CAST: Alternating State-Value Targets and Expanded Policy Gradients for Model-Based Reinforcement Learning

Pietro Noah Crestaz[1,2], Mohamed Yassine Kabouri[2,3], Nicolas Mansard[2,4], and Andrea Del Prete[1]

***Abstract*—Model-based reinforcement learning (MBRL) is a family of RL methods that learn a model of the environment and use it for action selection, making it well suited to robotics due to its sample efficiency. Combining learned models with online planning can further improve action selection, as the planner can exploit the model to find better actions than the learned policy alone. Recent methods combining learned policies with online planning typically learn the value of the policy rather than the stronger planner-guided behavior. We present CAST (Critic with Alternating State-value Target), which uses planner-guided behavior to improve value learning while regularizing the value estimate with the current policy. CAST replaces the action-value critic with a state-value critic, trained using a target that combines a real planner-guided transition and an imagined transition under the current policy. The resulting value function corresponds to an alternating process between planner-guided behavior and the current policy, allowing it to benefit from the stronger planner behavior while being regularised by the policy being learned. We evaluate CAST on the DeepMind Control and HumanoidBench Suites against several state-of-the-art methods, and demonstrate successful transfer to a physical Unitree Go2 quadruped performing a dynamic handstand.**

I. Introduction

Reinforcement Learning (RL) has emerged as a powerful framework for learning complex robotic behaviors directly through interaction with the environment [1]. It has enabled learning skills such as dexterous manipulation [2] and legged locomotion [3]. However, applying RL in the real world remains challenging due to its high sample complexity. Model-based reinforcement learning (MBRL) addresses this limitation by learning a model of the environment and using it for planning and policy optimization, reducing the number of real-world interactions required. Planning-based MBRL performs online trajectory optimization on the learned model, evaluating and refining candidate action sequences before execution, providing a mechanism for improving action selection beyond the learned policy alone [4], [5].

Recent methods combine online planning with an explicitly learned policy. MOPAC [6] learns both a Q_{π_θ} and a V_{π_θ}, using MPPI rollouts to augment the training data for a model-free actor-critic policy. LOOP [7] builds a planner on top of SAC [8], while TD-MPC [9] and TD-MPC2 [10] jointly learn a latent world model, a policy, and an action-value critic $Q(z, a)$ through temporal-difference (TD) learning, with the critic also providing the terminal value for planning. At runtime, the planner is used in a MPC setting and acts as a refinement β of the learned policy π_θ.

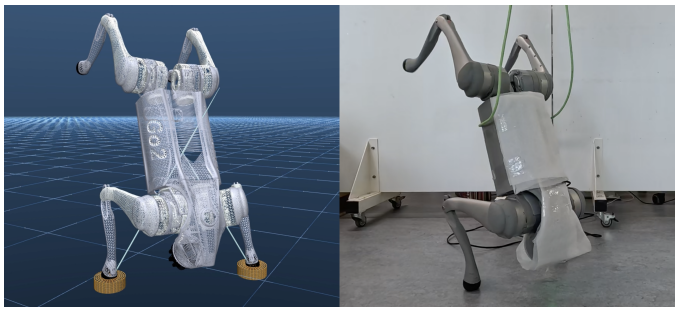

Fig. 1. **CAST sim-to-real transfer** on the Unitree Go2 quadruped performing a dynamic handstand. MuJoCo simulation (left) and real-robot execution (right).

Recent methods train the critic to learn the value of the optimized policy π_θ. They differ mainly in how they close the gap between π_θ and the planner-augmented policy β to address the well-known issue of distribution mismatch. BOOM [11] introduces a likelihood-free alignment loss that pulls the policy toward the planner's non-parametric action distribution. BMPC [12] instead replaces critic-based policy improvement with behavior cloning (BC), using a state-value critic learned on-policy. Both methods therefore learn the value of π_θ, and only differ in how they push π_θ closer to β.

We start instead from a simpler observation. The behavior policy β (the planner) generally outperforms the learned policy π_θ [10], [12], especially in the early training stage. We argue that the critic should therefore learn the value of β rather than the value of π_θ, so that policy improvement can draw on a better critic. However, since β generally outperforms π_θ by a wide margin, bootstrapping directly from β for policy improvement yields large policy updates and destabilizes training, a common failure mode in RL methods. We address this with a regularized TD-target construction that mixes one real transition under β with one imagined transition under π_θ. Under fixed policies and exact dynamics, the resulting Bellman operator has a unique fixed point corresponding to an alternating $\beta \to \pi_\theta \to \beta \to \pi_\theta$ process.

This choice also raises a second question, whether this critic should be an action-value $Q_\beta(z, a)$ or a state-value $V_\beta(z)$. Learning Q_β is impractical, since β's action distribution at a state is only available by re-running the planner, requiring a new planner query for every TD target rather than just reusing stored transitions. A state-value critic $V_\beta(z)$ avoids this, depending only on the states β visits, already recorded in

[1]Industrial Engineering Department, University of Trento, Trento, Italy.
[2]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France.
[3]Machines in Motion Laboratory, New York University, New York, USA.
[4]Artificial and Natural Intelligence Toulouse Institute (ANITI), Toulouse.
⋆Corresponding author: pietronoah.crestaz@unitn.it.
Project page: https://pietronoah.github.io/cast/

the replay buffer. MODIP [13] switches the learned policy for a diffusion policy, trained with BC over β, and trains the critic directly on planner-generated transitions, in order to avoid querying the policy during planning. The state-value formulation also changes policy optimization, since $V(z)$ takes no action as input and we cannot differentiate it directly with respect to the policy action as in the $Q(z, \pi_\theta(z))$ based objective. We instead reconstruct a one-step action-value estimate from the learned reward and dynamics models, and unroll π_θ through the model for k steps before bootstrapping with V.

We build this formulation into TD-MPC2 [10], retaining its latent world model and MPPI planner, while replacing the action-value critic with an ensemble state-value critic. We call the resulting method **CAST** (**C**ritic with **A**lternating **S**tate-value **T**arget), a novel MBRL algorithm that replaces the action-value critic with a state-value critic trained toward the value of the behavior policy β. The key contributions are:

1) A regularized hybrid Bellman target with a characterized fixed point, combining a real transition under β with an imagined transition under π_θ;
2) A k-step model-based policy-improvement objective that unrolls π_θ through the learned dynamics before bootstrapping.

