risk awareness in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19633–19641, 2025.

Michał Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzciński, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. In *41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 37342–37364. PMLR, 2024.

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

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

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

Harshit Sikchi, Wenxuan Zhou, and David Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pp. 1622–1633. PMLR, 2022.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

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.

Nadav Timor, Ravid Shwartz-Ziv, Micah Goldblum, Yann LeCun, and David Harel. On training in imagination. *arXiv preprint arXiv:2605.06732*, 2026.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*, 2020.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Siegel Noah, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7768–7778, 2020.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Guojian Zhan, Likun Wang, Xiangteng Zhang, Jiaxin Gao, Masayoshi Tomizuka, and Shengbo Eben Li. Bootstrap off-policy with world model. *Advances in Neural Information Processing Systems*, 38:134093–134121, 2026.

Hongyin Zhang, Shiyuan Zhang, Junxi Jin, Qixin Zeng, Yifan Qiao, Hongchao Lu, and Donglin Wang. Balancing signal and variance: Adaptive offline rl post-training for vla flow models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18755–18763, 2026.

A Theoretical Analysis

A.1 Proof of Theorem 1.

Proof. Set $w_i = n\omega_i/\kappa$. This gives a bijection between the feasible set in equation 7 and $\mathcal{W}_\kappa := \{w \in [0,1]^n : n^{-1}\sum_i w_i = 1/\kappa\}$. With the convention $0 \log 0 = 0$, define $\Psi_s(w) := \sum_i (w_i s_i - w_i \log w_i)$. A direct substitution gives

$$\left\langle \frac{\kappa}{n} w, s \right\rangle + \mathcal{H}\Big(\frac{\kappa}{n} w\Big) = \frac{\kappa}{n} \Psi_s(w) - \log \frac{\kappa}{n}.$$

Consequently, the original problem is equivalent to maximizing Ψ_s over $\mathcal{W}_\kappa$. The feasible set is compact and convex, and Ψ_s is strictly concave, so the optimizer exists and is unique. Moreover, $w_i = 1/\kappa$ is strictly feasible because $\kappa > 1$. The entropy derivative diverges as $w_i \downarrow 0$, so every optimal coordinate is positive and the lower-bound constraints are inactive.

Introduce a multiplier $\lambda \in \mathbb{R}$ for $\sum_i w_i = n/\kappa$ and multipliers $\nu_i \geq 0$ for $w_i \leq 1$. The Lagrangian is

$$\mathcal{L}(w, \lambda, \nu) = \Psi_s(w) - \lambda \left(\sum_{i=1}^n w_i - \frac{n}{\kappa} \right) - \sum_{i=1}^n \nu_i (w_i - 1).$$

Slater's condition holds, so the KKT conditions are necessary and sufficient. Stationarity gives $s_i - \log w_i^\star - 1 - \lambda - \nu_i = 0$, while complementary slackness gives $\nu_i(w_i^\star - 1) = 0$. If $w_i^\star < 1$, then $\nu_i = 0$ and $w_i^\star = \exp(s_i - 1 - \lambda)$. If $w_i^\star = 1$, stationarity requires $s_i - 1 - \lambda = \nu_i \geq 0$. Defining $b_\kappa := 1 + \lambda - \log \kappa$ combines both cases as

$$w_i^\star = \min\left\{1, \frac{\exp(s_i - b_\kappa)}{\kappa}\right\}.$$

It remains to determine the threshold. Define $R_\kappa(b) := n^{-1} \sum_i \min\{1, \exp(s_i - b)/\kappa\}$. This function is continuous and non-increasing, with limits one and zero as b tends to negative and positive infinity, respectively. Whenever $R_\kappa(b) < 1$, at least one term is unsaturated and decreases strictly with b; hence R_κ is strictly decreasing on the range containing $1/\kappa$. Because $1/\kappa \in (0, 1)$, there is a unique b_κ such that $R_\kappa(b_\kappa) = 1/\kappa$. Multiplying this equality by κ yields equation 9. Finally, $\omega^\star = \kappa w^\star / n$, which proves the stated closed form and the uniqueness of the optimizer. □

A.2 Proof of Theorem 2.