CAST is validated first in simulation against five representative RL baselines on 14 high-dimensional control task, and finally on physical hardware by deploying it on a Unitree Go2 quadruped.

II. Related Work

Planning with learned world models has become a central approach in MBRL. Sampling-based methods such as PlaNet [14] and TD-MPC/TD-MPC2 [9], [10] use learned dynamics and reward models to generate predictive trajectories and bootstrap finite-horizon trajectories with learned value estimates. Other approaches, such as MuZero [4] and EfficientZero [15] combine learned latent dynamics with tree search. Among these methods, TD-MPC2 is most closely related to our setting. It combines latent world models, MPPI planning, and an action-value critic that is used both as the terminal reward for planning and as the policy improvement signal.

Our work is also related to recent methods that address the mismatch between the learned policy and the planner in planning-based RL. Replay data are generated by the planner-augmented behavior policy β, while gradient-based updates optimize the parametric policy π_θ, introducing an inherent distribution mismatch. BOOM [11] addresses this mismatch by aligning the learned policy with the planner distribution using a forward KL divergence in the actor loss. The concurrent TD-MPC^2 policy-alignment method [16] similarly regularizes the policy distribution toward planner actions, but uses a reverse KL divergence. BMPC [12] instead uses a state-value critic trained using an on-policy model-based target and a Behavior Cloning (BC) loss for the actor.

Our work also employs a state-value critic, but with a different goal. Rather than aligning π_θ with β or imitating β's actions, we aim to have the critic represent the value of β directly, since β generally outperforms π_θ. Using V_β to bootstrap the policy optimization, however, performs poorly in practice, as we show in Sec. IV-D. We therefore define a regularized hybrid Bellman target consisting of one replay transition under β followed by one imagined transition under π_θ, and characterize the value function to which this training procedure converges.

The policy optimization component of CAST is related to imagination-based RL. Dreamer [17] optimize the policy by differentiating through latent trajectories generated by learned dynamics, following the pathwise-gradient view of deterministic policy gradients [18]. DMO [19] also propagates policy gradients through a learned dynamics model, while using a separate model for high-fidelity rollouts. These methods differ from TD-MPC2 in that online planning is not the central mechanism used during environment interaction. In TD-MPC2, the learned policy mainly provides an initial distribution for MPPI, while the planner produces the final actions executed in the environment. CAST combines this planning setup with model-based policy optimization by propagating gradients through multiple imagined transitions before bootstrapping with the state-value critic.

State-value functions have also been used in settings where explicit action-value estimation is undesirable or unnecessary. Implicit Q-Learning [20] learns a state-value function through expectile regression to approximate an upper expectile of Q, avoiding explicit maximization over out-of-distribution actions in offline RL, although it still learns an action-value function. POLO [21] and other value-function-based MPC methods [22] use a learned state-value function as the terminal cost for model predictive control, which is closer to our use of $V(z)$ for planning. MODIP [13] similarly replaces a policy-dependent action-value function with a terminal state value to avoid querying a diffusion policy during planning, reducing inference time. Our setting differs in that replacing $Q(z, a)$ with $V(z)$ is not just an architectural choice, but comes from our aim to learn the value of β rather than π_θ. Learning Q of β would indeed be extremely computationally demanding, requiring to explore both state and action spaces, while querying the planner β for the construction of TD targets. Learning V is instead much cheaper, as it requires only the exploration of the state space, and we can still estimate Q exploiting our knowledge of V and the learned reward/dynamic model. CACTO [23], [24] similarly trains the critic on cost-to-go along TO solutions without querying the actor.

Finally, learned world models have also been deployed on physical robotic systems. MoDem-V2 [25] demonstrate TD-MPC-style world models for real-world manipulation, while subsequent work has explored adapting pretrained world models to hardware [26]. RT-HCP [27] addresses a closely related problem, compensating for inference delays that exceed the robot's control frequency.

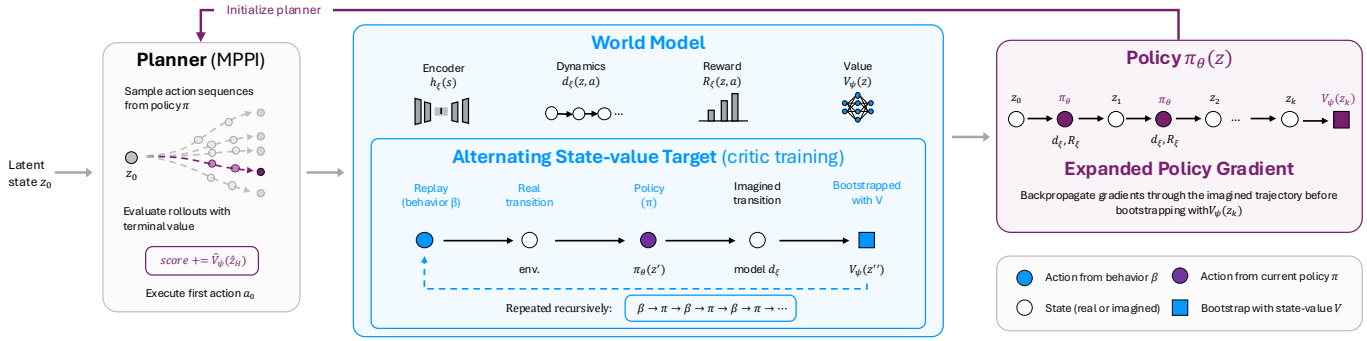

Fig. 2. **Overview of CAST**. A world model learns latent dynamics, rewards, and a state-value critic trained with **alternating state-value targets** that combine a replay transition with an imagined policy transition. The planner evaluates candidate trajectories using the terminal state value, while the policy is optimized through **expanded policy gradients** propagated across imagined rollouts.

III. Method

A. Background: TD-MPC2

We build on TD-MPC2 [10], which jointly learns a latent world model and a policy and interacts with the environment through online planning. Observations are mapped to a latent state z using an encoder $z = h_\xi(s)$, while a latent dynamics head $z' = d_\xi(z,a)$ and reward head $r = R_\xi(z,a)$ predict the next latent state and reward without decoding back to observation space. An ensemble of $N{=}5$ action-value functions $\{Q^i_\psi(z,a)\}^N_{i=1}$ and a stochastic maximum-entropy Gaussian policy $\pi_\theta(a \mid z)$ complete the model. All components are MLPs with LayerNorm and Mish activations. The latent state is projected onto a product of simplices using SimNorm [28] to keep its scale bounded and sparse. Reward and value targets are represented as soft cross-entropy distributions over symlog-spaced bins [29] rather than raw scalars, making training less sensitive to the scale of the task reward. A target ensemble $\bar{Q}_\psi$ is maintained using Polyak averaging, $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$, and TD targets use the minimum of two randomly sampled ensemble members. Given a H-step trajectory $(s_t, a_t, r_t, s_{t+1})^H_{t=0}$ sampled from the replay buffer, the encoder, dynamics, reward, and value heads are trained jointly by minimizing

$$\begin{aligned}\mathcal{L}_{\text{model}} = \mathbb{E}\Big[&\textstyle\sum_{t=0}^{H} \rho^t \big(|d_\xi(z_t,a_t) - \text{sg}(h_\xi(s_{t+1}))|^2_2 \\ &+ \text{CE}(R_\xi(z_t,a_t), r_t) + \text{CE}(Q_\psi(z_t,a_t), y_t)\big)\Big],\end{aligned} \tag{1}$$

where sg denotes the stop-gradient operator, $\rho \in (0,1]$ discounts errors further along the rollout, and CE is cross-entropy against a two-hot target. The one-step TD target for the action-value function is

$$y_t = r_t + \gamma, \bar{Q}_\psi\big(z_{t+1}, \tilde{a}_{t+1}\big), \qquad \tilde{a}_{t+1} \sim \pi_\theta(\cdot \mid z_{t+1}). \tag{2}$$

This implies that TD-MPC2 learns the value of the neural policy π_θ, as this is the policy used to sample a_{t+1}. However, the transitions in the replay buffer are not generated by π_θ. At each control step, actions are selected using the Model Predictive Path Integral (MPPI) planner [30]. The planner samples horizon-H action sequences from a diagonal Gaussian $\mathcal{N}(\mu, \sigma^2)$, rolls them out through d_ξ, and scores each sequence using

$$\hat{G}(z_t, a_{t:t+H}) = \sum_{h=0}^{H-1} \gamma^h R_\xi(z_{t+h}, a_{t+h}) + \gamma^H \bar{Q}_\psi\big(z_{t+H}, a_{t+H}\big), \tag{3}$$

where the terminal action $a_{t+H} \sim \pi_\theta(\cdot \mid z_{t+H})$ is sampled from the policy. A fraction of the sampled sequences are initialized from the policy prior, so π_θ contributes a subset of the candidate rollouts at each planning step. An elite set is selected from the rollout batch, and the mean of the sampling distribution is updated using a softmax over the elite scores. This procedure is repeated for a fixed number of iterations, after which the planner executes the first action of the resulting sequence. The policy prior is trained separately using a maximum-entropy objective similar to standard Soft Actor-Critic [8],

$$\begin{aligned}\mathcal{L}_\pi(\theta) = -\mathbb{E}\Big[&\textstyle\sum_{t=0}^{H} \rho^t \big(\alpha\, \mathcal{H}(\pi_\theta(\cdot \mid z_t)) \\ &+ Q_\psi(z_t, \tilde{a}_t)\big)\Big],\end{aligned} \tag{4}$$

where $\tilde{a}_t \sim \pi_\theta(\cdot \mid z_t)$, α is a fixed entropy coefficient, and gradients are taken only with respect to θ.

B. From Action-Values to State-Values

Our method replaces the action-value critic with a state-value critic. For any stochastic policy π, the corresponding action-value and state-value functions satisfy

$$Q_\pi(z,a) = R(z,a) + \gamma V_\pi\big(d(z,a)\big), \tag{5}$$

$$V_\pi(z) = \mathbb{E}_{a\sim\pi(\cdot|z)}\big[Q_\pi(z,a)\big]. \tag{6}$$

Here, the transition dynamics is deterministic, while the expectation in (6) is taken over the stochastic policy.

Following the value averaging of TDMPC2 [10], we similarly learn an ensemble of state-value functions $\{V^i_\psi(z)\}^N_{i=1}$ with the same architecture as the original TD-MPC2 critic. A target ensemble $\{\bar{V}^i_\psi\}^N_{i=1}$ is maintained using the same Polyak averaging rate τ. Whenever an action-value estimate is needed, we reconstruct a one-step estimate from the learned reward, dynamics models and target state-value function,

$$\hat{Q}(z,a) = R_\xi(z,a) + \gamma \bar{V}_\psi\big(d_\xi(z,a)\big). \tag{7}$$

This composed estimate is the starting point to define the value-learning target and to construct the policy objective in Sec. III-E.

Using a state-value critic also changes the terminal objective used by the planner. Rather than sampling a terminal action and evaluating $\bar{Q}_\psi(z_{t+H}, a_{t+H})$, CAST uses the state-value estimate directly,

$$\hat{G}(z_t, a_{t:t+H}) = \sum_{h=0}^{H-1} \gamma^h R_\xi(z_{t+h}, a_{t+h}) + \gamma^H \bar{V}_\psi\big(z_{t+H}\big). \quad (8)$$

This removes the need to sample a terminal policy action during planning.

C. A Hybrid Value Target: Mixing Behavior and Policy

The first design choice in CAST is how to train V_ψ. We aim for the critic to represent the value of β rather than of π_θ, but bootstrapping directly from the value of β performs poorly in practice because it leads to abrupt policy updates that destabilize training (see ablations in Sec. IV-D). We therefore combine one real transition generated by β with one imagined transition generated by the learned policy π_θ.