Proof. For a fixed state z, denote the objective in equation 10 by $\mathcal{F}_z(q)$. Its strict concavity and the KKT conditions give

$$\frac{\pi^+(a \mid z)}{\pi(a \mid z)} = \min\left\{\kappa, \exp\big(Q_\chi^\pi(z,a) - \chi \log \pi(a \mid z) - b_\kappa(z)\big)\right\},$$

where $b_\kappa(z)$ normalizes $\pi^+(\cdot \mid z)$. This is the population counterpart of the finite allocation in Theorem 1.

Using $\mathcal{H}(q) = -\mathbb{E}_q[\log q]$, the same objective can be written as

$$\begin{aligned} \mathcal{F}_z(q) = \mathbb{E}_{a \sim q}\left[Q_\chi^\pi(z,a) - \chi \log q(a \mid z)\right] \\ - (1 - \chi) D_{\mathrm{KL}}(q(\cdot \mid z) \,\|\, \pi(\cdot \mid z)). \end{aligned}$$

Because $\kappa \geq 1$, the current policy is feasible in equation 10. Since $1 - \chi \geq 0$, optimality of π^+ and non-negativity of the KL divergence imply

$$\mathbb{E}_{a \sim \pi^+}\left[Q_\chi^\pi(z,a) - \chi \log \pi^+(a \mid z)\right] \geq \mathcal{F}_z(\pi^+) \geq \mathcal{F}_z(\pi) = V_\chi^\pi(z).$$

Let $\mathcal{T}_\chi^q$ be the entropy-regularized Bellman operator for a policy q. Substituting the preceding inequality into this operator yields

$$\begin{aligned} \left(\mathcal{T}_\chi^{\pi^+} Q_\chi^\pi\right)(z,a) &= r(z,a) + \gamma\, \mathbb{E}_{z'} \mathbb{E}_{a' \sim \pi^+(\cdot \mid z')}\left[Q_\chi^\pi(z',a') - \chi \log \pi^+(a' \mid z')\right] \\ &\geq r(z,a) + \gamma\, \mathbb{E}_{z'} V_\chi^\pi(z') \\ &= Q_\chi^\pi(z,a). \end{aligned}$$

By monotonicity of $\mathcal{T}_\chi^{\pi^+}$, the preceding Bellman inequality can be applied repeatedly: for every integer $m \geq 1$,

$$Q_\chi^\pi \leq \left(\mathcal{T}_\chi^{\pi^+}\right)^m Q_\chi^\pi.$$

Since the operator is a γ-contraction, these iterates converge to its unique fixed point $Q_\chi^{\pi^+}$. Taking the limit in the inequality proves equation 11. □

B Additional Experimental Details

Table 1: Episode return at 1×10^6 environment steps across 7 DMControl and 11 HumanoidBench tasks. [Bracketed values] show 95% bootstrap confidence intervals over 5 seeds. **Bold** indicates the best mean.

Task	SAC	DreamerV3	TD-MPC2	TD-M(PC)^2	CAPI (Ours)
DeepMind Control Suite (DMControl)					
Humanoid-stand	6.4 [6, 6]	4.7 [4, 5]	663.7 [628, 692]	**899.9 [892, 908]**	885.9 [861, 906]
Humanoid-walk	2.0 [1, 2]	1.2 [1, 2]	620.1 [540, 676]	829.0 [705, 919]	**915.8 [905, 925]**
Humanoid-run	1.2 [1, 1]	0.6 [0, 1]	186.2 [175, 207]	255.1 [186, 305]	**502.9 [473, 533]**
Dog-stand	113.1 [30, 199]	20.5 [18, 22]	798.9 [776, 831]	698.5 [515, 882]	**939.6 [925, 961]**
Dog-walk	15.3 [12, 19]	6.7 [6, 7]	679.5 [633, 735]	883.1 [837, 919]	**920.3 [901, 932]**
Dog-trot	11.1 [8, 14]	7.4 [6, 8]	486.7 [457, 520]	805.5 [751, 837]	**894.4 [859, 927]**
Dog-run	11.9 [6, 17]	4.5 [4, 5]	179.2 [15, 288]	472.9 [470, 476]	**686.3 [636, 734]**
AVG.	23.0 [5, 54]	6.5 [3, 12]	516.3 [340, 675]	692.0 [516, 840]	**820.8 [699, 916]**
HumanoidBench (Unitree H1hand)					
H1hand-stand	26.8 [21, 33]	26.0 [11, 45]	192.7 [150, 229]	944.6 [938, 952]	**948.0 [935, 956]**
H1hand-walk	8.0 [5, 11]	11.2 [11, 12]	234.1 [125, 307]	920.4 [912, 929]	**932.3 [927, 936]**
H1hand-run	8.0 [7, 10]	14.3 [14, 15]	28.5 [27, 30]	518.0 [461, 572]	**828.1 [821, 840]**
H1hand-sit	194.4 [144, 244]	37.3 [29, 46]	573.8 [274, 757]	936.2 [932, 940]	**948.2 [946, 951]**
H1hand-slide	12.3 [11, 14]	6.5 [5, 8]	78.8 [71, 89]	432.6 [405, 467]	**590.5 [500, 675]**
H1hand-pole	76.8 [59, 95]	51.9 [44, 57]	99.6 [93, 107]	755.9 [683, 853]	**758.5 [692, 825]**
H1hand-hurdle	8.7 [7, 10]	3.6 [3, 4]	38.9 [26, 49]	169.9 [160, 177]	**193.7 [156, 215]**
H1hand-balance	39.0 [37, 41]	12.6 [12, 13]	24.9 [24, 26]	112.7 [84, 142]	**556.9 [405, 729]**
H1hand-crawl	459.2 [455, 465]	255.8 [245, 266]	**863.4 [823, 909]**	650.0 [612, 688]	684.1 [646, 745]
H1hand-maze	139.6 [137, 143]	112.8 [111, 114]	133.4 [123, 153]	351.1 [344, 356]	**352.5 [349, 354]**
H1hand-stair	14.4 [5, 25]	44.2 [40, 47]	43.1 [36, 50]	124.0 [113, 135]	**367.2 [268, 487]**
AVG.	89.7 [27, 178]	52.4 [19, 101]	210.1 [81, 380]	537.8 [355, 720]	**650.9 [500, 794]**

B.1 Analysis of Planner-Likelihood Regularization

We ablate the planner-likelihood coefficient λ defined in Section 4.3 on Dog-run ($d_a = 38$) and H1hand-walk ($d_a = 61$). Figure 6 shows the reward curves, and Table 2 reports gate diagnostics at convergence.

Results. The regularizer has different effects on the two tasks. On Dog-run, CAPI with $\lambda = 0$ reaches a final return of 686.3, compared with 548.3 when $\lambda = 1$. On H1hand-walk, the variant with $\lambda = 1$ reaches 934.9, compared with 331.8 when $\lambda = 0$.

Diagnostic interpretation. Closer actor–planner alignment is not uniformly beneficial: regularization reduces d_{AP} from 0.85 to 0.013 on Dog-run and from 1.11 to 0.075 on H1hand-walk, yet final return changes in opposite directions. These results highlight the distinct roles of planner guidance and capacity allocation. Planner guidance encourages proximity to a local planner reference, whereas capacity allocation controls the relative weight assigned to each actor sample.

The fraction of samples at the minimum weight also increases, from 22% to 41% on Dog-run and from 34% to 76% on H1hand-walk. Although the regularizer does not enter the score formula, it

Table 2: **CAPI diagnostics under planner-likelihood ablation (late-training averages).** The two CAPI variants use $\lambda = 0$ and $\lambda = 1$. d_{AP} denotes the mean L2 distance between actor and planner samples, $\bar{w}$ denotes the mean gate weight, and $\rho_{\downarrow}$ denotes the fraction of samples clamped to the minimum weight.

	Dog-run ($d_a = 38$)		**H1hand-walk** ($d_a = 61$)	
	CAPI ($\lambda = 0$)	CAPI ($\lambda = 1$)	CAPI ($\lambda = 0$)	CAPI ($\lambda = 1$)
Return (final eval)	**686.3**	548.3	331.8	**934.9**
d_{AP}	0.85	0.013	1.11	0.075
$\bar{w}$	0.19	0.16	0.15	0.12
$\rho_{\downarrow}$	22%	41%	34%	**76%**