Starting from a replay transition (s_t, r_t, s_{t+1}), we train the critic against the following two-step target using the same soft cross-entropy formulation used by TD-MPC2:

$$y_t = r_t + \gamma \Big[R_\xi(z_{t+1}, \tilde{a}_{t+1}) + \gamma \bar{V}_\psi(\tilde{z}_{t+2}) \Big], \quad (9)$$

where $z_{t+1} = h_\xi(s_{t+1})$, $\tilde{a}_{t+1} \sim \pi_\theta(\cdot \mid z_{t+1})$, and $\tilde{z}_{t+2} = d_\xi(z_{t+1}, \tilde{a}_{t+1})$. The first step is the real transition stored in the replay buffer, while the second step is generated under the current policy using the learned model. The value function therefore corresponds to the alternating policy sequence

$$\beta \to \pi_\theta \to \beta \to \pi_\theta \to \cdots. \quad (10)$$

The critic is neither the value function of β nor that of π_θ alone. Instead, it is the value function induced by their alternating composition.

This construction regularizes the critic toward V_β without collapsing onto it entirely. The replay transition keeps the target anchored to the stronger behavior policy β, while the imagined transition reintroduces π_θ every other step, preventing the large, destabilizing policy updates that a critic trained purely on β would otherwise produce.

D. Theoretical Analysis of Hybrid Value Targets

We next characterize formally the value function represented by this target. Assume that β and π_θ are fixed, the environment and learned dynamics are deterministic, and the value function can be represented without approximation error. Let $\beta(a \mid z)$ denote the stochastic behavior policy, which is stochastic due to the stochastic nature of the planner, and $\pi_\theta(a \mid z)$ the stochastic learned policy. The corresponding one-step Bellman operators are

$$(\mathcal{T}_\beta f)(z) = \mathbb{E}_{a\sim\beta(\cdot|z)} \left[R(z,a) + \gamma f\big(d(z,a)\big) \right], \quad (11)$$

$$(\mathcal{T}_\pi f)(z) = \mathbb{E}_{a\sim\pi_\theta(\cdot|z)} \left[R(z,a) + \gamma f\big(d(z,a)\big) \right]. \quad (12)$$

Although the dynamics are deterministic, both operators remain stochastic through their respective action distributions.

The target in (9) corresponds to the composition

$$\mathcal{T}_{\text{CAST}} = \mathcal{T}_\beta \mathcal{T}_\pi. \quad (13)$$

Proposition 1. *Assume that β and π_θ are fixed, the environment and learned dynamics are deterministic, and the value function can be represented without approximation error. Then $\mathcal{T}_{\text{CAST}}$ is a γ^2-contraction under the supremum norm and admits a unique fixed point V_{CAST} satisfying*

$$V_{CAST} = \mathcal{T}_\beta \mathcal{T}_\pi V_{CAST}. \quad (14)$$

Proof. Expanding the composition gives

$$\begin{aligned}(\mathcal{T}_\beta \mathcal{T}_\pi V)(z_0) = \mathbb{E}_{a_0\sim\beta(\cdot|z)} \Big[R(z_0, a_0) \\ + \gamma\, \mathbb{E}_{a_1\sim\pi_\theta(\cdot|z_1)} \big[R(z_1, a_1) + \gamma V(z_2) \big] \Big],\end{aligned} \quad (15)$$

where $z_1 = d(z_0, a_0)$ and $z_2 = d(z_1, a_1)$. Thus, one application of the CAST operator consists of one transition under the behavior policy followed by one transition under the current policy.

Both $\mathcal{T}_\beta$ and $\mathcal{T}_\pi$ are individually γ-contractions under the supremum norm [1]. Their composition is therefore a γ^2-contraction:

$$\begin{aligned}\left\| \mathcal{T}_\beta \mathcal{T}_\pi V - \mathcal{T}_\beta \mathcal{T}_\pi U \right\|_\infty &\le \gamma \left\| \mathcal{T}_\pi V - \mathcal{T}_\pi U \right\|_\infty \\ &\le \gamma^2 \left\| V - U \right\|_\infty.\end{aligned} \quad (16)$$

By the Banach fixed-point theorem, $\mathcal{T}_{\text{CAST}}$ therefore admits a unique fixed point V_{CAST} satisfying (14). □

Repeated application of the operator gives

$$V_{CAST} = (\mathcal{T}_\beta \mathcal{T}_\pi)^m V_{CAST}, \qquad m \ge 1, \quad (17)$$

and expanding the resulting sequence yields

$$V_{CAST}(z) = \mathbb{E}\left[r_0^\beta + \gamma r_1^\pi + \gamma^2 r_2^\beta + \gamma^3 r_3^\pi + \cdots \right], \quad (18)$$

where the expectation is taken over the stochastic actions sampled from β and π_θ. Eq. (18) shows that V_{CAST} discounts rewards under β and π_θ in alternation, placing it between V_β and V_{π_θ} rather than coinciding with either.

E. Expanded Policy Gradient Through the World Model

Policy improvement follows the same idea as Sec. III-B, so wherever the base method would query $Q_\psi(z, \pi_\theta(z))$, we substitute the composed estimator $\hat{Q}$ of (7), evaluated at the policy action:

$$\hat{J}_1(z;\theta) = R_\xi\big(z, \pi_\theta(z)\big) + \gamma\, \bar{V}_\psi\big(d_\xi(z, \pi_\theta(z))\big). \quad (19)$$

This is a one-step Bellman backup taken directly through the differentiable reward and dynamics heads. Recursively substituting (5)–(6) into themselves H_π times, with $a^{(i)} \sim \pi(\cdot \mid z^{(i)})$ and $z^{(i+1)} = d(z^{(i)}, a^{(i)})$ for $i = 0, \ldots, H_\pi - 1$, gives the standard k-step Bellman identity for the value function V_π, for any $H_\pi \ge 1$:

$$V_\pi(z) = \mathbb{E}\left[\sum_{i=0}^{H_\pi - 1} \gamma^i R\big(z^{(i)}, a^{(i)}\big) + \gamma^{H_\pi} V_\pi\big(z^{(H_\pi)}\big) \right]. \quad (20)$$

Algorithm 1 CAST: one training update. Marks changes relative to [10] (Sec. III-A).

Input: encoder h_ξ, dynamics d_ξ, reward R_ξ, value ensemble V_ψ & target $\bar{V}_\psi$, policy π_θ; replay buffer $\mathcal{D}$, policy optimization horizon H_π

Plan and collect

$z_t \leftarrow h_\xi(s_t)$

$a_t \leftarrow \mathrm{MPPI}\big(z_t;\, \hat{G}(z_t, a_{t:t+H})\big) = \sum_{h=0}^{H-1} \gamma^h R_\xi(z_{t+h}, a_{t+h}) + \gamma^H \bar{V}_\psi(z_{t+H})\big)$ ▷ Eq. (8)

$s_{t+1}, r_t \leftarrow env.step(s_t, a_t)$

store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$

Sample and build the hybrid value target

$\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{H} \sim \mathcal{D}$

$z_t \leftarrow h_\xi(s_t), \quad z_{t+1} \leftarrow h_\xi(s_{t+1}), \quad \tilde{a}_{t+1} \leftarrow \pi_\theta(z_{t+1})$

$y_t \leftarrow r_t + \gamma\big[R_\xi(z_{t+1}, \tilde{a}_{t+1}) + \gamma\, \bar{V}_\psi(d_\xi(z_{t+1}, \tilde{a}_{t+1}))\big]$

Update world model and critic

$\mathcal{L} \leftarrow \mathcal{L}_{\text{consist}} + \mathcal{L}_{\text{reward}} + \mathcal{L}_{\text{value}}(V_\psi, y)$

update $h_\xi, d_\xi, R_\xi, V_\psi$ with $\nabla\mathcal{L}$ ▷ single optimizer, π_θ excluded

Update policy: 1-step expanded gradient

for $t = 0, \ldots, H_\pi$ **do**

  $a \leftarrow \pi_\theta(z_t)$

  $J(z_t;\theta) \leftarrow R_\xi(z_t, a) + \gamma\, \bar{V}_\psi(d_\xi(z_t, a))$ ▷ Eq. (19)

end for

$\mathcal{L}_\pi \leftarrow -\sum_{t=0}^{H} \rho^t \big(\alpha\, \mathcal{H}(\pi_\theta(\cdot|z_t)) + J(z_t;\theta)\big)$

update θ with $\nabla_\theta \mathcal{L}_\pi$

Update target network

$\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$

We use the corresponding learned-model estimator by replacing the true dynamics and reward with d_ξ and R_ξ, the policy with π_θ, and the terminal value with the frozen target critic $\bar{V}_\psi$, while evaluating a single sampled trajectory,

$$J_k(z^0;\theta) \;=\; \sum_{i=0}^{H_\pi - 1} \gamma^i\, R_\xi\big(z^{(i)}, a^{(i)}\big) + \gamma^{H_\pi}\, \bar{V}_\psi\big(z^{(H_\pi)}\big). \quad (21)$$

Using the reparameterized policy, gradients flow through the full imagined chain $\pi_\theta \to R_\xi \to d_\xi \to \cdots \to \bar{V}_\psi$. The world model is thus seen as a fixed, differentiable simulator for policy improvement. We treat $H_\pi{=}1$ as our primary configuration throughout, and study $H_\pi{>}1$ as an ablation in Sec. IV.

F. Training Algorithm

Algorithm 1 summarizes one update of CAST. Data collection is unchanged, the planner of (8), seeded by the policy, selects environment actions following an MPC scheme: only the first action is applied to the real environment. The only changes are the critic's input, and the value target and policy objective that follow from it, both derived above.

IV. Experiments

We evaluate CAST on a diverse set of high-dimensional continuous-control tasks with the goal of answering three questions. (i) Does replacing the action-value critic with alternating state-value targets improve sample efficiency? (ii) Does this improvement preserve strong asymptotic performance? (iii) What is the contribution of each component of CAST to the overall improvement?

A. Experimental Setup

We compare CAST against five representative RL algorithms spanning both model-free and model-based approaches. SAC [8] provides a strong model-free baseline, while DreamerV3 [31] represents latent imagination-based RL. Our primary comparisons, however, are against the recent family of planning-based methods: TD-MPC2 [10], BMPC [12], and BOOM [11]. These methods share the same latent world model and planning framework, making them the most relevant baselines for evaluating the proposed critic formulation.

Experiments are conducted on the benchmark introduced by BOOM [11], comprising seven DeepMind Control Suite [32] tasks and seven HumanoidBench [33] locomotion tasks. These domains feature large state and action spaces, making both planning and value estimation more challenging than in standard low-dimensional benchmarks.

Unless otherwise stated, CAST employs the alternating state-value target introduced in Section III-B together with a one-step expanded policy gradient ($H_\pi = 1$). All methods are evaluated using three random seeds, and shaded regions denote 95% confidence intervals, computed as mean$\pm$ 1.96 std-dev$/\sqrt{n}$, where n is the number of seeds. Code will be made publicly available upon acceptance.

B. Sample efficiency

The primary objective of CAST is to improve sample efficiency by learning a critic closer to the value of β, without the instability of bootstrapping from β alone. We therefore begin by comparing the learning dynamics of all methods throughout training.

Fig. 3 reports learning curves for all fourteen benchmark environments, together with the average return across tasks shown in the upper-left panel. CAST matches or exceeds the performance of the selected baselines on the majority of environments and consistently attains high returns earlier during training. These improvements are most evident on several HumanoidBench tasks, including `H1hand-run`, `H1hand-pole`, and `Dog-run`, where CAST reaches strong performance with fewer environment interactions than BMPC and BOOM. Similar trends are observed across the DeepMind Control Suite, indicating that the proposed critic formulation improves learning efficiency across domains with different dynamics and action dimensionalities.

Fig. 4 reports a performance profile of AUC scores (Area Under the Curve): for each task, the area under a method's mean return curve from 0 to 2M steps, divided by the number of training steps and normalized by the best AUC any method