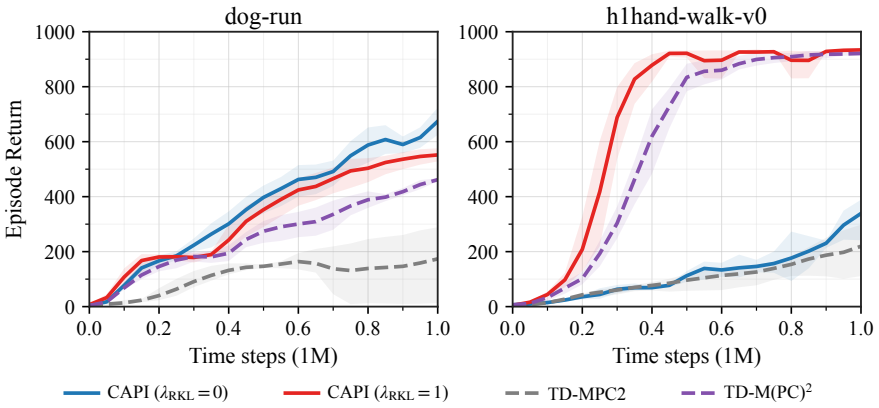


Figure 6: **Planner-likelihood regularization ablation.** The CAPI curves compare $\lambda = 0$ and $\lambda = 1$ for the optional planner-likelihood term in Section 4.3. **Left**: Dog-run ($d_a = 38$), where the term reduces return. **Right**: H1hand-walk ($d_a = 61$), where it improves return. Curves show mean returns, with shaded regions indicating 95% bootstrap confidence intervals obtained by resampling independent runs within each setting.

can change subsequent actor proposals and their scores, thereby affecting allocation under the same weighting rule. Since each weight multiplies both the base actor loss and the planner-likelihood term, a small weight attenuates both contributions. Consequently, closer actor–planner alignment does not determine performance by itself, as the regularizer also changes which proposals retain substantial influence on subsequent updates.

B.2 Capacity Threshold Computation

We solve for b_κ separately at each imagined rollout depth using the corresponding actor batch. Before root finding, scores are clipped at the batchwise 98th percentile to limit the effect of extreme values on the threshold. Writing $\widetilde{s}_i = \min\{s_i, q_{0.98}(s)\}$, the implemented normalization condition is

$$F(b) = \frac{1}{n}\sum_{i=1}^{n} \exp(\min\{\widetilde{s}_i - b, \log\kappa\}) - 1 = 0. \tag{14}$$

This is equation 9 evaluated on the clipped scores. For $\kappa > 1$, F is continuous and non-increasing, with a unique root in $[\min_i \widetilde{s}_i - \log\kappa,\ \max_i \widetilde{s}_i]$. Algorithm 2 performs 25 bisection iterations and returns the midpoint of the final bracket. Each iteration costs $O(n)$ and halves the bracket width.

Algorithm 2 Bisection for the capacity threshold

Require: Detached batch scores $s_1, \dots, s_n$, capacity ratio $\kappa > 1$
$q \leftarrow \mathrm{Quantile}_{0.98}(s_1, \dots, s_n)$
$\widetilde{s}_i \leftarrow \min\{s_i, q\}$ for $i = 1, \dots, n$
$b_{\mathrm{lo}} \leftarrow \min_i \widetilde{s}_i - \log \kappa$, $b_{\mathrm{hi}} \leftarrow \max_i \widetilde{s}_i$
for $j = 1, \dots, 25$ **do**
  $b_{\mathrm{mid}} \leftarrow (b_{\mathrm{lo}} + b_{\mathrm{hi}})/2$
  $f \leftarrow n^{-1} \sum_i \exp(\min\{\widetilde{s}_i - b_{\mathrm{mid}}, \log \kappa\}) - 1$
  if $f > 0$ **then**
    $b_{\mathrm{lo}} \leftarrow b_{\mathrm{mid}}$
  else
    $b_{\mathrm{hi}} \leftarrow b_{\mathrm{mid}}$
  end if
end for
return $(b_{\mathrm{lo}} + b_{\mathrm{hi}})/2$

Score clipping is used only for threshold computation. The original s_i enters the weight formula in equation 12. Since $\widetilde{s}_i \leq s_i$, clipping cannot increase the exact threshold or decrease the resulting weights. The normalized-share bound therefore remains valid, up to numerical root-finding accuracy.

B.3 Hyperparameters

Unless otherwise stated, we use the standard TD-MPC2 hyperparameters listed in Table 3.

B.4 Baseline Details

We benchmark against representative continuous-control algorithms using their official open-source implementations and default hyperparameters.

SAC. SAC is a maximum-entropy off-policy model-free algorithm. Code and results on HumanoidBench were obtained from `https://github.com/carlosferrazza/humanoid-bench`, with DMControl results from `https://github.com/nicklashansen/tdmpc2`.