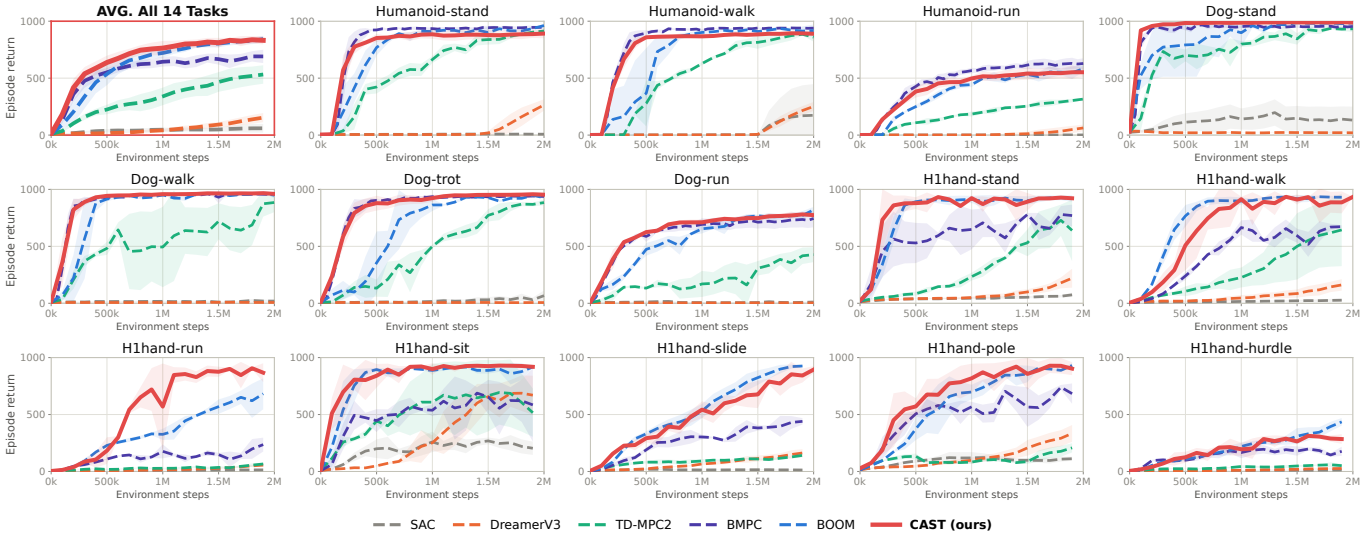

Fig. 3. Episode return vs. environment steps for our method against BOOM [11] and BMPC [12] (with SAC [8], DreamerV3 [31], and TD-MPC2 [10] also shown), on the 7 DMControl and 7 HumanoidBench tasks of Sec. IV-A. Top-Left panel: average return across the 14 tasks.

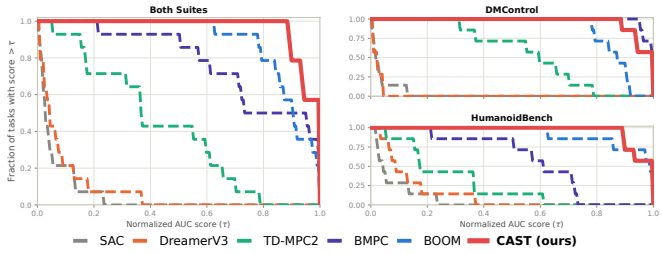

Fig. 4. **Performance profile of AUC-to-2M scores**, normalized per task by the best AUC any method reaches on that task: pooled across all 14 tasks (left), and per suite (top right: DMControl, bottom right: HumanoidBench). Each curve shows the fraction of tasks on which a method's normalized score exceeds τ, for all five baselines and our method.

reaches on that task, then pooled across all 14 tasks and per suite. For a threshold τ swept from 0 to 1, each curve plots the fraction of tasks on which a method's normalized score exceeds τ. On DMControl, our curve and BMPC's both remain at 1.0 up to $\tau \approx 0.9$; BOOM's curve falls off earlier. On HumanoidBench, our curve remains at 1.0 up to $\tau \approx 0.85$; BMPC's falls off earliest among the three methods. In average, CAST consistently achieves the best AUC performance profile among all the baselines, confirming that the improvements observed in the learning curves translate into faster policy learning.

C. Final Performance Comparison

Table I reports the average episode return after one and two million environment interactions. At 1M interactions, CAST achieves the highest average performance across the benchmark, outperforming both BMPC and BOOM while obtaining the best results on several HumanoidBench environments. These results are consistent with the learning curves presented in the previous section and indicate that the improved learning dynamics of CAST translate into stronger intermediate performance.

After 2M interactions, all planning-based methods approach convergence and the performance gap correspondingly decreases. Nevertheless, CAST remains competitive across the benchmark, outperforming BMPC and achieving performance comparable to BOOM.

D. Ablation Study

We isolate the contribution of each design choice in Secs. III-B–III-E. All ablations use the same architecture, planner, and training budget as the main comparison, varying only the axis under study.

a) Hybrid value target: We compare our hybrid target against two simpler alternatives: bootstrapping purely off-policy from the behavior policy β buffer,

$$y_t^{\text{off}} = r_t + \gamma \bar{V}_\psi(z_{t+1}), \tag{22}$$

and bootstrapping purely on-policy by resampling the action under π_θ at the real state z_t, following the target used in BMPC [12],

$$y_t^{\text{on}} = R_\xi(z_t, \tilde{a}_t) + \gamma \bar{V}_\psi\big(d_\xi(z_t, \tilde{a}_t)\big), \quad \tilde{a}_t \sim \pi_\theta(\cdot \mid z_t). \tag{23}$$

Fig. 5 reports this comparison on Dog-run and Humanoid-run, holding $H_\pi = 1$ fixed. On both tasks, the purely off-policy variant is consistently the worst of the three throughout training, consistent with the mechanism described in Sec. I: bootstrapping directly from V_β yields abrupt policy updates that destabilize learning. The purely on-policy variant is more competitive but still lags behind.

b) Behavior-to-policy ratio: We further vary the number of real transitions under β preceding the single imagined step under π_θ, denoted $H_V \in \{1, 2, 3\}$, with $H_V = 1$ recovering (9):

$$y_t^{(H_V)} = \sum_{i=0}^{H_V-1} \gamma^i r_{t+i} + \gamma^{H_V} \Big[R_\xi(z_{t+H_V}, \tilde{a}_{t+H_V}) + \gamma \bar{V}_\psi(\tilde{z}_{t+H_V+1}) \Big], \quad \tilde{a}_{t+H_V} \sim \pi_\theta(\cdot \mid z_{t+H_V}). \tag{24}$$

TABLE I

EPISODE RETURN AFTER 1M AND 2M ENVIRONMENT INTERACTIONS (MEAN ±95% CI OVER SEEDS). BEST RESULTS ARE SHOWN IN **BOLD**; SECOND-BEST ARE UNDERLINED.