DreamerV3. DreamerV3 is a model-based agent that trains its policy entirely through latent imagination, providing a non-MPC comparison. Sourced from `https://github.com/danijar/dreamerv3`.

TD-MPC2. TD-MPC2 combines latent dynamics learning with online MPPI planning and serves as the base algorithm for CAPI. Code and results were obtained from `https://github.com/nicklashansen/tdmpc2`.

TD-M(PC)^2. TD-M(PC)^2 is a recent TD-MPC variant that introduces policy constraints to mitigate value overestimation. Sourced from `https://github.com/DarthUtopian/tdmpc_square_public`.

B.5 Sample Efficiency

To complement the per-task learning curves, we quantify sample efficiency using **normalized AUC**. For each seed, the episode return is integrated over $[0, 1{\times}10^6]$ steps via trapezoidal quadrature and normalized by the maximum possible area ($10^6 \times 1000$), yielding a value in $[0, 1]$. A larger normalized AUC indicates greater cumulative return over the training horizon.

B.6 Qualitative Results

To provide an intuitive view of the behaviors learned by CAPI, we present qualitative examples of the trained policies. Figure 8 shows action sequences from successful rollouts across several challenging continuous-control tasks. These examples illustrate stable and coordinated locomotion behaviors, including obstacle avoidance in HumanoidBench.

Table 3: Hyperparameters

	Hyperparameter	Value
Training	Optimizer	Adam
	Learning rate	3×10^{-4}
	Encoder learning rate	1×10^{-4}
	Batch size	256
	Environment steps	1×10^{6}
	Buffer size	1×10^{6}
	Discount rate (γ)	0.99
	Gradient clipping norm	20
Planner	Horizon (H)	3
	MPPI iterations	6
	Number of Candidate trajectories	512
	Number of Elite trajectories	64
	Number of Policy-prior trajectories	24
Actor	Minimum log standard deviation	-10
	Maximum log standard deviation	2
	Entropy coefficient (α_{ent})	1×10^{-4}
	Pessimism coefficient (β_{pess})	**10.0**
	Log-clip range ($\ell_{\max}$)	**3.0**
	Capacity ratio (κ)	**8**
	Weight temperature (T_w)	**5.0**
	Planner-likelihood coefficient (λ)	**0 (DMControl)** **1 (HumanoidBench)**
Losses	Consistency loss coefficient (c_d)	20
	Reward loss coefficient (c_r)	0.1
	Value loss coefficient (c_q)	0.1
Architecture (5M)	Encoder layers	2
	Encoder dimension	256
	MLP hidden dimension	512
	Latent space dimension	512
	Task embedding dimension	96
	Critic ensemble size	5
	Distributional value bins	101
	Dropout rate	0.01
	MLP activation	Mish
	MLP normalization	LayerNorm