		1M Environment steps				2M Environment steps			
	Task	TD-MPC2	BMPC	BOOM	**CAST**	TD-MPC2	BMPC	BOOM	**CAST**
DMControl	Humanoid-stand	664 ± 30	**935 ± 10**	910 ± 19	885 ± 40	913 ± 17	946 ± 17	**962 ± 12**	891 ± 41
	Humanoid-walk	628 ± 15	**917 ± 12**	901 ± 22	867 ± 20	860 ± 40	**939 ± 4**	918 ± 53	890 ± 13
	Humanoid-run	185 ± 22	**566 ± 24**	443 ± 21	497 ± 23	316 ± 10	**629 ± 46**	583 ± 30	552 ± 39
	Dog-stand	799 ± 26	953 ± 11	898 ± 124	**987 ± 4**	933 ± 28	954 ± 19	985 ± 3	**988 ± 5**
	Dog-walk	494 ± 352	952 ± 15	952 ± 5	**958 ± 16**	885 ± 85	**962 ± 11**	949 ± 27	961 ± 3
	Dog-trot	500 ± 15	**936 ± 10**	863 ± 53	923 ± 32	884 ± 25	939 ± 13	927 ± 9	**953 ± 15**
	Dog-run	170 ± 114	693 ± 54	654 ± 41	**711 ± 25**	427 ± 66	745 ± 78	**821 ± 26**	774 ± 25
HumanoidBench	H1hand-stand	234 ± 34	648 ± 175	912 ± 32	**923 ± 11**	640 ± 270	770 ± 96	920 ± 28	**922 ± 7**
	H1hand-walk	226 ± 88	665 ± 63	898 ± 33	**914 ± 22**	644 ± 318	673 ± 12	931 ± 12	**934 ± 2**
	H1hand-run	28 ± 4	177 ± 62	326 ± 49	**570 ± 379**	66 ± 9	236 ± 61	682 ± 136	**867 ± 83**
	H1hand-sit	609 ± 336	538 ± 52	888 ± 16	**899 ± 51**	516 ± 212	585 ± 322	918 ± 5	**919 ± 15**
	H1hand-slide	92 ± 17	303 ± 20	531 ± 63	**543 ± 73**	141 ± 18	440 ± 29	**926 ± 9**	895 ± 31
	H1hand-pole	83 ± 15	568 ± 43	698 ± 64	**818 ± 128**	208 ± 40	683 ± 90	**930 ± 21**	903 ± 57
	H1hand-hurdle	44 ± 13	166 ± 50	**220 ± 33**	195 ± 58	51 ± 14	177 ± 40	**436 ± 34**	296 ± 66
Average (14 Tasks)		340 ± 77	644 ± 43	721 ± 41	**764 ± 63**	535 ± 82	691 ± 60	**849 ± 29**	839 ± 29

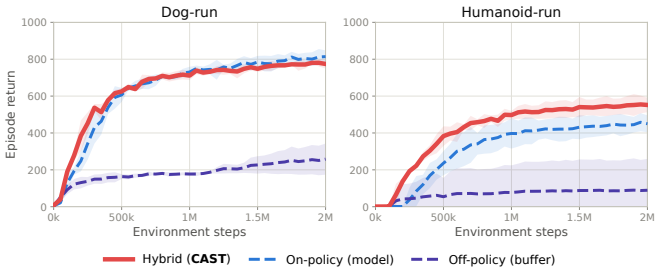

Fig. 5. **Value-target ablation** on Dog-run (left) and Humanoid-run (right): our hybrid target vs. purely on-policy and purely off-policy alternatives, $H_\pi = 1$ throughout. Mean ± 95% CI over three seeds.

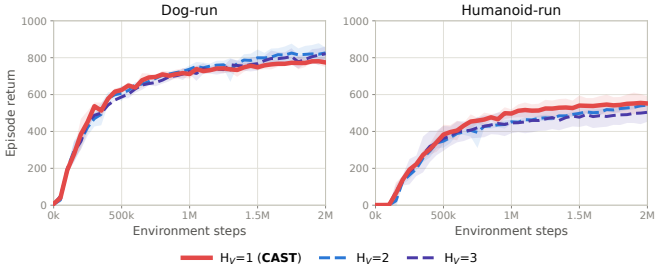

Fig. 6. **Behavior-to-policy ratio ablation** on Dog-run (left) and Humanoid-run (right): $H_V = 1$ (**CAST**) vs. $H_V = 2$ and $H_V = 3$, i.e. anchoring the value target to more real transitions under β before the single imagined step under π_θ. Mean ± 95% CI over three seeds.

Fig. 6 reports this comparison on Dog-run and Humanoid-run. On Dog-run, $H_V = 2$ and $H_V = 3$ remain competitive with $H_V = 1$, while on Humanoid-run both alternatives lag behind throughout training, indicating that anchoring the target to a single real transition before reintroducing π_θ is a well balanced choice.

c) Expanded gradient horizon: We vary $k = H_\pi \in \{1, 2, 3\}$, with k=1 as the reference configuration used everywhere else in this paper, holding the hybrid value target of Eq. (9) fixed. This isolates the effect of unrolling the policy gradient further through the learned model before

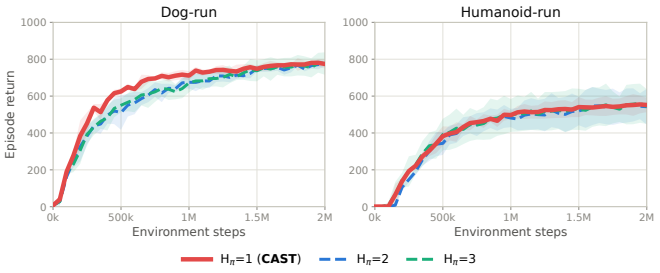

Fig. 7. **Expanded-gradient-horizon ablation** on Dog-run (left) and Humanoid-run (right): $H_\pi = 1$ (ours) vs. $H_\pi = 2$ and $H_\pi = 3$, hybrid value target fixed throughout. Mean ± 95% CI over three seeds.

bootstrapping. Fig. 7 reports this comparison on the same two tasks. On both, k=2 and k=3 track each other almost exactly throughout, and track k=1 closely as well, with k=1 holding a small lead on both tasks and with smaller CI.

E. Real-World Deployment

To demonstrate the practical applicability of CAST, we deploy a policy trained entirely in simulation on a Unitree Go2 quadruped. The task consists of performing a dynamic handstand by lifting the rear legs and balancing only on the front legs. Training combines domain randomization over the robot's dynamics with a curriculum that ramps up randomization strength as the policy's stand success rate improves, and the trained policy π_θ is deployed directly on hardware without online planning.

Figure 8 shows the deployment setup and representative snapshots of the learned behavior. The policy successfully transfers from simulation to hardware and executes stable handstand motions in real time. While this experiment is intended as a proof of deployment rather than a comparative evaluation, it demonstrates that CAST produces policies robust enough for direct real-world transfer.

Fig. 8. **Hardware deployment.** Snapshots of the learned policy π_θ executing a dynamic handstand on the Unitree Go2. See companion video.

V. Conclusion

We presented CAST, a planning-based model-based RL framework that replaces the action-value critic with a state-value formulation trained toward the value of the behavior policy β. Since bootstrapping directly from β performs poorly in practice, we introduced a regularized hybrid Bellman target that mixes real transitions under β with imagined transitions under the current policy obtained by k-steps rollout with the learned dynamics, with a characterized fixed point. Experiments on high-dimensional continuous-control tasks show strong sample efficiency and competitive final performance against established model-free and model-based baselines. We further demonstrated CAST on a Unitree Go2 quadruped using asynchronous predictive MPC, showing that planning latency can be hidden from the control loop without additional supervision.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018.

[2] O. M. Andrychowicz, B. Baker *et al.*, "Learning dexterous in-hand manipulation," *Int. J. Rob. Res. (IJRR)*, vol. 39, no. 1, Jan. 2020.

[3] N. Rudin, D. Hoeller *et al.*, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2021.

[4] J. Schrittwieser, I. Antonoglou *et al.*, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, 2020.

[5] D. Silver, J. Schrittwieser *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, 2017.

[6] A. S. Morgan, D. Nandha *et al.*, "Model predictive actor-critic: Accelerating robot skill acquisition with deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2021.

[7] H. Sikchi, W. Zhou *et al.*, "Learning off-policy with online planning," in *Conference on Robot Learning (CoRL)*, 2022.

[8] T. Haarnoja, A. Zhou *et al.*, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018.

[9] N. Hansen, X. Wang *et al.*, "Temporal difference learning for model predictive control," in *International Conference on Machine Learning (ICML)*, 2022.

[10] N. Hansen, H. Su *et al.*, "TD-MPC2: Scalable, robust world models for continuous control," in *International Conference on Learning Representations (ICLR)*, 2024.

[11] G. Zhan, L. Wang *et al.*, "Bootstrap off-policy with world model," in *Neural Information Processing Systems (NeurIPS)*, 2025.

[12] Y. Wang, H. Guo *et al.*, "Bootstrapped model predictive control," in *International Conference on Learning Representations (ICLR)*, 2025.

[13] Z. el Asri, P. Gratias-Quiquandon *et al.*, "Modip: Efficient model-based optimization for diffusion policies," *arXiv preprint arXiv:2606.10825*, 2026.

[14] D. Hafner, T. Lillicrap *et al.*, "Learning latent dynamics for planning from pixels," in *International Conference on Machine Learning (ICML)*, 2019.

[15] S. Wang, S. Liu *et al.*, "Efficientzero v2: mastering discrete and continuous control with limited data," in *International Conference on Machine Learning (ICML)*, 2024.

[16] H. Lin, P. Wang *et al.*, "TD-MPC^2: Improving temporal difference MPC through policy constraint," *arXiv preprint arXiv:2502.03550*, 2025.

[17] D. Hafner, T. Lillicrap *et al.*, "Dream to control: Learning behaviors by latent imagination," in *International Conference on Learning Representations (ICLR)*, 2020.

[18] T. P. Lillicrap, J. J. Hunt *et al.*, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2016.

[19] J. Amigo, R. Khorrambakht *et al.*, "First order model-based rl through decoupled backpropagation," in *Conference on Robot Learning (CoRL)*, 2025.

[20] I. Kostrikov, A. Nair *et al.*, "Offline reinforcement learning with implicit Q-learning," in *International Conference on Learning Representations (ICLR)*, 2022.

[21] K. Lowrey, A. Rajeswaran *et al.*, "Plan online, learn offline: Efficient learning and exploration via model-based control," in *International Conference on Learning Representations (ICLR)*, 2019.

[22] P. N. Crestaz, L. De Matteis *et al.*, "Td-cd-mppi: Temporal-difference constraint-discounted model predictive path integral control," *IEEE Robotics and Automation Letters*, vol. 11, no. 1, 2026.

[23] G. Grandesso, E. Alboni *et al.*, "Cacto: Continuous actor-critic with trajectory optimization—towards global optimality," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, p. 3318–3325, 2023.

[24] E. Alboni, P. N. Crestaz *et al.*, "Cacto-bic: Scalable actor–critic learning via biased sampling and gpu-accelerated trajectory optimization," *Robotics*, vol. 15, no. 8, 2026.

[25] P. Lancaster, N. Hansen *et al.*, "Modem-v2: Visuo-motor world models for real-world robot manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 7530–7537.

[26] Y. Feng, N. Hansen *et al.*, "Fine-tuning offline world models in the real world," in *Proc. CoRL*, 2023.

[27] Z. E. Asri, I. Laiche *et al.*, "Rt-hcp: Dealing with inference delays and sample efficiency to learn directly on robotic platforms," 2025.

[28] S. Lavoie, C. Tsirigotis *et al.*, "Simplicial embeddings in self-supervised learning and downstream classification," arXiv:2204.00616, 2022.

[29] M. G. Bellemare, W. Dabney *et al.*, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2017.

[30] G. Williams, A. Aldrich *et al.*, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[31] D. Hafner, J. Pasukonis *et al.*, "Mastering diverse domains through world models," arXiv:2301.04104, 2023.

[32] Y. Tassa, Y. Doron *et al.*, "DeepMind control suite," arXiv:1801.00690, 2018.

[33] C. Sferrazza, D.-M. Huang *et al.*, "HumanoidBench: Simulated humanoid benchmark for whole-body locomotion and manipulation," arXiv:2403.10506, 2024.