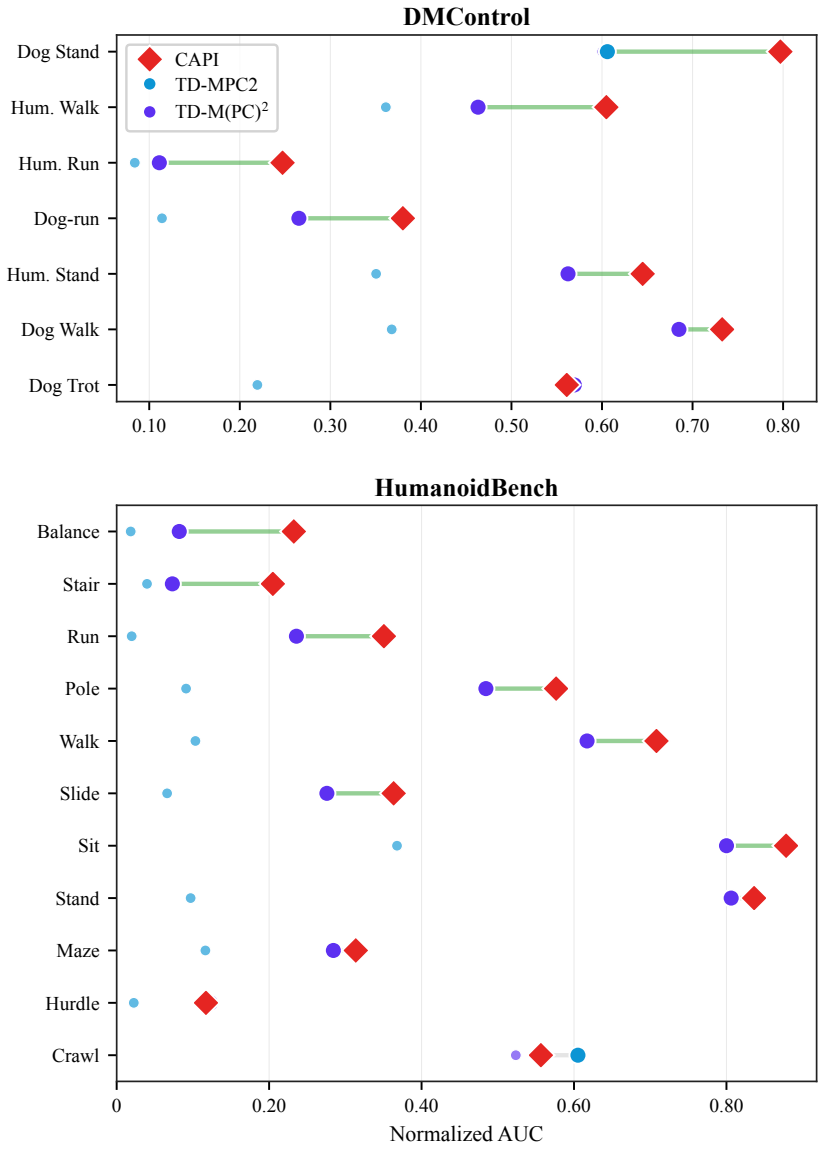


Figure 7: **Sample efficiency across all 18 tasks.** Normalized AUC scores are shown for the DM-Control (top) and HumanoidBench (bottom) benchmarks. Line segments connect the performance of the best baseline to CAPI. Green segments indicate an improvement by CAPI, whereas gray segments denote a baseline advantage.

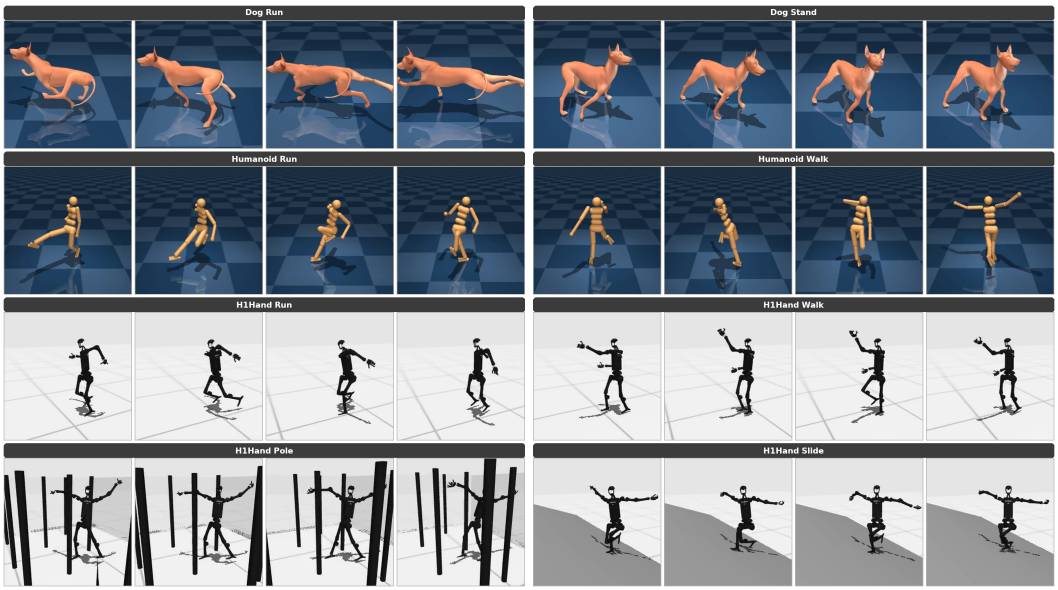


Figure 8: **Qualitative rollouts of CAPI policies.** The sequences show selected successful rollouts for quadrupedal and bipedal agents in the DeepMind Control Suite and HumanoidBench. The shown tasks are DMControl Dog-run, Dog-stand, Humanoid-run, and Humanoid-walk, and HumanoidBench H1hand-run, H1hand-walk, H1hand-pole, and H1hand-slide.